ACOUSTIC ACTIVITY RECOGNITION

A PROJECT REPORT

Submitted by

ASHWIN K ASHOK [RA1511003010616]

Under the guidance of

Mr.R.SRINIVASAN

(Assistant Professor, Department of Computer Science & Engineering)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Kancheepuram District

MAY 2019

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled "**ACOUSTIC ACTIVITY RECOGNITION** " is the bonafide work of "**ASHWIN K ASHOK [RA1511003010616]**, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mr.R.SRINIVASAN
GUIDE
Assistant Professor
Dept. of Computer Science & Engineering

SIGNATURE

Dr. B.AMUTHA
HEAD OF THE DEPARTMENT
Dept. of Computer Science & Engineering

Signature of the Internal Examiner

Signature of the External Examiner

Own Work Declaration

Department of Computer Science and Engineering

SRM Institute of Science & Technology

Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course :

Student Name :

Registration Number :

Title of Work :

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ABSTRACT

Sound is a very rich source of information. Nowadays, because of the easy and cheap availability of audio recording devices we no longer have problem in the acquisition of audio data. However, computing devices that can listen to sounds in the environment via microphones do not use the audio to derive useful insights about their physical and social context. For example, a smart speaker sitting on an office desk cannot understand its location or environment and neither can it understand the activities being performed by the user. This project describes an audio-based activity recognition system. This project makes use of audio data from professional sound effect libraries which usually find their use in the entertainment industry. The audio data from such sound effect libraries are used to derive as well as generate required audio based activity recognition classes. The project makes use of Convolutional Neural Networks to learn from the available sound data. The project tests the developed system across a range of environments and device categories and shows that the already existing devices that have microphones built into them the requisite capability to execute audio based activity recognition that is comparable to human accuracy.

ACKNOWLEDGEMENT

I express my humble gratitude to Dr. Sandeep Sancheti, Vice Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support. I extend my sincere thanks to Dr. C. Muthamizhchelvan, Director, Faculty of Engineering and Technology, SRM Institute of Science and Technology, for his invaluable support. I wish to thank Dr. B. Amutha, Professor Head, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for her valuable suggestions and encouragement throughout the period of the project work. I am extremely grateful to my Academic Advisor Dr. K. Annapurani, Associate Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for her great support at all the stages of project work. I would like to convey my thanks to our Panel Head, Dr.C.N.Subalalitha, Associate Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for her inputs during the project reviews. I register my immeasurable thanks to my Faculty Advisor, Mr. R. Srinivasan, Assistant Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for leading and helping me to complete this course. My inexpressible respect and thanks to my guide, Mr. R. Srinivasan, Assistant Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for providing me an opportunity to pursue my project under his mentorship. He provided me the freedom and support to explore the research topics of my interest. His passion for solving the real problems and making a difference in the world has always been inspiring. I sincerely thank staff and students of the Computer Science and Engineering Department, SRM Institute of Science and Technology, for their help during my research. Finally, i would like to thank my parents, my family members and my friends for their unconditional love, constant support and encouragement.

-Author

Ashwin K. Ashok

RA1511003010616

TABLE OF CONTENTS

ABSTRACT **iv**

ACKNOWLEDGEMENT **v**

LIST OF TABLES **viii**

LIST OF FIGURES **ix**

ABBREVIATIONS **x**

1 INTRODUCTION **1**

1.1 Overview 1

1.2 Properties of sound 2

1.3 Categories of Sound Effects 3

1.4 Scope of audio Classification 3

1.5 Applications of Audio based Event Classifier(AEC) 4

1.6 Where does Audio based Event Classification (AEC) fit in the scheme of things? 5

2 LITERATURE SURVEY **7**

2.1 Existing Work 7

2.2 Survey Table 10

2.3 Inference from Literature Survey 12

3 ENVIRONMENT SETUP **13**

4 DATASET and DATA PRE-PROCESSING **17**

4.1 Sound Sources 17

4.1.1 Tune and Test Sets 17

4.2 Sound Pre-processing 18

4.2.1 Amplitude Based Augmentation 18

4.2.2 Persistence Augmentation 19
4.2.3 Augmentation by mixing sound effects 20
4.2.4 Augmentation by combining augmentations 20
4.3 Featurization 20
4.3.1 Short time fourier transform 21
4.3.2 Log-Mel Spectrogram 22
4.3.3 Featurization steps 22

5 MODEL **23**

6 EVALUATION **25**
6.1 Context 25
6.2 Classes 25
6.3 Test devices 26
6.4 Testing 27

7 RESULT **28**
7.1 Accuracy 28
7.1.1 Better evaluation of real world Accuracy 29
7.1.2 Comparison to human accuracy 30
7.1.3 Effect of sound data augmentation 30
7.1.4 Performance across platforms 31

8 CONCLUSION **33**
8.1 FUTURE ENHANCEMENTs 33
8.1.1 Accuracy 33
8.1.2 Simultaneous Events 34
8.1.3 Privacy 34

A PROJECT CODE **35**
A.1 Model training and testing 35
A.2 Audio classification web classification 43

B SCREENSHOTS **47**

LIST OF TABLES

2.1 Literature Survey Table. 10

4.1 Sound sources used before data augmentation 17

4.2 Overview of the datasets used 18

6.1 Devices and configurations used for testing 26

LIST OF FIGURES

4.1 Data augmentation based on amplitude 19

4.2 Data augmentation based on persistence 19

4.3 Data augmentation based on mixing sounds 20

4.4 Effect of windowing function on a sine wave 21

5.1 System architecture 23

6.1 Samples of the log Mel spectrograms of 960ms audio for our 30 test event classes 26

7.1 Summary of the accuracy of audio based classification model 28

7.2 Summary of the accuracy of audio based classification model 30

7.3 Accuracy across different devices 32

ABBREVIATIONS

IOT	Internet Of Things
AEC	Audio based Event Classification
SVM	Support Vector Machine
RF	Radio Frequency
CNN	Convolutional Neural Network
MFCC	Mel-Frequency Ceptral Coefficient
IAAS	Infrastructure-as-a-service
CPU	Central Processing Unit
GPU	Graphical Processing Unit
RAM	Random Access Memory
TPU	TensorFlow Processing Unit
STFT	Short-time Fourier Transform
API	Application Program Interface

CHAPTER 1

INTRODUCTION

1.1 Overview

Microphones can be easily found in many of the electronic gadgets of our day to day life. Microphones can be found inside smart home speakers, phones, tablets and Internet Of Things (IOT)-devices. This makes acquisition of sound an easy task. However, even with the easy acquisition of sound data, these modern computing devices equipped with microphones are not able to decode the happenings in their environment. This is a huge waste of data. For example, a smart speaker sitting on an office desk cannot understand its location or environment and neither can it understand the activities being performed by the user. Similarly a smartwatch which is worn on the wrist of the user is exposed to the sounds of various activities performed by the user who is wearing it like cooking, cleaning, coughing etc. Tapping into the enormous knowledge and insights presents in such audio data should enable us to improve the quality of human activities.

Humans are able to classify and recognize audio signals all the time without conscious effort. Humans can easily identify the difference between the sound of a phone a ringing from that of a door bell ringing. However, even for humans this task of recognizing sound becomes difficult when there is disturbing noises in the environment or when the the sound is weak. This project takes into to consideration all these real world problems and tries to recreate such situations by augmenting the sound data that is available.

Sound-based classification of activities is not something that is new. There have been efforts into task of classifying audio data into pre-defined classes. For example, ward et al. [21] developed a necklace with a microphone embedded in it. This was done to distinguish and recognize certain tools used by the user wearing the necklace based on the sound produced by the tools when they are being used by the user. However, in cases such as these the system is constrained to be used in a pre-defined environment and limited by the different class of sounds it can recognize.

This project seeks to build a system which has a wider scope of classification and able to recognize activities in a wider range of environments. The system developed can be deployed in an existing system without the need for any hardware change. This project takes existing sound effect libraries for developing the system. These sound effect libraries were used mainly because these are of high-quality and have very less noise in them and are also well labeled. These source of sound data are mainly used in the entertainment industry for post-production uses.

Another huge advantage of using such clean sound data is that, such sounds can also be modified and transformed to generate many variations of the same sound by adjusting and tuning some of the audio properties of these sound data. We can adjust audio properties like amplitude and also produce multiple variations in the data by mixing sounds with different types of background sounds. The important properties of sound effects libraries that make it a good sound data source are as follows:

1. **Atomic:** A sound labeled under a specific class like rain, door bell will only contain a clear and individual sound corresponding to that label.

2. **Pure :** The sound data found in sound effect libraries tend to be of high quality devoid of any disturbances and noise. Such pure sound data is highly desirable in producing set of new sounds that are obtained by transforming or mixing these pure sounds.

3. **Diverse:** These sound effect libraries have a collection of sound data that correspond to a very wide range of activities. Hence, these are very useful for developing a general sound based activity recognition system.

1.2 Properties of sound

1. **Scalable :** Sound is scalable and variations in both amplitude and duration can done.

2. **Transformable:** Sound can be transformed by changing its reverb, equalization(eq) and damping. In this manner a sound can be made to feel like having been recorded in a living room or in an auditorium.

3. **Additive:**Sounds can also easily added to form multiple variations of sounds. For example, a sound of a car can be combined with the sound of busy market or that of a waterfall to make it feel as if the car is running in either the market or near the waterfalls.

1.3 Categories of Sound Effects

The main categories of sound effects available are natural sounds, hard sounds and background sounds. Natural sounds are the sounds found in the environment like the sound of birds or that of the waterfalls. These sounds give a feeling a particular environment present. Hard sounds are the sounds that are our main concern in this project. Hard sounds are the sounds that are produced due to different activities like coughing, water flowing from the tap etc. Background sounds are generally noises. These are undesirable and we try to get rid of these sounds to obtain maximum accuracy.

1.4 Scope of audio Classification

Like all the other research works, audio based event classification problem should be first broken into smaller problems such that it can be solved easily and bit by bit at a time. These smaller problems are solved with the help of specifically designed individual systems. The result of smaller problems can be further connected together in future and can result in a multi-dimensional system, which can be helpful for sound classification. One of the most recent problems to be looked into as part of audio based classification is that of speech classification. Audio classification can be helpful in speech understanding and classification and some of the recent systems to consider this problem have shown to work really good for this task. On a similar context audio classification has also be looked into solve the problems of music understanding and composing based procedures. Audio classifications tend to work with models that tend to learn features of the audio signals and hence could prove to be useful in music classification and generation. This can be further extended to classify human accent and also solve the problem of distinguishing between human speakers. These are some of the fields in which the knowledge of audio based classification and understanding could prove to be helpful.

1.5 Applications of Audio based Event Classifier(AEC)

Some of the present and possible future applications of Audio based event classifier are listed below:

1. **Computing Tools :** Audio based event classifier can be used as computing tool for extraction of desired type of audio from various video sources. This can be helpful for building audio database for research work and other purposes like devising a speech recognition model and improving speech to text models.

2. **Consumer Electronics:** Sound can be transformed by changing its reverb, equalization(eq) and damping. In this manner a sound can be made to feel like having been recorded in a living room or in an auditorium.

   The application of many electronic devices can be improved by combining audio based event classifier with them. For instance, It can be combined with the Telephone or cell phone and help users to differentiate between type of responding signals. The combined device could inform users that if responding signal is coming from fax-machine, an answering-machine, human or computer-generated. Based on this, automatic responses can be generated from the combined device instructing the telephone or cell phone to hold, connect, hang-up, redial, etc.

   Audio based event classifier can also be combined with television and radio. In television, the classifier can detect the advertisement and respond to it automatically by muting the sound, surfing different channel or by blanking out the screen and resuming to same channel when advertisement ends. When combined with radio it can be used search the desired signal like weather report, music etc.

3. **Automatic equalization:** Sounds can also easily added to form multiple variations of sounds. For example, a sound of a car can be combined with the sound of busy market or that of a waterfall to make it feel as if the car is running in either the market or near the waterfalls.

Applying an array of band pass filters, in parallel, to a sound signal, is equalization. Relative amplification or reduction of signal in each filter range or channel leads to signal modification. This process deliberates an equalized signal of perceptual quality than the indigenous signal by a higher margin.

The main problem with existing process is that it is manually controlled and hence prone to many mistakes. Thus, there is need of automatic equalizer that can give efficient results. Automatic equalizer detects the type of signals and adjust its setting to improve and give most enhanced version of signal. Devices exist today that identify feedback in public address systems, and use an equalizer to attenuate the channel or channels where the feedback is happening.

Hearing aids and public address systems make use of equalization filters. There are numerous situations based on which possible filters are incorporated in modern hearing aids. The user must manually alter the settings but, detection of current environment is possible by an automatic equalization system. The required pre-fabricated setting can be applied or a separate environment can be initiated specific to user requirements.

1.6 Where does AEC fit in the scheme of things?

AEC comes under the wider research field of Auditory scene understanding and knowledge analysis. Under this research field, Audio based event classification provides the solution to problems that involves understanding a sound and classifying it under a specific category.

The research in the field of Auditory scene understanding and knowledge analysis has many parts of prepossessing. The audio based event classification related work has brought improvements in signal segmentation and data augmentation. These works have proved to be of great use in the overall filed of Auditory scene understanding and knowledge analysis ans its subsequent applications. The works related to Audio based event classification has brought about important improvement in noise filters as well. These filters have helped in improving the accuracy of event and activity understanding.

Polyphonic signals which contain multiple audio streams are considered for Audio based event classifier research. Polyphonic signals like auditory scene ordinarily arise in natural environment. It is salient to pursue this study regardless of advanced stream separation, as it is probable that stream separation will be a computationally acute process. It would be informative to determine the contents of auditory scene ahead of trying to segregate it.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing Work

ubi:- There are many ways to recognize and understand human activity. Some these methods make use of special types of sensors likes geophones and Radio Frequency (RF) tags. There are also computer vision based methods that try to recognize human activities. However, this project is concerned with work related to recognizing and understanding human activities based the unique sound generated by each of these different activities.

Machine learning models have long been used on top of publicly available sound datasets to implement audio based event classification. For example, Salamon et al. [17] used a scattering transform for urban sound classification which included sounds like car horn, birds etc. However foggia et al. [8] used an approaach similar to bag of words along with multi-class Support Vector Machine (SVM) for recognizing sounds that would help with surveillance. These sounds mainly included the sound of people screaming, gunshot, glass breaking etc.

Sound data has also been used with Markov models for developing audio based event recognition. For example, Eronen et al. [7] made use of hidden Markov models and clustering techniques for audio based context and scene understanding. This included understanding vehicle related sounds and some general sounds in the common environments like closing of doors at home etc.

With the recent advancements in the field of Artificial intelligence, deep learning models have become prominent in most of the advanced use cases. Deep learning based models have also been applied to sound based event classification. For example, Lane et al. have made use of neural networks with many layers for audio classification for the DeepEar [10] study. This work has contributed extensively to the current state of audio based classification works. SoundNet [2] is another such work which has been of great significance in towards audio based scene and event recognition. This work uses video data along with computer vision to recognize

the objects in a scene and with the help of the label corresponding to the action for a specific time duration in the video is used to learn the sound.

Convolutional Nerual Networks are good in extracting features directly from raw input and this has been verified from tasks corresponding to image recognition. Based on this understanding, the work in [**?**] tries to use Convolutional Neural Network (CNN) to obtain features from raw sound data and then use SVM classifier to do audio classification. However, certain studies have tired to investigate into the use of multiple layers in a Convolutional Neural Network for audio based event recognition. For example, the work in [**?**] is one such work. This work on audio classification tires to study the effectiveness of multiple layers in audio classification. With multiple experiments, it was found out that having higher number of layers don't guarantee a better model in audio classification.

The traditional Convolutional Neural Networks tend to have drawbacks which make it difficult be implemented for the purpose of audio data classification. The main problem in using traditional Conolutional Neural Network for audio based event classification is that these traditional neural networks use pooling layers for feature dimension reduction and these can lead to loss of information and affect the accuracy of the classifier.

Ephrat et al. [6] and Phan et al. [15] have also used Convolutional Neural Network based audio recognition system on very specific set of sounds and were successful with a specific domain based sounds. Uses of other types of neural networks have also been seen in recent times. Another neural network that can bee seen in audio classification and event recognition is Recurrent Neural Network. However, use of Recurrent Neural Network has proved not to improve the performance to very extent. Rather, in cases other than a specific domain of sounds, these Recurrent Neural Networks have in some cases performed poorly to Convolutional Neural Networks. However, Parascandolo et al. [14] uses a combination of Convolutional Neural Networks and Recurrent Neural Network and has managed to get slightly improved performance. Hershey et al. [16] tries to run different architectures of Convolutional Neural Network on publicly available sound datasets.

From many of the works seen till now, we find that all the models that have been investigated into require adequate amount of good quality data. This matter has made people to look into data augmentation. With data augmentation we try to improve data space by bringing about realistic variations in the already existing data set. For instance, McFee et al. [3]

implemented data augmentation by adding background noise to the existing sound data. They also brought about multiple variation in sound by shift in pitch and also by dynamic range compression. Similarly, Salamon et al. [17] implemented a similar kind of data augmentation and then implemented Convolutional Neural Networks to do sound based event classification and this proved to be a good way of improving performance of the model. There has also been work on real-time activity tracking based on sound. For instance ShotSpotter [1] try to recognize the sound of gunshot and tries to help with surveilance. There are also work that have been done in the pre-processing stage of audio classification. For instance, Stork et al. [18] used Mel-Frequency Ceptral Coefficient (MFCC) along with non-Markovian ensemble voting to recognize and distinguish between several human activities . Similarly, Lu et al. [11] showed sound detection using microphone for speech, music and ambient sound detection. There have also been work done on portable systems and certain systems that can be worn on the body for recording vibrations. There are platforms like the Mobile sensing platform [4] and also Bodyscope [19] that makes use of such portable devices. There are also works carried out to incorporate accelerometers to better understand the output of such portable systems. The work carried out by Ward et al.[21] makes use of such portable devices. However such systems are been found out to be good only with very specific set of activities and have been found to be not so good for general activity recognition.

The task of audio classification is especially difficult because sound generated from set of random events like car horn, mobile ringing etc are very different and produce random patterns of sound and are not as structured in human speech in terms sounds being produced. Also, the sound produced by such randoms events in the environment tend to show a much wider frequency and also a wider range in amplitude and this is mainly because these sound events are not produced by human vocal system and hence they are not constrained by the limitations of the human vocal system.

In the field of audio based event recognition we can also find works that try to make use of spectrograms. For instance, in [20] it is seen that good performance could be achieved by using auditory images and spectrogram image features. Feature vectors are deduced from such representations and are used in with machine learning techniques. An evaluation and benchmarking of such tasks can be found in the work done with Dennis [5].

2.2 Survey Table

Table 2.1: Literature Survey Table.

Objective	Methodology	Implementation	Results	Limitation	Future Work
1.Feature learning with Deep scattering for urban sound analysis.	Use scattering transform of a signal instead of spectrogram	Implemented CNN with scattering transform of a signal	Good output. Model learns the pattern from the scattering transform	Using scattering transfrom with CNN gave poor performance with noisy data	Use RNN with raw input to understand and map the signal
2.Activity recognition of assembly tasks using body-worn microphones and accelerometers	Machine learning for raw audio input	Use audio data from microphone and accelerometers with machine learning for audio classification	Classification output is calibrated using the data from accelerometers.	Models effectiveness is dependent on the features selected for training the machine learning model	Better featuraization for better models. Use deep neural networks.
3.Deep convolutional neural networks and data augmentation for environmental sound classification	Audio data augmentation for better output	Data augmentation using fundamental properties of sound and comparison with non-augmented data based models	Data augmentation proves to an effective way to overcome lack of sound data. The model learns better with better quality of data	Over fitting is an issue with excessively augmented data	Data augmentation using different environmental noises. This the real-world performance of model as noise is inherent in the environment.

Objective	Methodology	Implementation	Results	Limitation	Future Work
4. CNN architectures for large scale audio classification	Test different CNN architectures for audio classification	Draw inference from output of each CNN run on the same audio data set	CNNs are good for audio classification. High layers don't always lead to better performance.		
5. SoundNet: Learning sound representations from unlabeled video	Learn from audio extracted from video	This work uses video data along with computer vision to recognize the objects in a scene and with the help of the label corresponding to the action for a specific time duration in the video is used to learn the sound	Model's scope of classification is very wide, however performance is average	Very computationally intensive. Performance depends on the amount of data segmented and extracted from videos for each event being considered for classification.	
6. Reliable detection of audio events in highly noisy environments	classification of real world noisy data using machine learning	Mulit-class SVM over noisy data	Good performance. The model is able to learn the noise in the environment really well and isolate it for understanding the event under focus	Event classification is affected by simultaneous occurrence of multiple events	Use deep neural network for better pattern detection. Use data augmentation to improve data
7.Convolutional recurrent neural networks for polyphonic sound event detection	Classification of human activities using CNN and RNN	Spectrogram of sound input is given to CNN to better understand local context and then combined with RNN to understand the pattern	Better output as compared to average CNN architectures	Computationally intensive. Needs large amount of data. With RNN combined with CNN, there is increased chances of overfitting	Data augmentation to improve data quality. Using pretrained CNN networks for training.

2.3 Inference from Literature Survey

Many similar attempts have been carried out by researchers in the past that have resemblance to our work for purpose of industry as well as academia. Many features of the proposed model are directly referred or taken from these research works. For example, the work of Ward et al. [21] implements a necklace with microphone embedded inside it and performs audio classification to distinguish between the workshop tools being used by the user who wears the necklace. Similarly, the work of foggia et al. [8] uses an approaach similar to bag of words along with multi-class SVM for recognizing sounds that would help with surveillance. These sounds mainly included the sound of people screaming, gunshot, glass breaking etc.

All the research works studied and mentioned above were mostly focused on specific set of sound data. Also, these methods do not look into the pre-processing of the raw input data. The methods seen till now can be used as a method to build upon for developing better sound classification systems, however none of these systems can be implemented as a general purpose audio based event classification system. We need to look into tuning our deep-learning models based on effective transformation and augmentation of data. We also need to understand about cross-platform performance to further tune the model such that we can create a general purpose audio classification model.

The research papers have helped us to gain the thorough insight about audio classification models that have existed till. After reviewing the above mentioned papers, it can be inferred that good quality audio data is important and data augmentation implemented over multiple sources of audio data is good way for doing so. Also, we learn that converting udio data as spectrograms gives better performance when used with Convolutional Neural Networks, rather than using simple raw audio data. In this project we aim to build on top of these insights to develop our audio based event classification system.

CHAPTER 3

ENVIRONMENT SETUP

The project is made of many modules. However in terms of development, the project can be divide into two many parts. First part is mainly concerned about the development of the deep-learning model and its corresponding pre-processing and execution. Second part of the project is concerned about the deployment of the developed deep-learning model as a web service. The development of these two parts of the project used many software tools and applications and the most important of these are as follows:

1. **Google cloud platform**: Google Cloud Platform is a combinations of wide range of cloud computing services offered by Google. The platform consists of multiple hosted services for on-demand computation, storage services and application development and deployment which all are run on top of Google hardware. In this project, we are making use of Google clod platform for providing us compute power in the form of Virtual machines and also for storage space. For the compute part, we have used Google compute engine to deploy virtual machine with the reacquired computation configuration. The Google compute engine an Infrastructure-as-a-service (IAAS) that provides cloud based computation resources as a service.The virtual machine deployed using this Infrastructure-as-a-service is the fundamental resource that is powering the computation needs of the entire project. The virtual machine is the compute resource that consists primarily of the Central Processing Unit (CPU), Graphical Processing Unit (GPU) and Random Access Memory (RAM). The virtual machine for this project is configured with the following resource:
   a.CPU: 8-core skylake processor
   b.GPU: Nvidia Tesla K80
   c.RAM:16 GB

The deployed virtual machine in this project is initialized with ubuntu 18.04.

The Google cloud platform is used for cloud storage services for enabling in-cloud storage and processing of data. This project uses 120 GB of Solid state drive(SSD) attached to the custom configured virtual machine.

2. **Anaconda**: Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing like data science, machine learning applications, large-scale data processing, predictive analytics, etc. It is mainly popular because of its simplifed package management and deployment. Anaconda has an internal package management system called "Conda" and is responsible for package handling and versioning. The Anaconda distribution is hugely popular among all type of python based scientific computation users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS.

   Anaconda is one of several Python distributions. Anaconda is used for scientific computing, data science, statistical analysis, and machine learning.The package manager is also an environment manager, a Python distribution, and a collection of open source packages and contains more than 1000 R and Python Data Science Packages.

   Anaconda is mainly helpful in following tasks:

   1.Installing Python on multiple platforms

   2.Separating out different environments

   3.Dealing with not having correct privileges

   4.Package management

3. **Python virtual environments**: Python virtual environments is to create an isolated environment for Python projects. This enables each project to have its own dependencies and is not be isolated from other projects and its libraries.We can create mupltile environmetns as needed using command line tools like virtualenv, pyenv or conda.

   In this project we have created a single isolated environments that is then populated with the required deep-learning based libraries and packages. The created environment is also

populated with flask library and its dependencies for enabling web services based upon the developed deep-learning model.

4. **Tensorflow**: Tensorflow is a library developed by Google. Tensorflow isa library which is used for numerical computation and large-scale machine learning. It uses Python to provide a convenient front-end Application Program Interface (API) for building applications with the framework, while executing those applications in high-performance C++. Tensorflow has shown to give good performance and supports multiple GPUs and CPUs. Tensorflow also provides tools for tuning a network and monitoring performance like Tensorboard.

   TensorFlow can be seen as primary means to train and run deep neural networks many advanced deep-learning based tasks.TensorFlow allows developers to create dataflow graphs-structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or tensor.

   TensorFlow provides all the functionalities to the programmer through Python language. However the actual math operations are not performed in Python. The libraries of transformations that are available through TensorFlow are written as high-performance C++ binaries. Python just directs traffic between the pieces, and provides high-level programming abstractions to hook them together.

   TensorFlow applications can be run on most any target that's convenient: a local machine, a cluster in the cloud, iOS and Android devices, CPUs or GPUs. If you use Google's own cloud, you can run TensorFlow on Google's custom TensorFlow Processing Unit (TPU) silicon for further acceleration. The resulting models created by TensorFlow, though, can be deployed on most any device where they will be used to serve predictions.

5. **Keras**: Keras is a python library that utilizes tensorflow or theano as its backend. It provides an high-level API over neural networks.It was developed to make implementing deep learning models as fast and easy as possible for research and development. Keras was developed to make the usage of neural networks very user friendly. Apart from

being ease to use and also learn, keras is really powerfull in developing and trainig neural network based models. Keras supports both CPU and GPU based proccessing and also supports distributed model training.

Keras, having its backend supported by either Theano or Tensorflow, doesnot do any of the low-level operations. Whenever a task is provided to keras, it converts the given input into a form that is understandable by the underlying backend and makes the backend execute the task.

6. **Flask**: Flask is a web framework. Flask provides us with the tools, libraries and technologies that allow to build a web application. Flask is a micro-framework in Python. It is ligth-weigth framework that allows for rapid development and deployment of web applications.

   Flask being a micro-framework has very little dependency to external libraries. This is what makes the framework to be light-weight and with very little dependency , we only have to look out for very few external package updates. However, with flask being a light-weight frameworks, makes the developer to have to do a lot of the tasks manually and this can bring in the need to introduce multiple plugins.

   The major dependencies of flask are: Werkzeug a WSGI utility library jinja2 which is its template engine

7. **Jupyter notebook**: The Jupyter Notebook is an open-source web application that is used to create and share documents that contain live code, equations, visualizations and narrative text. Jupyter notebook make it easy to implement visualizations and share code and also provides for online collaboration. In this project Jupyter notebook has deployed in the virtual machine inside Google cloud and is providing us with an online interface to write code and execute it without having to take direct connection to the terminal of the virtual machine.

CHAPTER 4

DATASET AND DATA PRE-PROCESSING

4.1 Sound Sources

There are multiple sound libraries available. However, we have used the libraries mentioned in the Table 4.1. The data in these sound libraries are of good quality and have been used for tasks in the field of audio based application.

Table 4.1: Sound sources used before data augmentation

Name	Total sounds	sounds used	Hours used
BBC Sound Library	29K	740	1.9
Network Sound Library	10K	492	1.3
Soundsnap	250K	4072	10.4
Freesound	372K	8929	22.2
AudioSet	200K	7899	18.2

4.1.1 Tune and Test Sets

The project uses the sound data mentioned in Table 4.1. From these sources of sound, we have taken out those sound data that covers the 30 classes of sound that were of Interest. We break this entire data set into training data and test data. They are names as SFX-original and SFX-Test. The project also builds the separate augmented data based on amplification, persistence of sound and mixing. The data augmentation based on amplitude is called SFX-AMP. The data augmentation based on persistence of sound is called SFX-PERSIST and the sound data augmentation based on mixing is called SFX-MIX. A separate dataset consisting of all the augmentations made is also formed and is called SFX-ALL. The table 4.2 gives the general information of the different groups of data being used.

Table 4.2: Overview of the datasets used

Name	Contains	Data hours
SFX-ORIGINAL	Processed nut not augmented data	54.6
SFX-AMP	SFX-ORIGINAL + Amplitude based augmentation	152.0
SFX-PERSIST	SFX-ORIGINAL + Persistence based augmentations	136.2
SFX-MIX	SFX-ORIGINAL + Mixing based augmentation	300.6
SFX-ALL	SFX-ORIGINAL + SFX-AMP + SFX-PERSIST + SFX-MIX	479.5
SFX-TEST	Unaugmented sounds	8.8

4.2 Sound Pre-processing

For pre-processig of the sound data, we first bring all the sounds or audio data into the same format. This is done to create a single processing pipeline that wouldn't need any intermediate processing because of the difference in format. The project uses 16khz as the standard format. Also, as part of the pre-processing we have also we remove the existence of minor or inaudible frequencies that exist for a duration greater than two seconds. This gives us a set of standard sound data. However to further improve the quality of data we do data augmentation. The following types of data augmentation have been implemented for the sound data:

4.2.1 Amplitude Based Augmentation

By bringing about a change in amplitude we get a new sound data which is different form the existing one. Here, by doing data augmentation based on amplitude we are creating two separate sound data. One of this has a lower amplitude which results in quieter sound data and the other leads to a louder sound data due to higher amplitude.

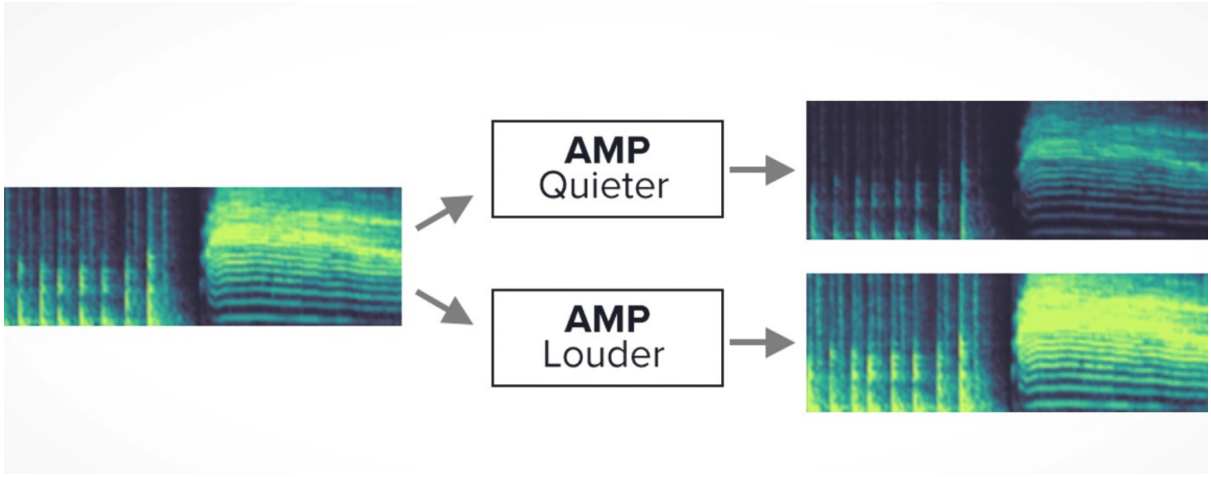

Figure 4.1: Data augmentation based on amplitude

4.2.2 Persistence Augmentation

Persistence of a sound data is an attribute that includes reverberations and non-linear damping. By changing these attributes we can simulate a change in the physical environment of the sound.

For producing persistence transformation, we use high quality reverb effects that are provided with Adobe Audition. Apart from these we create custom reverbs by capturing impulse functions from spaces like bathrooms, small office area, auditorium etc.

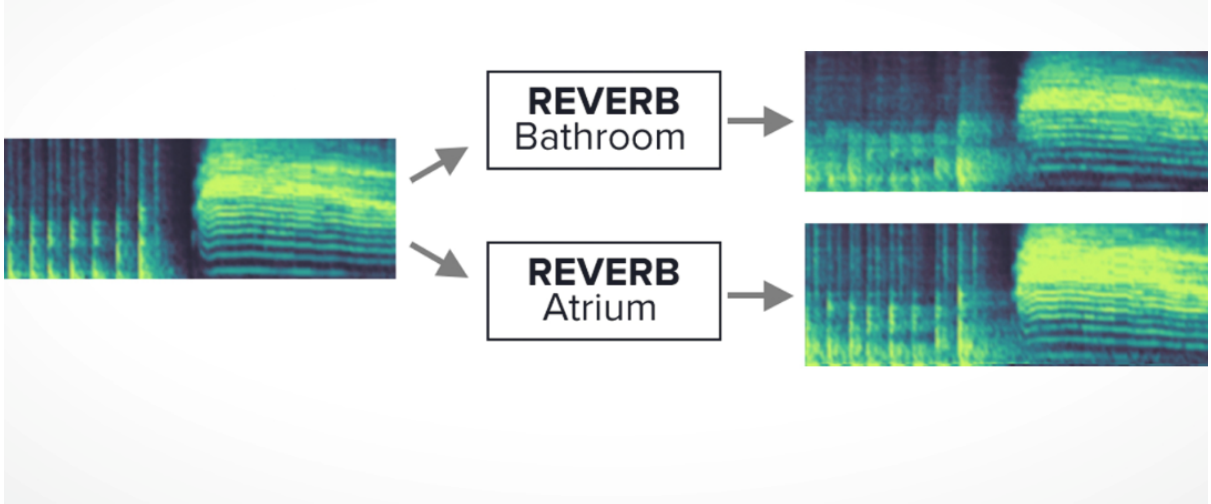

Figure 4.2: Data augmentation based on persistence

4.2.3 Augmentation by mixing sound effects

In this data augmentation we aim to produce new sound effects that are generated by mixing two different sound effects. However, in this mixing process we concentrate mainly on the adding general activities with background noises like that of waterfalls or traffic noise. The main aim of augmenting the data in this way is to generated important sound variations on of events and hence enable the model to learn and understand how different events have different attributes when present in other environments.

Figure 4.3: Data augmentation based on mixing sounds

4.2.4 Augmentation by combining augmentations

This type of data augmentation involves combining the different types of augmentations with different sound data and hence create much greater variety in the data. For instance, we can take the sound of someone coughing and make its amplitude higher and then mix this augmented data with auditorium-like reverberation and further improve the variety and complexity of this sound by mixing it with some background noise like that of the ventilation system. However, this method needs to be done systematically and logically as the wrong type of augmentation and manipulation could result in a drastic change in the main sound event in focus.

4.3 Featurization

The featurization process for audio data is different as compared to other types of data. The featurization in this case of audio data uses the following:

4.3.1 Short time fourier transform

Short-time Fourier Transform (STFT) is obtained by taking Fourier transforms in sequence of windowed signal. Short-time Fourier transform gives us the time-localized frequency information for situations in which frequency components of a signal vary over time, whereas the standard Fourier transform provides the frequency information averaged over the entire signal time interval.

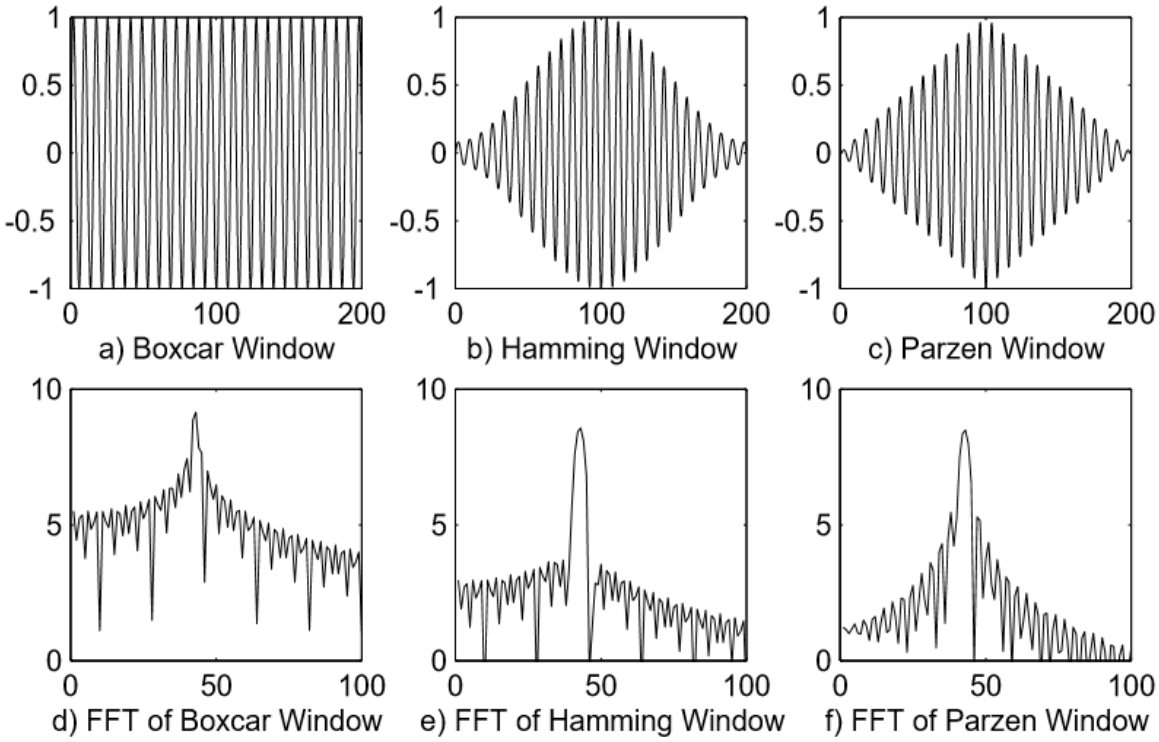

Figure 4.4: Effect of windowing function on a sine wave

The classic Fourier transform detect the presence of particular frequency in a signal. The main disadvantage with this classic Fourier transform is that it only work well with the signals of relatively less frequency changes but create problem when a signal of large frequency spectral is encountered. For example, a song is a kind of signal that contains different notes and hence different frequencies. The Fourier transform will get confused between so many frequencies and will show wrong information about the different frequencies that are present.

The above mentioned problem can be solved by STFT. The input signal with large frequency spectrum can be segmented sequentially into small frames or windows of signal. After segmentation, Fourier transform is applied to every window or frame sequentially. The output

of each frame represents the change in frequency spectrum with the time, such that there is no confusion and we can have a clear time-dependent depiction of change in frequency of an input signal.

4.3.2 Log-Mel Spectrogram

2003-07:- The spectrum of a signal gives us the distribution of frequencies in that given signal. From [12], we understand that humans perform some kind of spectral analysis during hearing, hence we try to implement the spectrogram as a way to design an approach that learns from the spectrum of audio signal.

The spectrogram is the time-varying spectrum of a signal. To produce a spectrogram, a given signal is first segmented into frames, then the spectrum is calculated on each frame and then these spectra are shown as a time-varying spectrum. This gives us a way to measure the frequency content of the signal changes over time. Several physical features of the spectrum of a given signal could be used for classification. However, in this project we feed the spectrogram to a neural network capture the features and hence learn from the spectrogram.

4.3.3 Featurization steps

For doing so, we first break the data into smaller segments where each segment is 960 ms. Then we compute the short-time fourier transform of each segment. This uses a step size of 10 ms and a window of 25 ms. The short time fourier transform produces a 96 length spectrogram. This spectrogram is then converted into a 64-bin log-scaled Mel spectrogram and generate a 96x64 input frame for every 960 ms of audio. This is then given to the classification model.

CHAPTER 5

MODEL

After featurization step is completed, we obtain the data in an appropriate form such that it now be feed to our neural network. CNN or ConvNet is an artificial neural network that has been so far used popularly for analyzing images. Apart from image analysis, CNN can be used for classification problem and data analysis. The CNN is an artificial neural network which specializes in identifying or detecting patterns and make interpretation from them. This pattern detection ability is what makes CNN so effective for image analysis. In this project we use CNN to analyze and learn from spectrogram. A spectrogram is really a special image containing different patterns, many of which exhibit local relationships but only weak absolute locality, i.e. a recognisable sound event may appear at different times and in slightly different frequency ranges.

In the figure 5.2, the different steps used in our approach are shown. The Convolutional neural network is applied after the Log Mel Spectrogram is generated.

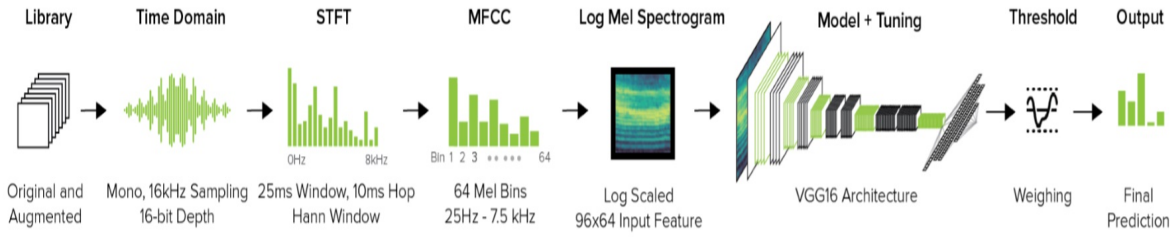

Figure 5.1: System architecture

A CNN consists of many hidden layers known as convolutional layers, these layers take in the input, then modifies the input in some way and then output the modified input to the next layer. This modification operation is known as a convolution operation. The number of filters that each convolutional layer has should be defined. These filters present in convolutional layer detect the patterns present in the input. The pattern could be edges, corners, etc. Some filter may detect edges (vertical, horizontal, diagonal), some may detect corners, some may detect texture, etc. As the depth of the network increases the filters become more and more

sophisticated. In deeper layers instead of just edges and simple shapes the filters are capable of detecting even more sophisticated and advanced patterns as a whole.

Here we are using the VGG16 architecture for training and testing our approach.VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.It makes the improvement over AlexNet by replacing large kernel-sized filters with multiple 3x3 kernel-sized filters one after another. The architecture contains four convolutional layers (3x3 kernel, step size = 2, depth = 64, 128, 256, and 512, ReLU activation [13]), with intermediary max pool layers, and a 128-wide fully connected embedding layer. We modified this pre-trained model by removing the last fully connected layer and replacing it with our own fully connected layer, using a sigmoid activation function. Finally, we tune the entire architecture with our sound effect datasets.

CHAPTER 6

EVALUATION

This evaluation aims to understand the performance of the classifier system developed for audio event classification. This also tries to understand the improvement in performance due to data augmentation.

6.1 Context

For our evaluation, we selected seven location contexts in which everyday activities occur: 1) bathroom, 2) bedroom, 3) house entrance, 4) kitchen, 5) office, 6) outdoor and 7) workshop. These contexts offer realistic scenarios with constrained event classes. For instance, it is highly unlikely for a blender" event to occur in a bathroom, or for "chopping" to happen in a bedroom. This permits us to tune models for particular contexts with more tractable class sets. As part of the evaluation, we we consider seven location contexts where our day-to-day activities mainly tend to occur.

6.2 Classes

For each of the contexts under consideration, we tried to select event by looking into facts like the likeliness of such as event occurring in such a context, the frequency of the event occurring in that context and also the usefulness of the information regarding such a event occurring in that context. By doing so, we ended up with 30 events across the select contexts.

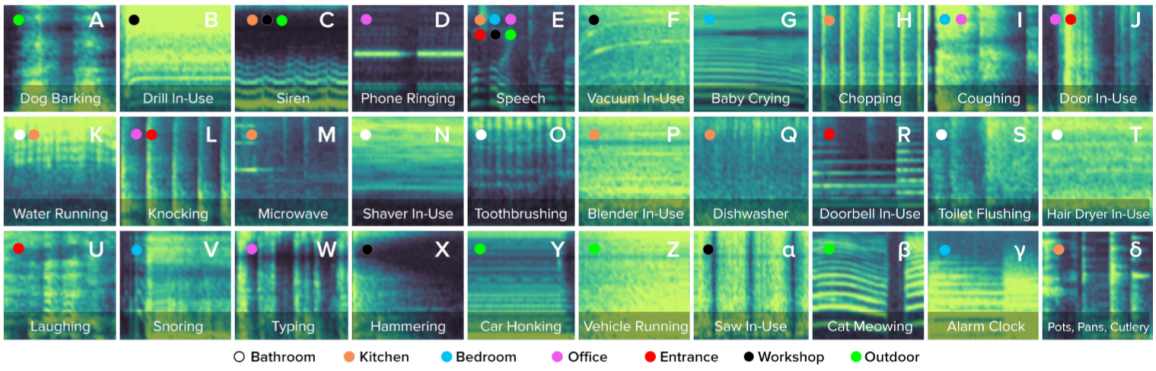

Figure 6.1: Samples of the log Mel spectrograms of 960ms audio for our 30 test event classes

6.3 Test devices

In order to test the robustness of our models across different microphones, placements, and platforms, we developed software to capture and stream audio from a diverse set of hardware platforms (Table 3). These devices connected over various means to an independent laptop capable of recording synchronized streams and performing live classification.

In order to understand the performance of the model across different microphones across different platforms and scenarios, we created programs that help us to record and stream audio from different types of devices equipped with a microphone. The Table 6.3 gives the list of test apparatus being used. The devices used in these tests communicate with a laptop that has the classifier model running in it and tries to provide instantaneous classification results based on the input audio that is captured by these devices.

Table 6.1: Devices and configurations used for testing

Device type	Implementation	Communication type
Smartphone	iphone with an ios app	wifi
Smartwatch	LG smartwatch with android app	Bluetooth
Smart speaker	Google home	USB
IOT sensor	Raspberry pi with microphone	Bluetooth

6.4 Testing

The model is tested on the SFX-TEST dataset to analyse the performace of the model. However, to further validate the test we test the model on random sounds captured from muliptle common environments. These tests were carried out by multiple participants of the test. These participants helped in testing the model across different types of rooms and outdoor places. These participants also had helped in testing with at least two types of devices. Most of these participants had a smarthpone and also helped with recording sound using theIOT device given to them. These devices used had a simple interface that made this process easy to execute. The interface of all these devies were made such that it is easy to capture and label the data.

The data is collected from participants for obtaining a set of data that is not modified or augmented in any way. Also, these data were collected without modifying the surroundings in anyway and can be considered as data collected completely in the wild.

For each location, the test acquires at least three samples of data for each event. For instance, for an environment like bathroom and an event like water flowing from the tap, we try to take three samples of the same event in the same environmental conditions. The data collected were mainly obtained as part of the daily work routine of each participant. This helped in making sure that the events recorded were as natural as possible in terms of their occurrence during the day and this also made sure the data had real world noise in it. Having the noise captured from the environment as part of the testing is a great test of model's ability to react on real world data.

CHAPTER 7

RESULT

The results of the experiments conducted are discussed here. The results of these experiments have been summarized in the Table 7.1. Here we first look into the model tested on the data present in our hands. We then look into the performance of the model based on the data provided by external participants and the performance analysis based on human annotated data will help us understand the adaptability of the model in real world situation. As part of this analysis, we also look into the effects of the augmentations and modifications that were done on to the data.

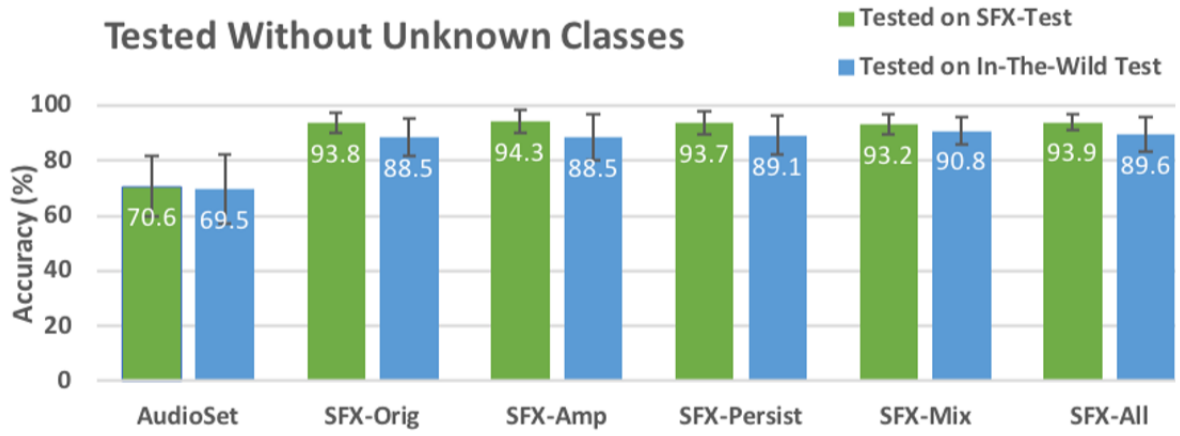

Figure 7.1: Summary of the accuracy of audio based classification model

7.1 Accuracy

For calculation of accuracy, we find the predictions that the model gives out. The output given out by the model is the best predicted result based on cumulative confidence.

The model when tested on SFX-ALL and SFX-TEST, we achieved an accuracy of 93.9%. When tested on the data obtained by external participants, the average accuracy was lower and equal to 89.6%. However, when we the model is tuned on the dataset named "AudioSet", the

system achieves a relatively very less accuracy of 70.6% and 69.5% on SFX-TEST and on the "In-the-wild" dataset respectively. This result proves that tuning the model on the sound effects based dataset with clear and atomic audio data has provided significant boost to the accuracy of model.

For comparison of our model performance we also look into the models developed for work related to audio based classification.SoundNet [2] is a good example of audio classification. In soundNet, the model is tested across 10 classes and it achieves an accuracy of 88% and to compare it with our model, we see that our model gives an accuracy of 82.1% on 30 classes.Finally, in the activity recognition domain, BodyScope [19] used a wearable necklace to achieve a recognition accuracy of 71.5% (tested on in-the-wild data) across four activities (eating, drinking, speaking and laughing). Similarly, SoundSense [9] recognizes three classes (speech, music and ambient sound) with an accuracy of 84%. We provide comparable accuracies, while offering a much richer set of activities without requiring any in situ training.Another work relevant to audio classificatio is BodyScope [19]. In Bodyscope, a necklace is worn by the user and this necklace has a microphone for audio acquisition and hence helps in audio classification across four different activities. The bodyscope shows an accuracy of 71.5%. In a similar case, we find that SoundSense [9] is able to classify audio into three classes with an accuracy of approximately 84%. We can see that our model provides comparable accuracies and also provides a better and larger set of activities

7.1.1 Better evaluation of real world Accuracy

The 89.6% accuracy obtained for this model is a value corresponding to standard evaluation procedures and might not be the best of the measures to be considered for the understanding the real world performance of the model. In real world situations there are chances of the model being exposed to sounds that were never introduced to the model. Evaluating the model based on test data that contains only known classes doesn't seem to be the best measure of real world performance. As a measure to overcome this, we formulated an experiment that induces nearly 20% of "Unknown classes" in to the the test data. By doing this we try to see how effectively the model is able to understand a data sample as "Unknown class" and ignore it. Here the model identifies a sample as "Unknown class" when none of the classes being considered apart from "Unknown class" crosses over the minimum confidence threshold. Using this method over the

"In-the-wild" dataset, we see the accuracy to be of 80.4% and this can be considered as the nearest estimate of the real world performance of the model.

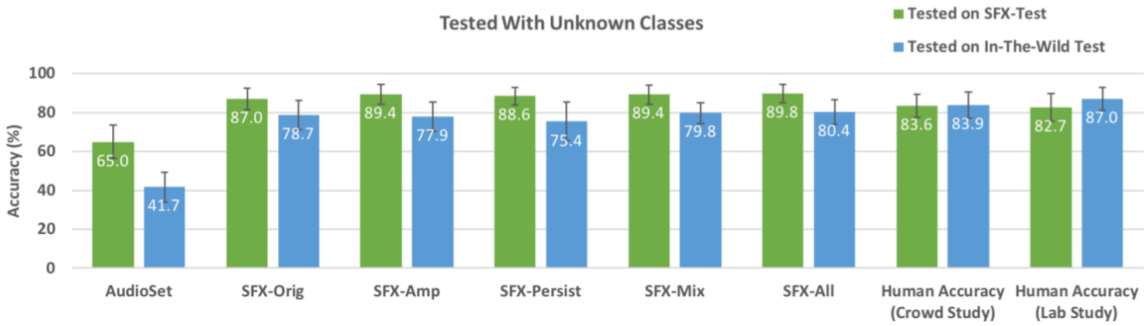

Figure 7.2: Summary of the accuracy of audio based classification model

7.1.2 Comparison to human accuracy

The goal of any model would be to reach the level accuracy equivalent to humans. Comparing the model predictions to that of human judgments is a good way to understand the performance of the model. For doing this we use two methods, first method is to run a crowd-sourced study and the second method, is by testing the model using participants in the lab. The second method of using participants in the lab was to create a dataset with reduce errors in the form of malicious or apathetic crowdworkers giving wrong inputs into the data collection process as part of the first method.

For the crowd-sourced study we used the Amazon Mechanical Turk. For both the above mentioned methods, we use a common web interface to collect data from people. The interface is simple and plays a sound and provides the user with a drop down menu to select a label that the user feels to be the best description of the played sound. The sounds played in these two methods were part of the SFX-TEST dataset. The accuracy of the model came out to be 83.6% for our crowd-sourced study and 82.7% for the lab based study.

7.1.3 Effect of sound data augmentation

The experiments run till now were based the on dataset SFX-ALL and this contains both the original data and also the augmented data. In order to understand the effect of sound data augmentation, we run the same tests over the four differet datasets which contains the different

types of data augmentation. These datasets are SFX-AMP which contains the augmentation based on amplification, SFX-PERSIST which contains data augmentation based on persistence, SFX-MIX which contains data augmentation by mixing of multiple sound sources and SFX-ALL which is the dataset that contains all the data from all other types of data augmentation. We then run the same set of experiments over the non-augmented dataset which is called SFX-ORIGINAL. We then compare the results of the model to check if augmentation proves to be a effective way to improve performance.

By doing this experiment, we notice that with the current data that we have, data augmentation proves to be an effective way to improve the performance of the model. Upon comparing the non-augmented dataset performance with model run on SFX-AMP dataset, we see that the model tuned on dataset that is augmented based on amplification helps in improving the performance by 1.6%. By doing similar comparison we observe that SFX-PERSIST provides an improvement of 0.4% and SFX-MIX provides an improvement of 1.8%. Also, by comparing the model tuned using SFX-ALL, which contains the mix of all types of data augmentations, we see that all the different types of data augmentations together provide an overall improvement in performance of about 2.4% as compared to using the original dataset which did not contain any augmentation at all.

7.1.4 Performance across platforms

We also try to understand the performance of the model across different platforms. For doing this we test the model across different devices. We observe that the microphone in the laptop provides the best performance with accuracy 86.1%. The device with second best accuracy turns out to be the smart watch which is providing an accuracy of 84.1%. We also observe that the IOT device provided the poorest performance. The performance across multiple devices is shown in Figure 7.3. We observe that the performance of the system depends not only on the parameter of the model but also on hardware and external factors. Here the key hardware being used is the microphone and the tests show that the quality of the microphone effects the audio acquisition and hence can affect the performance of the system. Also, the experiments show that the location is also a factor that affect the performance of the model. Here we see that the devices that are closer to the audio event is able to do better audio acquisition and hence show better performance. For instance, during an event like coughing a smart watch seems to

be the closest device and hence provides the best classification output. This factor can also be understood by looking at the case of IOT devices. The figure 7.3 shows that IOT devices tend to have the poorest performance. This is mainly because these IOT devices tend to be the farthest from most of the audio events and hence tend to have a poor audio acquisiton which eventually leads to poor model performance.

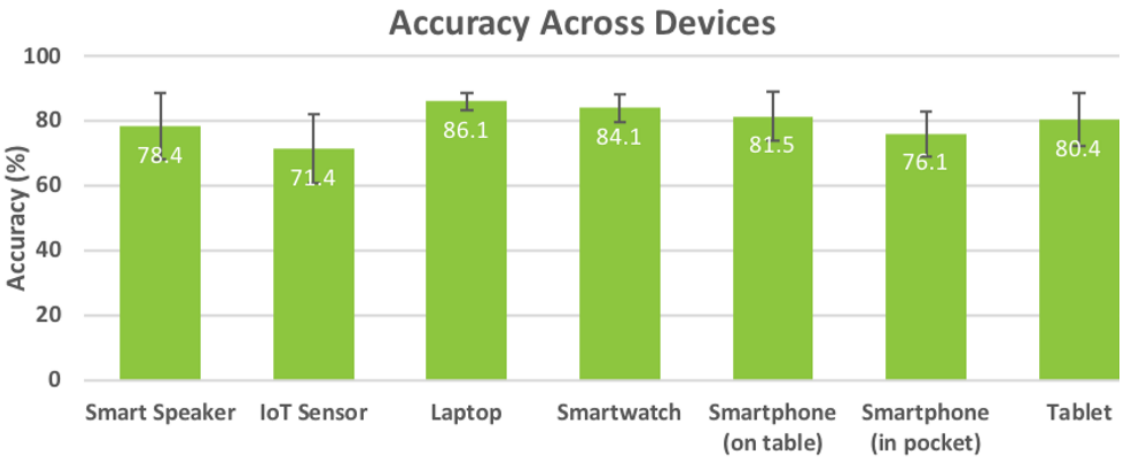

Figure 7.3: Accuracy across different devices

CHAPTER 8

CONCLUSION

The project shows that microphone present across a wide variety of devices can be used for audio classification quite effectively without having to makes any changes to the hardware. The project also shows the importance of reliable sound data and the effectiveness of the pre-processing where in data augmentation is applied. We also look into the effectiveness of our approach and its corresponding model across different devices. We see that our approach is not perfect however it achieve comparable performances with respect to related works. However, in specific contexts of audio based event classification, our model out-performs other approaches that currently exist.

8.1 FUTURE ENHANCEMENTs

Our evaluations show promising results that could make sound-based activity recognition more practical. That said, our work also has limitations, which we now discuss along with directions for future work.

The evaluation and testing of the model shows that the approach shown in the project has potential for further development. Our work includes some limitation that can be tackled to improve the model and hence make it more practical solution for real-time audio based event classification.

8.1.1 Accuracy

The accuracy of the system is an area where improvements are definitely possible. The average accuracy of our approach is not good enough to challenge human judgments and hence the approach in our case needs to be improved for it be a robust and practical solution to be used to in a day to day application. This might be achieved by looking into different types of neural network architectures.

8.1.2 Simultaneous Events

The approach shown in this project is suitable for individual occurrences of events. However, in many situations and conditions in the real world, it can turn out to be quite difficult to get a single sound event. The occurrence of multiple sound events is somethings where this approach suffers badly and hence its performance is affected badly. Noise in the environment combined with simultaneous occurrence of events is leads to poor accuracy and hence is something that can be improved for developing this approach even better. Such development can also prove to be of great importance for other works related to audio based event understanding and classification.

8.1.3 Privacy

One important part of this project is audio acquisition using microphones. However, doing real time audio acquisition can be considered as a breach of privacy. By keeping the microphones in always listen mode, we tend to intrude into the privacy of people and their personal space. This might be resolved by ensuring that the data acquired is not stored or send anywhere. Another solution to this problem could be that the data acquired can be encrypted before being used for any purpose. However, with the demand for IOT systems increasing, this is a problem not just with present with a microphone equipped device, rather this is problem faced by all data acquisition based IOT devices. An effective solution to this problem could prove to be helpful to a lot IOT based devices as well.

APPENDIX A

PROJECT CODE

A.1 Model training and testing

```
#!/usr/bin/env python
# coding: utf-8
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')


df = pd.read_csv('input/train_post_competition.csv')
df.head()

from IPython.display import Audio
file = '786ee883.wav'
path = 'input/audio_train/'
Audio(filename=path+file)

import wave

def get_length(file):
    audio = wave.open(path+file)
    return audio.getnframes() / audio.getframerate()
```

```
get_length(file)


from joblib import Parallel, delayed

with Parallel(n_jobs=10, prefer='threads', verbose=1) as ex:
    lengths = ex(delayed(get_length)(e) for e in df.fname)


df['length'] = lengths
df.head()


df = df.query('length <= 6').reset_index(drop=True)
print(df.shape)
df.head()

import librosa

y, sr = librosa.load(path+file)
# y : audio data
# sr: sample rate

plt.plot(y)
plt.title(f'Sample rate = {sr}', size=18);


mfcc = librosa.feature.mfcc(y, sr, n_mfcc=40)
print(mfcc.shape)

plt.figure(figsize=(10,5))
plt.imshow(mfcc, cmap='hot');
```

```
def obtain_mfcc(file, features=40):

    y, sr = librosa.load(path+file, res_type='kaiser_fast')
    return librosa.feature.mfcc(y, sr, n_mfcc=features)


obtain_mfcc(file).shape

mfcc.shape



def get_mfcc(file, n_mfcc=40, padding=None):

    y, sr = librosa.load(path+file, res_type='kaiser_fast')
    mfcc = librosa.feature.mfcc(y, sr, n_mfcc=n_mfcc)
    if padding: mfcc = np.pad(mfcc, ((0, 0), (0, max(0,
    padding-mfcc.shape[1]))), 'constant')
    return mfcc.astype(np.float32)



mfcc = get_mfcc(file, padding=200)
print(mfcc.shape)
plt.figure(figsize=(12,5))
plt.imshow(mfcc, cmap='hot');

print(get_mfcc(df.sort_values('length').fname.iloc[-1]).shape)
```

```
from functools import partial

n_mfcc = 40
padding = 259
fun = partial(get_mfcc, n_mfcc=n_mfcc, padding=padding)

with Parallel(n_jobs=10, prefer='threads', verbose=1) as ex:
    mfcc_data = ex(delayed(partial(fun))(e) for e in df.fname)


mfcc_data = np.stack(mfcc_data)[..., None]
mfcc_data.shape



lbl2idx = {lbl:idx for idx,lbl in enumerate(df.label.unique())}
idx2lbl = {idx:lbl for lbl,idx in lbl2idx.items()}
n_categories = len(lbl2idx)


n_categories = len(lbl2idx)

df['y'] = df.label.map(lbl2idx)
df.head()


from sklearn.model_selection import train_test_split

x_train, x_val, y_train, y_val = train_test_split(mfcc_data, df.y,
test_size=0.2, random_state=42)
x_train.shape, x_val.shape
```

```
from keras.models import Model
from keras.layers import Dense, Conv2D,
BatchNormalization, Dropout, Input, GlobalAvgPool2D,
GlobalMaxPool2D, concatenate
from keras.optimizers import Adam, SGD
import keras.backend as K



bs = 128
lr = 0.003

m_in = Input([n_mfcc, padding, 1])
x = BatchNormalization()(m_in)

layers = [10, 20, 50, 100]
for i,l in enumerate(layers):
    strides = 1 if i == 0 else (2,2)
    x = Conv2D(l, 3, strides=strides, activation='relu',
    padding='same',
                    use_bias=False,
                    kernel_initializer='he_uniform')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.02)(x)

x_avg = GlobalAvgPool2D()(x)
x_max = GlobalMaxPool2D()(x)

x = concatenate([x_avg, x_max])
x = Dense(1000, activation='relu', use_bias=False,
kernel_initializer='he_uniform')(x)
```

```
x = Dropout(0.2)(x)
m_out = Dense(n_categories, activation='softmax')(x)

model = Model(m_in, m_out)
model.compile(Adam(lr),
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
model.summary()

log1 = model.fit(x_train, y_train, bs, 15,
validation_data=[x_val, y_val])



K.eval(model.optimizer.lr.assign(lr/10))
log2 = model.fit(x_train, y_train, bs, 10,
validation_data=[x_val, y_val])



def show_results(*logs):
    trn_loss, val_loss, trn_acc, val_acc = [], [], [], []

    for log in logs:
        trn_loss += log.history['loss']
        val_loss += log.history['val_loss']
        trn_acc += log.history['acc']
        val_acc += log.history['val_acc']

    fig, axes = plt.subplots(1, 2, figsize=(14,4))
    ax1, ax2 = axes
    ax1.plot(trn_loss, label='train')
    ax1.plot(val_loss, label='validation')
```

```
    ax1.set_xlabel('epoch'); ax1.set_ylabel('loss')
    ax2.plot(trn_acc, label='train')
    ax2.plot(val_acc, label='validation')
    ax2.set_xlabel('epoch'); ax2.set_ylabel('accuracy')
    for ax,title in zip(axes, ['Train', 'Accuracy']):
        ax.set_title(title, size=14)
        ax.legend()



show_results(log1, log2)



sample = df.sample()
sample_file = sample.fname.iloc[0]
sample_label = sample.label.iloc[0]

mfcc = get_mfcc(sample_file, n_mfcc, padding)[None, ..., None]
y_ = model.predict(mfcc)
pred = idx2lbl[np.argmax(y_)]



print(f'True␣␣␣␣␣␣␣=␣{sample_label}')
print(f'Prediction␣=␣{pred}')
Audio(path + sample_file)



def get_mfcc2(file, n_mfcc=40, padding=None):
    y, sr = librosa.load(file, res_type='kaiser_fast')
    mfcc = librosa.feature.mfcc(y, sr, n_mfcc=n_mfcc)
    if padding: mfcc = np.pad(mfcc, ((0, 0), (0, max(0,
    padding-mfcc.shape[1]))), 'constant')
```

```
    return mfcc.astype(np.float32)

mfcc = get_mfcc2("test_audio.wav", n_mfcc, padding)[None,
..., None]
y_ = model.predict(mfcc)
pred = idx2lbl[np.argmax(y_)]
print(pred)

model.save('best_model.h5')


from keras.models import load_model
model = load_model('best_model.h5')


import librosa
n_mfcc = 40
padding = 259
mfcc = get_mfcc("047b3d34.wav", n_mfcc, padding)[None,
..., None]
y_ = model.predict(mfcc)
pred = idx2lbl[np.argmax(y_)]
print(pred)
```

A.2 Audio classification web classification

```
from flask import Flask,render_template,
flash, url_for, request,redirect,Blueprint
from flask import Response, make_response
import requests
import json
import sys, os
import math
import numpy as np
import pandas as pd
import wave
import librosa

from keras.models import Model
from keras.layers import Dense, Conv2D, BatchNormalization, Dropout, Inpu
from keras.optimizers import Adam, SGD
import keras.backend as K

from keras.models import load_model

import pymongo
from pymongo import MongoClient
mongoClient = MongoClient('localhost',27017)
db=mongoClient['coughTracker']
user_collection=db.users

bs = 128
lr = 0.003

df = pd.read_csv('input/
train_post_competition.csv')
```

```
def obtain_mfcc(file, features=40):
    y, sr = librosa.load(path+file, res_type='kaiser_fast')
    return librosa.feature.mfcc(y, sr, n_mfcc=features)

def get_mfcc(file, n_mfcc=40, padding=None):
    y, sr = librosa.load(file, res_type='kaiser_fast')
    mfcc = librosa.feature.mfcc(y, sr, n_mfcc=n_mfcc)
    if padding: mfcc = np.pad(mfcc, ((0, 0), (0, max(0, padding-mfcc.shap
    return mfcc.astype(np.float32)

app = Flask(__name__)
app.secret_key='asdasd^%$%^&asdjh%^$f^'
@app.route('/login')
def login():
    return render_template("login.html")

@app.route('/logout')
def logout():
    resp = make_response(redirect(url_for('login')))
    resp.set_cookie('clientId','', expires=0)
    return resp

@app.route('/loginVerify',methods=['GET', 'POST'])
def loginVerify():
    clientId = request.form['clientId']
    resp = make_response(redirect(url_for('index')))
    resp.set_cookie('clientId', clientId ,max_age=60*60*12)
    return resp

@app.route('/index')
def index():
```

```
    if request.cookies.get('clientId') is not None:
        clientId = request.cookies.get('clientId')
        print("Inside index")
        print(clientId)
        if user_collection.find_one({"clientId":clientId}) is not None:
            coughCount = user_collection.find_one({"clientId":clientId})[
            print("inside if of index")
            print(coughCount)
            if coughCount > 3 :
                return render_template('index.html', clientId=clientId , c
            else:
                return render_template('index.html', clientId=clientId , c
        else:
            data = {"clientId":clientId ,"coughCount":0}
            user_collection.insert_one(data)
            print("New client created")
            return render_template('index.html', clientId=clientId , cough
    else:
        return redirect(url_for('login'))

@app.route('/saveSound',methods=['GET', 'POST'])
def saveSound():
    data = request.data
    print("hello")
    #print(data)
    with open("test_audio.wav","wb") as fo:
        fo.write(data)
    print(request)
    return Response("{'a':'b'}", status=201, mimetype='application/json')

@app.route('/audioClassify',methods=['GET', 'POST'])
def audioClassify():
```

```
    model = load_model('best_model.h5')
    model._make_predict_function()
    n_mfcc = 40
    padding = 259
    mfcc = get_mfcc("test_audio.wav", n_mfcc, padding)[None, ..., None]
    y_ = model.predict(mfcc)
    pred = idx2lbl[np.argmax(y_)]
    print(pred)
    clientId = request.cookies.get('clientId')
    print("Inside audioClassify")
    print(clientId)
    if pred == "Cough":
        coughCount = user_collection.find_one({"clientId":clientId})["cou
        print("current coughCount")
        print(coughCount)
        print("coughCount+1")
        coughCount=coughCount+1
        print("New coughCount:")
        print(coughCount)
        user_collection.update_one({"clientId":clientId},{"$set": { "coug

    flash("Sound is :"+pred)
    K.clear_session()
    os.system("rm -rvf test_audio.wav")
    #return render_template('index.html')
    return redirect(url_for('index'))


if __name__ == '__main__':
    context = ('ssl.cert', 'ssl.key')
    app.run(host='0.0.0.0',port=8124,ssl_context=context)
```

APPENDIX B

SCREENSHOTS

REFERENCES

[1] (2018). "Shotspotter." *www.shotspotter.com, Retrieved on April 3, 2018.*

[2] Aytar, Y., Vondrick, C., and Torralba, A. (2016). "Soundnet: Learning sound representations from unlabeled video." *Advances in Neural Information Processing Systems (.*

[3] Bello, B. M. . E. J. H. . J. P. (2015). "A software framework for musical data augmentation." *In Proceeding International Society for Music Information Retrieval Conference (ISMIR 2015), pp. 248-254.*

[4] Choudhury, T., Consolvo, S., Harrison, B., Hightower, J., LaMarca, A., LeGrand, L., and et al, A. R. (2008). "The mobile sensing platform: An embedded activity recognition system." *IEEE Pervasive Computing, vol. 7, no. 2, pp. 32-41, April-June 2008. DOI: 10.1109/MPRV.2008.39.*

[5] Dennis, J., Tran, H. D., and Chng, E. S. (2013). "Image feature representation of the sub bandpower distribution for robust sound event classiïňĄcation." *Audio, Speech, and Language Processing, IEEE Transactions on, vol. 21, no. 2, pp. 367âĂŞ377, 2013.*

[6] Elizalde, B., Badlani, R., Shah, A., Kumar, A., and Raj, B. (2017). "Never-ending learner of sounds." *NIPS Workshop on Machine Learning for Audio.*

[7] Eronen, A. J., Peltonen, V. T., Tuomi, J. T., Klapuri, A. P., Fagerlund, S., Sorsa, T., Lorho, G., and Huopaniemi, J. (2006). "Audio-based context recognition." *IEEE Transactions on Audio, Speech, and Language Processing 14, no. 1 (2006): 321-329.*

[8] Foggia, P., Petkov, N., Saggese, A., Strisciuglio, N., and Vento, M. (2015). "Reliable detection of audio events in highly noisy environments." *Pattern Recognition Letters 65 (2015): 22-28.*

[9] Hong Lu, W. P. . N. D. L. . T. C. . A. T. C. (2009). "Soundsense: scalable sound sensing for people-centric applications on mobile phones." *Proceedings of the 7th international conference on Mobile systems, applications, and services (MobiSys '09). ACM, New York, NY, USA, 165-178.*

[10] Lane, N. D., Georgiev, P., and Qendro, L. (2015). "Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning." *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15). ACM, New York, NY, USA, 283-294. DOI: https://doi.org/10.1145/2750858.2804262.*

[11] Lu, H., Pan, W., Lane, N. D., Choudhury, T., and Campbell, A. T. (2009). "Soundsense: scalable sound sensing for people-centric applications on mobile phones." *Proceedings of the 7th international conference on Mobile systems, applications, and services (MobiSys '09). ACM, New York, NY, USA, 165-178.*

[12] Moore, B. C. M. (1995). "Hearing." *Academic Press, Toronto.*

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

[14] Parascandolo, G., Heittola, T., Huttunen, H., and Virtanen, T. (2017). "Convolutional recurrent neural networks for polyphonic sound event detection." *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 25, 6: 1291-1303. DOI: https://doi.org/10.1109/TASLP.2017.2690575*.

[15] Phan, H., Hertel, L., Maass, M., and Mertins, A. (2016). "Robust audio event recognition with 1-max pooling convolutional neural networks.." *https://arxiv.org/abs/1604.06338*.

[16] Plakal, S. H. . S. C. . D. P. E. . J. F. G. . A. J. . R. C. M. . M. (2017). "Cnn architectures for largescale audio classification." *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 131-135. IEEE.*

[17] Salamon, J. and Bello, J. P. (2017). "Deep convolutional neural networks and data augmentation for environmental sound classification." *IEEE Signal Processing Letters. 24.3 (2017): 279-283*.

[18] Stork, J. A., Spinello, L., Silva, J., and Arras, K. O. (2012). "Audio-based human activity recognition using non-markovian ensemble voting." *ROMAN, 2012 IEEE, pp. 509-514. IEEE.*

[19] Truong, K. Y. . K. N. (2012). "Bodyscope: a wearable acoustic sensor for activity recognition." *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12). ACM, New York, NY, USA, 341-350. DOI=http://dx.doi.org/10.1145/2370216.2370269*.

[20] Walters, T. C. (2011). "Auditory-based processing of communication sounds." *Ph.D. thesis, University of Cambridge*.

[21] Ward, J. A., Lukowicz, P., Troster, G., and Starner, T. E. (2006). "Activity recognition of assembly tasks using body-worn microphones and accelerometers." *IEEE transactions on pattern analysis and machine intelligence, 28, no. 10: 1553-1567.*