

ROBUST AUDIO AND WIFI SENSING VIA DOMAIN ADAPTATION AND KNOWLEDGE SHARING FROM EXTERNAL DOMAINS

Md Tamzeed Islam

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science in the College of Arts and Sciences.

Chapel Hill
2021

Approved by:
Shahriar Nirjon
Mary Baker
Mohit Bansal
Henry Fuchs
Jasleen Kaur

© 2021
Md Tamzeed Islam
ALL RIGHTS RESERVED

ABSTRACT

Md Tamzeed Islam: Robust Audio and WiFi Sensing via Domain
Adaptation and Knowledge Sharing From External Domains
(Under the direction of Shahriar Nirjon)

Recent advancements in machine learning have initiated a revolution in embedded sensing and inference systems. Acoustic and WiFi-based sensing and inference systems have enabled a wide variety of applications ranging from home activity detection to health vitals monitoring. While many existing solutions paved the way for acoustic event recognition and WiFi-based activity detection, the diverse characteristics in sensors, systems, and environments used for data capture cause a shift in the distribution of data and thus results in sub-optimal classification performance when the sensor and environment discrepancy occurs between training and inference stage. Moreover, large-scale acoustic and WiFi data collection is non-trivial and cumbersome. Therefore, current acoustic and WiFi-based sensing systems suffer when there is a lack of labeled samples as they only rely on the provided training data.

In this thesis, we aim to address the performance loss of machine learning-based classifiers for acoustic and WiFi-based sensing systems due to sensor and environment heterogeneity and lack of labeled examples. We show that discovering latent domains (sensor type, environment, etc.) and removing domain bias from machine learning classifiers make acoustic and WiFi-based sensing robust and generalized. We also propose a few-shot domain adaptation method that requires only one labeled sample for a new domain that relieves the users and developers from the painstaking task of data collection at each new domain. Furthermore, to address the lack of labeled examples, we propose to exploit the information or learned knowledge from sources where available data already exists in volumes, such as textual descriptions and visual domain. We implemented our algorithms in mobile and embedded platforms and collected data from participants to evaluate our proposed algorithms and frameworks in an extensive manner.

To my parents and my wife

ACKNOWLEDGEMENTS

I would like to express my utmost gratitude to my supervisor Professor Shahriar Nirjon for his constant guidance throughout the whole Ph.D. His support and direction helped me in developing myself as an independent researcher, which has been a top priority from the beginning of the Ph.D. program.

I am extremely grateful to all of my committee members Prof. Henry Fuchs, Prof. Jasleen Kaur, Prof. Mohit Bansal and Dr. Mary Baker. I would also like to thank Professor Gary Bishop for his counsel. Their feedback and suggestions have been very beneficial for my dissertation. I highly appreciate their help during the course of the major steps in my Ph.D.

I am also thankful to Ivan Tashev from Microsoft Research, Wontak Kim from Amazon Lab126 and Sirajum Munir from Bosch Research for providing me experience and advice on research that I used in my graduate life.

Moreover, I would like to thank my parents (Jannatul Ferdaus and Md Rafiqul Islam) for all their sacrifices and hard work to help me in pursuing my dream. I am also grateful to my sister (Halima Sadia), parents-in-law (Lailun Nahar and Sheikh Baharul Islam), brother-in-law (Muhammad Delower Hossain Khan), sister-in-law (Naznin Nahar Bipasha), nephews (Saadid Arham Khan and Saahir Afran Khan) and niece (Tunaz Tareq Islam). I am also thankful to all of my friends (especially Jacky, Refat, Badhan, Alameen, Arannya) living in different parts of the world.

Finally, I want to show my appreciation for my wife (Bashima Islam) for her unconditional love, support and help in my whole Ph.D. Her insight and motivation has pushed me towards right directions whenever I lost my way.

TABLE OF CONTENTS

LIST OF FIGURES	xiii
LIST OF TABLES	xvii
CHAPTER 1: INTRODUCTION	1
1.1 Overview	1
1.2 The Proposed Robust Classification Framework	3
1.3 Usage Scenario	4
1.4 Thesis Statement	5
1.5 Contribution List	5
1.6 Thesis Organization	5
CHAPTER 2: BACKGROUND	8
2.1 Channel State Information (CSI)	8
2.2 Word Embedding	9
2.3 Attribute Embedding	10
2.4 Source Separation	11
2.5 Fast ICA	12
2.6 Measure of Residual Signals	12
2.7 Multi-Head Attention	13
CHAPTER 3: DOMAIN ADAPTATION FOR ACOUSTIC EVENT RECOGNITION	14
3.1 Introduction	14
3.2 Domain Shift in Acoustics	17
3.2.1 Causes of Acoustic Domain Shifts	17
3.2.2 Domain Effect on Acoustic Feature	18

3.2.3	Why Domain Inference is Hard?	18
3.3	Overview of Sound-Adapter	20
3.3.1	Sound-Adapter	20
3.3.2	Specialty of Sound-Adapter	21
3.4	Unsupervised Domain Discovery	21
3.4.1	Event-Agnostic Audio Representation	22
3.4.2	Domain-Wise Clustering	23
3.4.3	Putting it All Together	26
3.5	Domain Adaptive Classification	27
3.5.1	Domain Adaptation Layer	27
3.5.2	Domain Adaptation Process	28
3.5.3	Mathematical Formulation	28
3.5.4	Neural Network Architecture	29
3.6	Evaluation	29
3.6.1	Datasets	29
3.6.2	Performance of Domain Counting	30
3.6.3	Performance of Domain Discovery	31
3.6.4	Performance of Domain Adaptation	35
3.6.5	Beyond Acoustics	38
3.7	Discussion	38
3.8	Related Work	40
3.9	Summary	40
CHAPTER 4: DOMAIN ADAPTATION FOR WIFI-BASED ACTIVITY RECOGNITION		42
4.1	Introduction	42
4.2	Preliminary Study	44
4.2.1	Effect on CSI	44
4.2.2	Effect on Activity Classifier	44
4.3	WiDeo System Design	45
4.4	Rf-Visual Joint Representation	46

4.4.1	Video Feature Extraction	47
4.4.2	RF Feature Extraction	48
4.4.3	Joint Space Feature Mapping	48
4.4.4	Loss Function	49
4.5	Activity Classifier	50
4.6	Domain Adaptation	50
4.6.1	Update RF Representation	51
4.6.2	Update Activity Classifier	51
4.7	Experimental Setup	52
4.7.1	WiFi CSI Collection	52
4.7.2	Noise Reduction	53
4.7.3	Gesture Segmentation	54
4.8	Experimental Results	54
4.8.1	Performance of RF-visual joint feature	54
4.8.2	Performance of environment adaptation	55
4.8.3	Performance of person-environment adaptation	56
4.9	Related Work	56
4.10	Summary	58

CHAPTER 5: ACOUSTIC EVENT RECOGNITION THROUGH EXTERNAL KNOWLEDGE TRANSFER 59

5.1	Introduction	59
5.2	Concept Applications	61
5.2.1	Indoor Usage Scenarios	61
5.2.2	Outdoor Usage Scenarios	62
5.3	Overview of Sound-Semantics	63
5.3.1	Robust Audio Representation	63
5.3.2	Cross Modal Projection	64
5.3.3	Two-Level Classifier	65
5.4	Robust Audio Representation	66
5.4.1	Basic Approach	66

5.4.2	Analytical Formulation	66
5.4.3	Neural Network Architecture	67
5.5	Cross Modal Projection	68
5.5.1	Neural Network Architecture	68
5.5.2	Loss Function	69
5.6	Two-Level Classifier	71
5.6.1	Heard vs. Unheard Determination	71
5.6.2	Classification	72
5.7	Implementation Notes	72
5.8	Empirical Evaluation	72
5.8.1	Datasets	73
5.8.2	Performance of Robust Audio Representation	73
5.8.3	Performance of Proposed Neighbour-Aware Loss	74
5.8.4	Performance on <i>Unheard</i> Class	75
5.8.5	Performance on <i>Heard</i> Class	76
5.8.6	Performance with <i>Distractor</i> Words	77
5.9	Real world evaluation	78
5.9.1	Kitchen Monitoring	78
5.9.2	Street Monitoring	80
5.10	Related Work	81
5.11	Discussion	82
5.12	Summary	83

CHAPTER 6: WIFI-BASED ACTIVITY RECOGNITION THROUGH EXTERNAL KNOWLEDGE TRANSFER 85

6.1	Introduction	85
6.2	Example Usage	88
6.3	WiFringe System Design	89
6.4	State-Aware Representation	90
6.4.1	The Need for State-Aware Representation	90
6.4.2	Rationale Behind Deep Neural Networks	92

6.4.3	Detailed Algorithmic Steps	93
6.5	Cross-modal Projections	95
6.5.1	The Need for Cross-Modal Projections	95
6.5.2	Rationale Behind Multiple Projections	96
6.5.3	Detailed Algorithmic Steps	97
6.6	Two Stage Classifier	99
6.6.1	The Need for Two-Stage Classifier	99
6.6.2	Detailed Algorithmic Steps	100
6.7	Experimental Setup	101
6.7.1	WiFi CSI Collection	101
6.7.2	Noise Reduction	102
6.7.3	Gesture Segmentation	102
6.7.4	Data Augmentation	102
6.7.5	Empirical Dataset	103
6.8	Experimental Results	104
6.8.1	Accuracy of Unseen Class Detection	104
6.8.2	Accuracy of Seen Class Detection	105
6.8.3	Accuracy of Seen vs Unseen Class Detection	106
6.8.4	End to End Evaluation	107
6.8.5	Execution Time	108
6.9	Related Work	108
6.9.1	Device-free Sensing	108
6.9.2	Zero Shot Learning	110
6.10	Discussion	111
6.11	Summary	112
CHAPTER 7: PRIVACY PRESERVING ACOUSTIC SENSING		113
7.1	Introduction	113
7.2	Usage Scenarios	115
7.2.1	Voice-Only Mode	116

7.2.2	Personalized Commands	116
7.3	Overview of SOUNDSIFTER	116
7.3.1	Basic Workflow	117
7.3.2	Initial Setup and Programming	117
7.3.3	Usage Scenarios	118
7.4	Studying the Problem	118
7.4.1	Need for Source Separation	119
7.4.2	Number of Microphones	119
7.4.3	Benefit of Rate Adaptation	120
7.4.4	Modeling Sounds	121
7.4.5	Need for Residue Removal	121
7.4.6	Analog Data Acquisition	122
7.5	Algorithm Design	122
7.5.1	Frequency Adaptation	122
7.5.2	Signal Prediction and Filling	124
7.5.3	Modeling and Recognizing Sounds	125
7.5.4	Eliminating Secondary Residues	127
7.6	Implementation Notes	128
7.6.1	Amazon Alexa Voice Service	128
7.6.2	SoundSifter in a BeagleBone Black	128
7.6.3	Libraries	129
7.7	Evaluation	129
7.7.1	Microbenchmarks	129
7.7.2	Algorithm Evaluation	130
7.7.3	Full System Evaluation	134
7.8	Discussion	136
7.9	Related Work	136
7.10	Summary	138

CHAPTER 8: CONCLUSION AND FUTURE WORKS	139
8.1 Summary of the Dissertation	139
8.2 Future Directions	140
APPENDIX A: WIFI CSI COLLECTION	141
A.1 CSI Data Capture	141
A.2 CSI Data Processing	141
A.3 Word Embedding	141
APPENDIX B: AUDIO PROCESSING	142
B.1 Synchronous Audio Recording	142
B.2 Audio Data Processing	142
B.3 Source Separation	142
B.4 Gradient Reversal	142
B.5 High Sampling Rate	142
REFERENCES	143

LIST OF FIGURES

1.1	Overview of the proposed dissertation	4
1.2	Use-case scenario of the proposed dissertation	4
2.1	Multipaths caused by human movement	8
2.2	Illustration of Word2Vec	10
2.3	Generic model for source separation	11
2.4	Self-attention block	13
3.1	Overview of Sound-Adapter	15
3.2	Data collection with five types of microphones.	18
3.3	The same sound clip recorded by five different devices have significant variations in their spectrograms.	19
3.4	k-Means produces event-wise clusters whereas Sound-Adapter produces domain-wise clusters.	19
3.5	Sound-Adapter’s training pipeline	20
3.6	The Domain Discovery Network.	22
3.7	Illustration of cluster update step.	25
3.8	Retrofitting an existing audio classifier by inserting domain adaptation layers.	27
3.9	Distribution of neural network activations (a) before and (b) after domain adaptation.	28
3.10	Performance of domain counting	31
3.11	Sound-Adapter has superior performance in domain discovery than the baselines for our empirical dataset.	31
3.12	Sound-Adapter’s domain discovery algorithm’s performance for different scenarios	32
3.13	Sound-Adapter’s domain discovery algorithm is extendable to domains caused by environment effect.	35
3.14	Sound-Adapter’s domain adaptation’s performance	36
3.15	Sound-Adapter’s domain discovery algorithm is extendable to image modality.	38
4.1	Spectrogram of same activity in 2 different room	44
4.2	WiFi-based classifier’s performance drops when environment changes between training and testing phase	45
4.3	(a) Training pipeline for source domain (b) Pipeline for domain adaptation (c) Pipeline for inference	45

4.4	Joint Representation Learning using video and WiFi signal	46
4.5	WiFi-based activity classifier	50
4.6	Domain adaptation for WiFi-based representation extraction	51
4.7	(a) Before denoising CSI stream has dominant noise. (b) After denoising all high frequency noise components are removed.	52
4.8	(a) Intel Nuc with antennas. Volunteers performing different activities at (b) Lab 1 (c) Lab 2 (d) Lab 3 (e) Office (f) Lab 4.	53
4.9	Performance of RF-visual joint feature	54
4.10	Performance of environment adaptation	55
4.11	Performance of domain adaptation's performance for mismatched person and environment	56
5.1	Kitchen activity monitoring application.	62
5.2	Sound-Semantics acoustic processing pipeline.	63
5.3	Preserving semantics in audio representation.	64
5.4	Cross-modal Projection maps audio to its corresponding label in word embedding space.	64
5.5	The architecture of the Siamese network used in robust audio representation	67
5.6	(a) MSE Loss, (b) Neighbour-aware Loss	69
5.7	The accuracy of <i>heard</i> and <i>unheard</i> class detection depends on the threshold Π 's value.	71
5.8	Performance of neighbour-aware loss	75
5.9	Sound-Semantics' accuracy in unheard audio event classification	76
5.10	Sound-Semantics' accuracy in heard audio event classification	76
5.11	Sound-Semantics' performance with random word injection	77
5.12	Sound-Semantics' performance across scenarios for indoor environment	78
5.13	Sound-Semantics' performance across scenarios for outdoor environment	80
5.14	Discussion on Sound-Semantics' performance	83
6.1	WiFringe vs state of the art	85
6.2	WiFringe is able to recognize activities for which is has not been explicitly trained for.	88
6.3	WiFringe signal processing pipeline.	89
6.4	Different activities have different state sequences.	91
6.5	Looking back deeper in time improves modeling accuracy.	91

6.6	CNNs learn local patterns that characterize each state, whereas RNNs (LSTMs) learn sequence of states.	92
6.7	Network architecture for state-aware representation.	93
6.8	Cross-modal projections map a CSI stream to its corresponding label in the word and attribute embedding spaces.	96
6.9	State Aware Representation is projected into both Word-embedding and Attribute space which are then aggregated for a joint projection.	97
6.10	(a) State-aware activity representation, (b) Spectrogram level feature	100
6.11	(a) Intel Nuc with Antennas. (b) Experimental Setup.	101
6.12	WiFringe’s accuracy for unseen activity classification.	104
6.13	WiFringe’s accuracy is higher than baseline algorithms in seen class detection. . . .	105
6.14	The accuracy of <i>seen</i> and <i>unseen</i> class detection depends on the threshold Ω ’s value.	106
6.15	WiFringe’s performance across scenarios	107
6.16	WiFringe’s performance in unseen activity recognition is dependent on its relationship with classes in seen category.	111
7.1	(a) System interface, and (b) block diagram of SoundSifter.	117
7.2	Frequency plots of variety of sounds.	119
7.3	Source separation error for different number of microphones.	120
7.4	CPU usage increases with sampling rate.	120
7.5	Effect of residue on primary source.	121
7.6	Components and interconnections inside SoundSifter’s audio processing pipeline. . .	122
7.7	2-Step Look-back performs better than 1-Step Look-back for loud music	123
7.8	Predicting missing values of a signal (a) using linear (b) and spline (c) interpolation.	125
7.9	Leveraging isolated secondary sources to nullify their residue through a customized adaptive noise cancellation process.	127
7.10	Alexa-enabled Raspberry Pi in a case.	128
7.11	(Left) SoundSifter in its open case. (Right) the case sits on top of Alexa as a cover. .	129
7.12	Run-time Analysis.	130
7.13	SoundSifter has higher accuracy than 1 Step Look-back for all scenarios.	131
7.14	Frequency adaptation reduces CPU usage especially for lower frequency ranges. . . .	132
7.15	Linear Interpolation is more effective than Spline Interpolation for Signal Prediction and Filling.	132

7.16 SoundSifter’s data augmentation technique improves modeling accuracy 133

7.17 Confusion matrix for primary speaker recognition among up to 10 persons. 133

7.18 Residue Removal reduces noise residual from the primary signal. 134

7.19 SoundSifter’s performance for noisy voice mode 135

7.20 SoundSifter’s performance for personalized mode 135

LIST OF TABLES

2.1	Word definitions and attributes.	11
3.1	Device List	30
3.2	Environment Description	30
3.3	Dataset Statistics	30
4.1	Data Collection Environment	53
5.1	Runtime of our algorithms	72
5.2	Proposed Robust Audio Representation preserves better clustering properties	74
5.3	Dataset for Kitchen Monitoring.	78
5.4	Dataset for Street Monitoring	80
6.1	Activities used for evaluation	103
6.2	Inference time of deep networks.	108
7.1	Data augmentation techniques applied to increase user-contributed samples.	126
7.2	CPU and memory usage.	130
7.3	Description of the empirical dataset.	131
7.4	Description of the scenarios.	134

CHAPTER 1

Introduction

1.1 Overview

Having reached the milestone of human-level speech understanding by machines, continuous listening devices are now becoming ubiquitous. Today, it is possible for an embedded device to continuously capture, process, and interpret acoustic signals in real-time. Tech giants like Apple, Microsoft, Google, and Amazon have their own versions of continuous audio sensing and interpretation systems. Apple’s Siri [1] and Amazon’s Alexa [2] integrated home-hub devices— *Google Home*, *Amazon Echo*, *Apple Pod* understand what we say, and act on them to fetch us a web page, schedule a meeting, find the best sushi in town, or tell us a joke. These platforms are equipped with microphones which enable many potential application of acoustic sensing such as speech recognition [3], speaker identification [4], music recognition [5], sleep quality monitoring [6], intrusion detection [7], elderly monitoring [8], activity recognition [9], and health vitals monitoring [10, 11].

With the ubiquity of acoustic sensing systems, the diversity in their sound types as well as their intended purposes are evolving—making it hard for us to acquire a large corpus of labeled training data for all possible types of acoustic events that an application wants to detect. For example, in applications like sleep monitoring, home intrusion detection, and health monitoring, we may want to collect multiple instances of rare, inconvenient, and unwanted acoustic events such as snoring, breaking in, sneezing, and wheezing. Such requirements create challenges to developing personalized and environment-specific acoustic event recognition systems which primarily rely on user-contributed labeled data for training the classifier. Hence, the research community for long has sought after a solution that can classify audio using few or no training examples. Moreover, recent studies [12, 13] have shown that classifiers are influenced by subtle characteristics of the recording devices and their performance drops unless the training and test scenarios are recorded with the same device. A shift in the distribution of data due to device heterogeneity is known as the *domain shift* problem. It is well-established [14, 12, 15] that machine learning classifiers perform sub-optimally when there are

discrepancies in the distributions between the training and the testing data due to domain shift. An audio classifier trained on a particular training dataset is biased by the recording device (i.e., the *source domain*) that is used to collect the training samples. During inference, when this classifier is applied to audio samples recorded by a different recording device (i.e., the *target domain*) a significant accuracy loss (up to 20%) is observed [13]. Therefore, a mechanism is needed to mitigate the effect of recording devices’ diversity on machine learning algorithms to enable large scale crowd-sourced audio data collection.

The new generation of home-hub devices come with camera integrated in them, therefore opening doors to user gesture and activity monitoring. This eventually leads to potential use-case of gesture based media control, smart home appliances control, patient monitoring and so on. However, camera based monitoring systems are privacy invasive, sensitive to light conditions and fails behind occlusions. Current home-hub devices are also equipped with WiFi cards for communicating with server via internet, which gives us an unique opportunity for non-intrusive sensing using the reflected WiFi signals from the users. The accessibility of WiFi signal characteristics such as the signal strength (RSSI) and channel state information (CSI) in commodity WiFi chipsets, make WiFi an attractive technology for human activity monitoring. Alternate solutions that use wearables like smartwatches and activity trackers are becoming less attractive due to their usage adherence issue, and systems that use cameras to monitor home activities raise serious privacy concerns. WiFi sensing, on the other hand, is device-free, non-intrusive, and less privacy-invasive. Hence, in recent years, we have seen an increase in WiFi-based sensing and inference systems whose feasibility has been demonstrated in application scenarios such as home activity monitoring [16], sleep monitoring [17], gesture-based smart device interaction [18, 19], and health vitals tracking [20]. Existing machine learning algorithms perform remarkably well in these applications, given that they enjoy the availability of labeled training data that are involved in the application [21]. However, for WiFi sensing, machine learning algorithms face major hurdles regarding lack of labeled and uniform data.

Due to the non-trivial nature of WiFi data collection and labelling segments of the signal as particular gesture, there is a scarcity of datasets in the WiFi based activity monitoring community. Moreover, collecting data for all possible type of daily activities is not feasible as well. Current WiFi-based activity recognition systems employ either template matching algorithms or machine learning classifiers — ranging from traditional classifiers like support vector machines [22] to advanced

deep convolutional neural networks [23]. These algorithms require a decent number of training examples for each class of activity in order for the system to accurately classify them. Furthermore, the capability of these systems are fundamentally limited by the number of activity classes for which the system has been trained for. When these systems are presented with a completely new type of activity, there is no built-in mechanism to make an educated guess about the possible class label for that unseen example. On the other hand, WiFi signals change due to change in environment. Therefore, activity classification model trained in one environment performs sub-optimally with data from another environment. Current systems require large scale labeled data collection across all environments to perform classification which makes these algorithms hard to scale.

In this thesis, I propose to augment sensing ability of these home-hub devices in terms of robust acoustic and WiFi sensing. By exploiting information or learned knowledge from sources where available data already exists in volume, such as textual descriptions and visual domain I propose to augment audio and WiFi based sensing to deal with a limited number of labeled dataset. On the other hand, I also propose to remove the effect of sensor and environment heterogeneity from the feature representation. To achieve this, I propose the first system capable of multi-source to multi-target domain adaptation for acoustic event recognition tasks by discovering latent domains from a dataset and removing their effect on machine learning based classifiers. Furthermore, I propose a video and WiFi based joint feature extraction algorithm for WiFi sensing to adapt into a new environment with minimum effort from the developer and user’s end.

1.2 The Proposed Robust Classification Framework

I propose a generic signal processing framework (Figure 1.1) for class label inference which can be applied to any signal type. The first step of the proposed pipeline is a *Personalized Filtering* module that isolates privacy-invasive signal from a mixture of signals. This step also enforces privacy policies to restrict intrusive signals from going into untrusted systems to protect privacy of the user. The second step is *Domain Adaptation* to remove heterogeneity caused due to sensor and environment variance across data samples. This step nullifies bias due to any data collection related issue and makes the classification model robust when they mismatch between training and testing phase. The last step of the proposed unified framework is *Knowledge Transfer and Classification*. Through this step, the proposed thesis leverages knowledge from external domain such as text to embed semantic knowledge into signal features which makes the classifier robust to lack of labeled training examples.

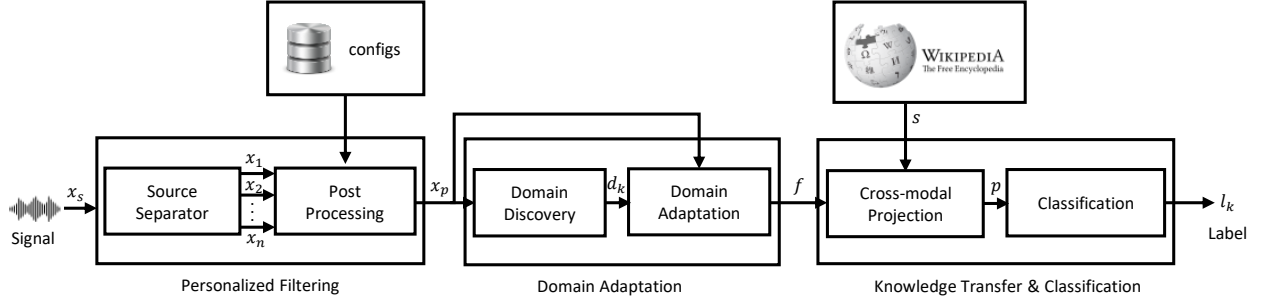


Figure 1.1: Overview of the proposed dissertation

1.3 Usage Scenario

I envision that with my contributions, the home-hub devices such as Amazon Echo will be equipped with acoustic and RF sensing ability. In Figure 1.2 the final outcome of my proposed thesis is depicted. As depicted in Figure 1.2, there are three Amazon echo and one Google home in the

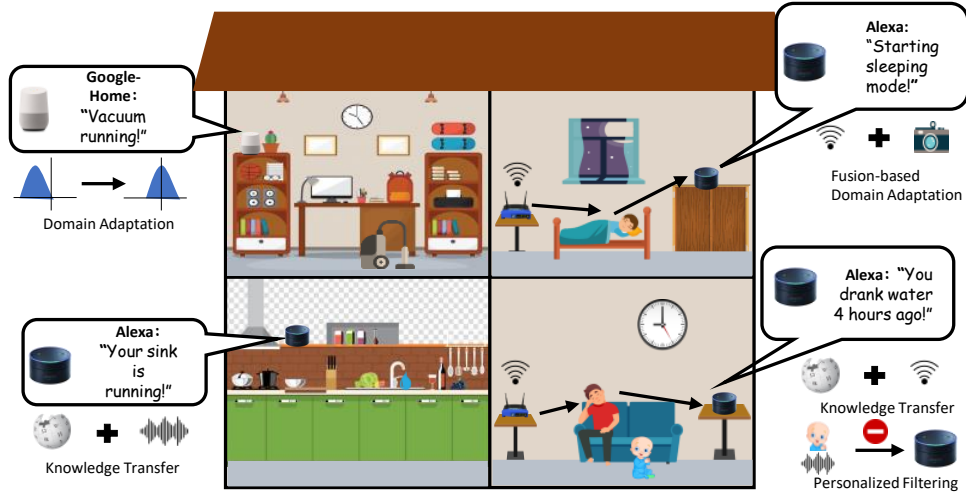


Figure 1.2: Use-case scenario of the proposed dissertation

four rooms of the house. In the left room on the bottom floor the Amazon echo detects that the *sink is running* by using our proposed acoustic classifier which embeds semantic knowledge from large scale text data such as Wikipedia. In the right room on the bottom floor the WiFi sensing activity recognition detects that the user has not drank water for a long time using cross-modal embedding of RF signal leveraging semantics from textual data. Moreover, the proposed privacy-preserving acoustic sensing stops the sound of the baby from going in the Amazon echo device. On the top floor there is a Google home in the left room that uses our proposed domain adaptation algorithm to shift the data into a reference distribution to remove sensor heterogeneity from neural network

model. On the right room of the top floor, the WiFi sensing based activity recognizer adapts its model using proposed WiFi and video based joint representation.

1.4 Thesis Statement

Semantic knowledge transfer from external sources, along with domain adaptation to nullify sensor and environment bias, yields robust and scalable signal classifiers than traditional supervised models.

1.5 Contribution List

The following are the research contributions of my thesis:

- A multi-source to multi-target domain adaptation model by discovering latent domains in unsupervised manner. Using the proposed algorithm, I remove domain-specific distribution shifts from the activation outputs of the neural network classifier in order to improve the accuracy and robustness. Moreover, a one shot domain adaptation technique is proposed to adapt domain sensitive feature extraction for new domains.
- A method for semantic knowledge transfer for general purpose signal classification where contextual information from text domain is embedded to signal feature. This cross-modal knowledge transfer enhances classifiers by overcoming the lack of adequate training examples per class.
- A privacy-preserving sensing framework by reducing contextual information leakage from continuously observing systems. The proposed system is able to isolate signal sources, identify primary source, and post process the stream to remove residuals and perform signal classification.

1.6 Thesis Organization

The remainder of the thesis is organized as follows:

- In Chapter 2 we present a background overview for the rest of the chapters. We start with WiFi based sensing and multi-modal knowledge sources. Then we describe the algorithms for source separation, acoustic residue quantification. Finally, we introduce the concept of multi-head attention mechanism for sequence modeling.

- In Chapter 3 we introduce Sound-Adapter- the first system to adapt acoustic classifier models that are trained on data from more than one acoustic conditions (i.e., multi-source domain adaptation). The proposed method does not assume availability of recording metadata (i.e., *domain labels*) in the training data—which makes the adaptation problem harder. To solve this, we propose the first multi-task deep neural network architecture to cluster audio samples according to their domain in an *unsupervised* way. Using the inferred domain information, we perform *domain adaptation* to remove biases due to domain heterogeneity from the machine learning model.
- In Chapter 4 we present WiDeo for one shot adaptation of WiFi based activity classifier for new environment. We propose a novel representation extraction algorithm using the supervision of visual data during training phase. Our proposed feature extraction explicitly learns the corresponding relation between WiFi signal and movement of human body parts. For environment adaptation we propose a framework that relies on only one labeled sample per activity class in comparison with current state-of-the-art solutions which are not suitable for few shot adaptation.
- In Chapter 5 Sound-Semantics is presented, which proposes a novel approach to acoustic event classification that exploits knowledge from the textual domain to deal with a well-known pain point in audio event classification—i.e., the lack of adequate training examples. We show that by exploiting existing context-aware semantic representation of English words (e.g., Google’s word-to-vector [24]) that is generated from a massive amount of English texts on the Internet, it is possible to classify acoustic events even when there are a few or no training examples for a wide variety of sounds.
- In Chapter 6 we focus on our proposed WiFi based activity classification system WiFringe, which is a WiFi CSI-based device-free human gesture recognition system that recognizes *named* gestures, i.e., activities and gestures that have a semantically meaningful name in English language, as opposed to arbitrary free-form gestures. Given a list of activities (only their names in English text), along with zero or more training examples (WiFi CSI values) per activity, WiFringe is able to detect all activities at runtime. We show for the first time that by utilizing the state-of-the-art semantic representation of English words, and verb attributes learned from

how a word is defined (e.g, American Heritage Dictionary), we can enhance the capability of WiFi-based named gesture recognition systems that lack adequate training examples per class.

- In Chapter 7 we study the overhearing problem of continuous acoustic sensing devices and develop a system called *SoundSifter* that mitigates personal or contextual information leakage due to the presence of unwanted sound sources in the acoustic environment. SoundSifter has hardware and software to capture the audio, isolate signals from distinct sound sources, filter out signals that are from unwanted sources, and process the signals to enforce policies such as personalization before the signals enter into an voice controlled system like Amazon Echo or Google Home.
- In Chapter 8, we conclude with the thesis. We present a summary of the proposed systems and frameworks. We also provide some possible future directions for augmenting acoustic and RF based sensing.

CHAPTER 2

Background

2.1 Channel State Information (CSI)

In wireless communication, a transmitter and a receiver talk to each other using a certain frequency or a range of frequencies, which is often called a *channel*. The *Channel State Information* (CSI), as the name implies, describes the properties of a wireless channel. For example, when wireless signals travel through the medium, they fade, they get reflected and scattered by obstacles on the way, and their power decays with the distance traveled. The CSI is a measure of all these phenomena of a wireless channel. Mathematically, using the frequency domain terms, we express the relationship between the transmitted signal $X(f, t)$, the channel frequency response (CFR) $H(f, t)$, and the received signal $Y(f, t)$ as: $Y(f, t) = H(f, t) \cdot X(f, t) + N(f, t)$, where $N(f, t)$ denotes the noise. The CSI comprises of the CFR values, i.e., $\{H(f, t)\}$.

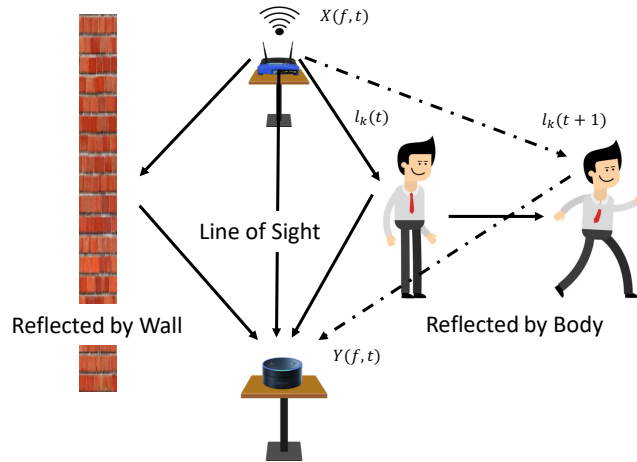


Figure 2.1: Multipaths caused by human movement

Data communication over a WiFi channel happens by sending the data bits simultaneously over

64 distinct frequencies (called the *sub-carriers*) in parallel. From the WiFi Network Interface Card (NIC) [25], it is possible to obtain the CSI values of these subcarriers. The frequency response, $H(f, t)$ of each sub-carrier is a complex number. For N_{TX} transmitting antennas, N_{RX} receiving antennas, and N_S sub-carriers, we get a CSI matrix of complex numbers having the dimensions of $N_{TX} \times N_{RX} \times N_s$.

For a signal received through N paths, the CFR can be expressed as follows:

$$H(f, t) = e^{-j2\pi\Delta f t} \sum_{k=1}^N a_k(f, t) e^{-j2\pi f \tau_k(t)}$$

where $a_k(f, t)$ denotes attenuation and initial phase offset of the k th path, $e^{-j2\pi f \tau_k(t)}$ denotes the phase shift on the k th path with a propagation delay of $\tau_k(t)$. If a person moves and the length of the k th path changes by $\Delta l_k = l_k(t+1) - l_k(t)$, then the propagation delay $\tau_k(t)$ is expressed as $\tau_k(t) = \frac{\Delta l_k}{c}$, where c is the speed of light. Due to the synchronization issue between the sender and the receiver a phase shift of $e^{-j2\pi\Delta f t}$ is observed, where Δf is the carrier frequency difference between the sender and the receiver.

2.2 Word Embedding

The process of *Word Embedding* [26] maps words in a natural language to vectors of real numbers in a manner that words that are commonly used in the same textual context are positioned closely in the vector space. Intuitively, word embedding expresses the distributional semantics of words with the motto that “*a word is characterized by the company it keeps*” [27]. Word embedding extracts distributional similarities among the words from a large scale text data, such as the Wikipedia, by observing the words that appear with similar words which define their context. For example, consider the words: love and adore. Syntactically these two words are quite different, but they often appear in similar semantic contexts, i.e., with similar words. Hence, the word embedding process would map these two words to two vectors whose distance is relatively closer than the embedding of two random words.

In Figure 2.2(a), we plot the embedding of six English words. We only show the first two principal components to be able to visualize the vector representation in 2D. We observe that words with similar semantic meaning are closer in the word embedding space, e.g., {run, walk, move}, whereas words having different semantic meaning are far, e.g., run and eat. In Natural Language Processing

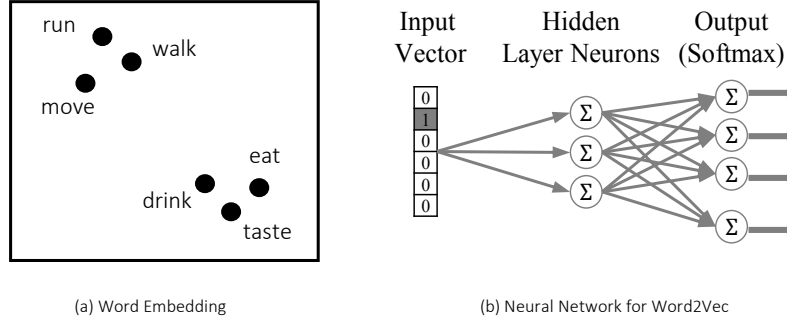


Figure 2.2: (a) Semantically similar words appear close to each other in the word embedding space (b) A two layer neural network is used for Word2Vec [24] word embedding

(NLP), researchers use techniques such as neural network [28] and co-occurrence matrix [29] to project words onto a k -dimensional vector space, and words appearing in similar textual contexts get closer embedding in the vector space. One of the most popular method to extract word embedding from a large scale corpus is *Word2Vec* [24], which uses a two layer neural network to predict the surrounding words of a target word.

Figure 2.2(b) shows a two-layer neural network, where a word, represented by its one-hot encoding [30], is the input and the last layer of the network outputs the probability of other words' being its neighbour. Word-pairs of the form $\{(word, neighbor)\}$ are used to train this network, which explicitly learns to predict neighbour words of a given word, and in the process, the network implicitly learns the context of the words. After the training phase, the output of the hidden layer is used as the embedding that projects an input word to a low-dimension vector space.

2.3 Attribute Embedding

While word embedding captures the co-occurrence information of words used in the same context, it does not quite describe the attributes of a verb. Recently, natural language processing community has proposed an effective method to learn the attributes of English *verbs* from their dictionary definitions [31]. In this new method, verbs are expressed in terms of a set of attributes. Each verb is expressed as a vector of real numbers where each element of the vector corresponds to an attribute.

Table 2.1 provides a simplified example. Three verbs: *Drink*, *Sip*, and *Drool* are expressed in terms of four attributes: *Motion*, *Social*, *Object*, *Head*, where the attributes correspond to the degree of motion, degree of social engagement, use of objects, and use of head, respectively. We intentionally left the list of attributes open to emphasize that additional attributes are necessary to encode the

Table 2.1: Word definitions and attributes.

Word	Representation
Drink	Dictionary: "To take into the mouth and swallow a liquid." Attributes: $(Motion, Social, Object, Head, \dots) = (low, solitary, true, true, \dots)$
Sip	Dictionary: "To drink in small quantities." Attributes: $(Motion, Social, Object, Head, \dots) = (low, social, true, true, \dots)$
Drool	Dictionary: "To let run from the mouth." Attributes: $(Motion, Social, Object, Head, \dots) = (none, solitary, false, true, \dots)$

dictionary definitions of a large number of English verbs.

2.4 Source Separation

The term ‘*blind source separation*’ [32] refers to the generic problem of retrieving N unobserved sources only from the knowledge of P observed mixtures of these sources. The problem was first formulated to model neural processing of human brains [33], and has later been extended and studied in many other contexts such as biomedical applications [34, 35], communication [36, 37], finance [38, 39], security [40, 41], and acoustics [42, 43, 44]. To the acoustic processing community, this problem is popularly known as the ‘*cocktail party problem*,’ where sources represent human voices.

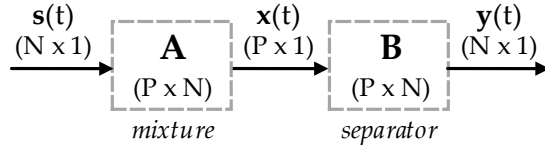


Figure 2.3: Generic model for source separation

Here, each microphone observes a weighted combination of N sound sources. Assuming $\mathbf{s}(t) = (s_1(t), \dots, s_N(t))^T \in \mathbb{R}^N$ denotes the sources, $\mathbf{x}(t) = (x_1(t), \dots, x_P(t))^T \in \mathbb{R}^P$ denotes the observed mixtures, $(.)^T$ stands for matrix transpose operation, and \mathbf{A} denotes an unknown mapping from \mathbb{R}^N to \mathbb{R}^P , we can write: $\forall t \in \mathbb{Z} \quad \mathbf{x}(t) = \mathbf{A}\mathbf{s}(t)$.

The equation above is of a *linear instantaneous* model, which is the most commonly used model for source separation. This does not explicitly model noise, as it can be implicitly modeled as an additional source. In Figure 2.3, \mathbf{A} is a *mixing* matrix that mixes N acoustic sources to produce P output streams. Retrieving the sources is equivalent to finding \mathbf{B} , the inverse or *separator* matrix. The separated outputs are expressed as: $\forall t \in \mathbb{Z} \quad \mathbf{y}(t) = \mathbf{B}\mathbf{x}(t)$. When $N \leq P$, \mathbf{A} is invertible. But

for $N > P$, additional assumptions (e.g., sparsity [45]) may be required.

2.5 Fast ICA

There are many solutions to the source separation problem that make different assumptions about sources and use different mixing systems [46, 39, 47]. *Independent Component Analysis* (ICA) is one of the most popular solutions. This approach assumes that the acoustic sources are statistically independent from each other – which is in general true for our application scenario. For example, voice commands to an Amazon Echo device and unwanted background sounds are unlikely to have any statistical correlations among themselves.

Fast ICA [48] is a popular, efficient independent component analysis-based source separation algorithm. It isolates the sources by iteratively maximizing a measure of mutual independence among the sources. In Fast ICA, *non-Gaussianity* [32] of the sources is taken as the measure.

Using matrix notation, $\mathbf{x}(t)$ is expressed as $\mathbf{X} \in \mathbb{R}^{P \times T}$. FastICA iteratively updates a weight vector $\mathbf{W} \in \mathbb{R}^P$ to maximize non-Gaussianity of the projection $\mathbf{W}^T \mathbf{X}$ using the following two steps in a loop:

$$\begin{aligned} \mathbf{W}^+ &\leftarrow \mathbf{E}\{\mathbf{X}\phi(\mathbf{W}^T \mathbf{X})\} - \mathbf{E}\{\phi'(\mathbf{W}^T \mathbf{X})\}\mathbf{W} \\ \mathbf{W} &\leftarrow \mathbf{W}^+ / \|\mathbf{W}^+\| \end{aligned} \tag{2.1}$$

Here, $\phi(x) = \tanh(x)$, $\phi'(x)$ is its derivative, and $\mathbf{E}\{\}$ is average over columns of a matrix. \mathbf{W} is initialized to a random vector, and the loop stops when there is no significant change in it. Note that, for simplicity, we only show how to isolate one source in Equation 2.1; for multiple sources, this needs to be repeated for each source. We also omit the preprocessing steps that involves prewhitening [32] matrix \mathbf{X} .

2.6 Measure of Residual Signals

Because no source separation is perfect, there are always residues of secondary sources within the isolated stream of audio that is supposed to carry signals from the primary source only. We use a metric to quantify this residual with the following equation:

$$\xi_i = \|x_i(t) - y_i(t)\|_2 \tag{2.2}$$

Here, ξ_i denotes the amount of residuals in the i^{th} source after source separation, which is expressed as the l^2 -norm of the difference between primary signals before and after source separation. We use this metric in our evaluations to quantify the quality of source separation.

2.7 Multi-Head Attention

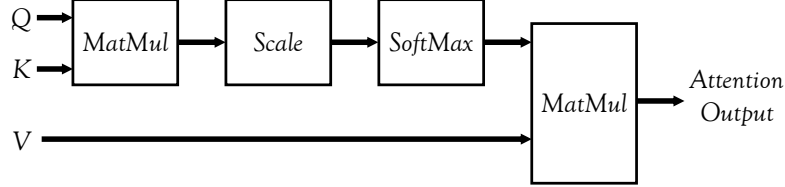


Figure 2.4: Self-attention block

Sequence modeling such as recurrent neural networks (RNN) [49] extracts the temporal information from a timeseries data, by producing a hidden state for current timestep's data, so that it incorporates the contextual representation of neighboring timestep's data. For any given instant the generated representation from RNN only depends on its immediate past or future timestep's representation to gather the contextual information. Thus, RNNs lack the capability of capturing information from input data that are distant in time. To address this, recent Transformer [50, 51] architecture has been proposed, which is based on the method of self-attention mechanism. Self-attention computes the contextual relevance of all timesteps data for a particular instant data's representation. The multi-head attention block is consisted of single self-attention function. To calculate single self-attention for input vector at each timestep, three sets of vectors namely a *Query vector* (Q), a *Key vector* (K), and a *Value vector* (V) are created by multiplying the input vectors by three trainable matrices that are trained during the training phase. The final output attention is a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key [51], which is defined as follows:

$$Attention(K, Q, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.3)$$

Here, d_k is used for scaling which is the dimension of key vector (K). For multi-headed attention, there are multiple projected versions of queries, keys and values for which the attention function is performed in parallel, yielding multiple output values that are concatenated into a final vector.

CHAPTER 3

Domain Adaptation for Acoustic Event Recognition

3.1 Introduction

Recent advancements in embedded acoustic sensing and inference systems have enabled personal voice assistants [52] like Siri, Alexa and Cortana, and computational ear-pieces aka *hearables* like smart hearing aids, personalized amplifiers, and earbuds [53, 54, 55]. These platforms have opened doors to many applications such as activity and event monitoring [56, 57], speech recognition [58, 3], music recognition [5], sleep quality monitoring [59], elderly monitoring [8], and health vitals monitoring [10, 11]. These audio classifiers are trained on large and representative datasets. Typically, a developer collects training data from as many sources as possible and trains the classifier with labeled samples originating from multiple datasets—which are likely to be recorded in different acoustic conditions, e.g., microphones and environments.

Recent studies [12, 13, ?] have shown that classifiers are influenced by subtle characteristics of the recording device and the recording environment, and their accuracy drops unless the training and test data are recorded in the same conditions. A shift in the distribution between training and testing data is known as the *domain shift* problem. It is known that machine learning classifiers perform sub-optimally when there are discrepancies in the distributions between the training and the testing data due to domain shifts [14, 12, 15]. An audio classifier trained on a particular training dataset is biased by the recording device and the acoustic environment (i.e., the *source domain*). During inference, when this classifier is applied to audio samples recorded by a different recording device or a different environment or a different combination of these (i.e., the *target domain*) up to 20% loss of accuracy is observed [13].

State-of-the-art works [13, 12] on domain shift problem in audio classification only addresses the *single source domain to single target domain* scenario. These works assume only one source domain (e.g., one recording device) in the training dataset and do not provide – nor they require – any methods to identify the source domains. These solutions are hard to scale in practical scenarios

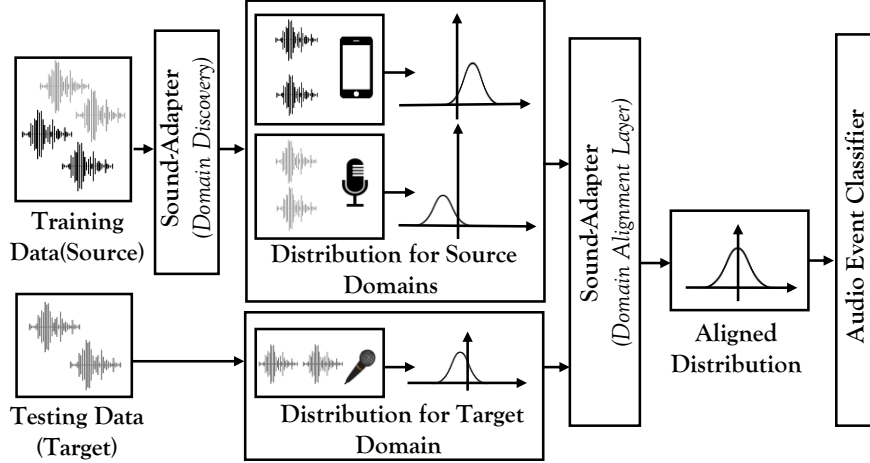


Figure 3.1: The training dataset having two domains is passed through the *Domain Discovery* module to infer the domain. Each domain has a different distribution, which is aligned by the *Domain Alignment Layer*. The audio event classifier is trained on the aligned distribution of data.

where the training data are recorded with a wide variety of devices and in many environments, and no meta-data about these recordings are available. Therefore, currently, there is no scalable solution to the multi-source to target audio domain adaptation problem.

In this chapter, we propose *Sound-Adapter* — the first system capable of multi-source domain adaptation for acoustic event recognition. Here, *domains* refer to conditions that alter the characteristics of the data. As depicted by Figure 6.1, the central idea behind *Sound-Adapter* is to discover domains from the input data in an unsupervised fashion, and then to align the input data from each domain by normalizing them to match a reference distribution. In the training stage, this domain discovery and alignment step does not require the *domain labels* (i.e., which microphone and environment the audio sample came from), but it does require the *class labels* (i.e., what sound it is). When a classifier is trained on the aligned distributions, it becomes unbiased to any particular domain. During inference, the same alignment step is applied to the test data (from the target domains) – which improves the accuracy and robustness of classification.

To the best of our knowledge, we propose the first deep neural network architecture for unsupervised domain discovery to infer the domain information from audio samples containing only their class labels and no domain labels. This is particularly challenging since the variation across different domains is subtle and often gets masked by the variance caused by the traits of different semantic classes. In most cases, samples tend to cluster according to their semantic category (i.e, class labels) because of the dominant features of the classes. Therefore, the unsupervised domain discovery is not

as straightforward as applying a standard clustering algorithm such as k-means [60] on the data samples.

In Sound-Adapter, we propose a multi-task neural network architecture that uses a *Gradient Reversal Layer* [61] to remove class-specific features from the audio representation along with a structure-preserving *auto-encoder* branch to keep the domain specific traits intact. On top of that, we use a deep embedding clustering branch to cluster audio samples based on the acquired *domain-dominant* representation to get the desired domain membership information. Finally, for acoustic event recognition, we propose a *Domain-Adaptation* layer to align distributions of different domains for multi-source audio domain adaptation. We use the domain membership information acquired from the domain discovery network to remove domain-specific distribution shifts from the activation outputs of the acoustic event classifier in order to improve the accuracy and robustness of the classifier.

To evaluate Sound-Adapter, we collect an empirical audio dataset in a testbed containing five types of microphones and four different environments. We evaluate both Sound-Adapter’s individual algorithms and end-to-end pipeline’s performance on the empirical dataset as well as a publicly available dataset [62] under different scenarios. Results demonstrate that Sound-Adapter has a mean accuracy of 87% for domain discovery in a five-domain scenario and improves the accuracy of the audio classifier by up to 21% compared to the state-of-the-art.

The contributions of this chapter are the following:

- We present Sound-Adapter– the first system to address multi-source audio domain adaptation for accurate and robust acoustic event classification.
- We propose a novel unsupervised neural network architecture to cluster audio samples according to their recording devices to infer their domain information.
- We propose a domain-adaptation layer for multi-source to multi-target domain adaptation for domain-invariant acoustic event classification.
- We collect an empirical audio dataset containing five types of microphones’ recordings and 2000 audio clips of total 2.8 hours. We will release the complete dataset to the public upon publication.

- We evaluate Sound-Adapter extensively both on the empirical and a publicly available dataset. Sound-Adapter has almost 30% and 21% higher accuracy for domain discovery and acoustic event classification compared to state of the art algorithms for a multi-source domain scenario.

3.2 Domain Shift in Acoustics

Domain shift refers to shifts in the data distribution between training and inference stages. In this section, we describe the causes and effects of domain shifts in acoustics as well as the challenges to address the issue.

3.2.1 Causes of Acoustic Domain Shifts

In acoustics, several factors contribute to domain shifts, e.g., heterogeneity of recording hardware, diversity of sound sources, changes in acoustic environment, positioning of the recorder, software encoding of audio files, and combination of two or more of these factors. We highlight two major factors:

- *Device Heterogeneity*: An audio signal goes through multiple hardware and software components and signal processing phases before getting recorded by the system. Each of these components may alter the signal characteristics and introduce variability in the recorded signal. For instance, at the microphone hardware, signals capture sensor-specific variability such as changes in chemical compounds in the microphone chip and imperfection in the chip assembly line [63]. Then the DSP chips perform noise filtering and audio enhancement – which add variability due to inconsistent responses to different frequencies [64, 65]. Signals get perturbed further as they are processed by the OS-dependent implementations of the system calls [13, 66]. Implementation differences in audio processing algorithms such as microphone array processing and beam forming add an extra layer of variability [13]. All these steps introduce variance in the recorded audio [13, 63].

- *Acoustic Environment*: Audio signals are also influenced by the environment where they are recorded. Room parameters such as the shape, size, and the presence or absence of insulating or reflective materials contribute to the amount of reverberation that gets added to the signal. Additionally, the relative position between the sound source and the microphone influences the acoustic characteristics as the attenuation of the signals depends on the distance. Moreover, background noise present in ambient environment varies the signal-to-noise ratio (SNR) of the captured signal. As the acoustic features are influenced by the environmental effects, the performance

of machine learning-based audio classifiers are also influenced by it. Recent works [?, ?] report an average classification accuracy drop of 22.5% when the classifier is employed in different reverberant environments.

3.2.2 Domain Effect on Acoustic Feature

Although a sound recorded in different domains may sound similar to our ears, there are subtle differences in their acoustic features that embed these device and environment dependent variability. To demonstrate the effect of domain heterogeneity on acoustic features, we conduct an experiment where we play audio clips from ESC-10 [67] dataset on a speaker and five different types of microphones, equidistant from the speaker, record the audio simultaneously (Figure 3.2). We use ESC-10 since it is commonly used in machine learning-based acoustic event recognition algorithm evaluations [68]. The details of the experimental setup is described in Section 3.6.1.

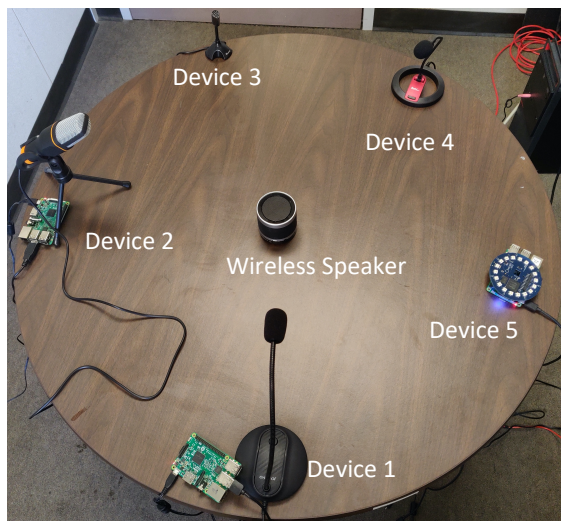


Figure 3.2: Data collection with five types of microphones.

In Figure 3.3, we observe that for the same sound (a five second segment of *sneezing*), signals recorded by different microphones have visible differences in their spectrogram [69]. Different microphones and their positional differences (with respect to the room) cause variations in the spectral representation. Thus, acoustic features computed from them are likely to be different.

3.2.3 Why Domain Inference is Hard?

Predicting the domain of a given sound clip is non-trivial since the perturbation in an audio recording due to domain shifts is subtle, and the acoustic characteristics of a specific sound type is

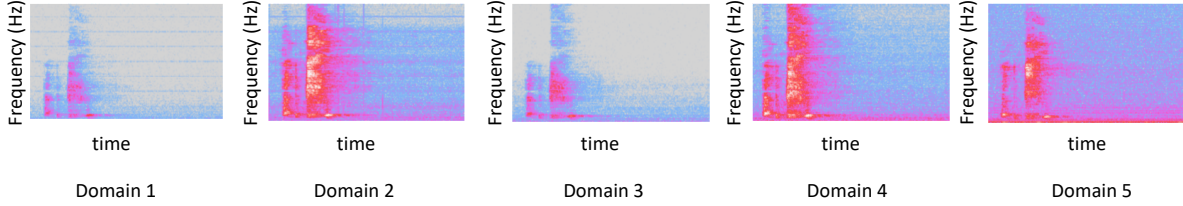


Figure 3.3: The same sound clip recorded by five different devices have significant variations in their spectrograms.

generally dominated by the actual content of the audio. In other words, the same sound recorded by two different devices is harder to distinguish than distinguishing two different sounds recorded by the same device.

In an unsupervised scenario – where we do not have the domain information corresponding to each audio clip – the problem is even harder to solve. Common clustering algorithms like k -means [60] fail to cluster sounds based on their domain for the same reason discussed above.

More formally, we identify two types of information embedded into an audio clip – *Domain Descriptors* that capture domain-specific information, and *Event Descriptors* that capture the information corresponding to the actual content of a specific sound type. Of these two, event descriptors are far more dominating than the domain descriptors. This becomes evident when we cluster audio clips characterized by any feature representation that does not explicitly nullify the effect of event descriptors. Due to the dominance of event descriptors, audio clips having the same class-label cluster together – as opposed to forming domain-wise clusters.

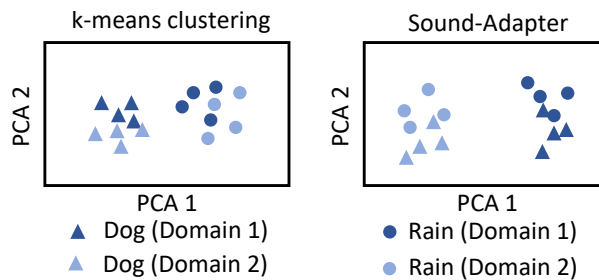


Figure 3.4: k-Means produces event-wise clusters whereas Sound-Adapter produces domain-wise clusters.

Figure 3.4 (left) shows the result of k -means clustering on audio data from ESC-10 dataset for two randomly picked sound types $\{Rain, Dog\}$. Both sounds are recorded using two different microphones (i.e., there are two domains). The first two principle components are used to visualize the data points. We observe that the audio clips get clustered according to their class types, i.e.,

Rain and Dog – which is not what we want when our goal is to infer the domain. The Figure on the right shows the expected plot where the audio clips are clustered according to their domains – which is what Sound-Adapter achieves in this chapter.

3.3 Overview of Sound-Adapter

3.3.1 Sound-Adapter

Sound-Adapter is an acoustic event¹ recognizer that infers the domain information of the training audio samples without any supervision or prior knowledge of the source domains, and then uses the inferred domain information to remove domain-specific distribution shifts from the acoustic event classifier in order to achieve a highly accurate and robust audio classifier. Sound-Adapter is agnostic to the cause of the domain shift — it removes domain shifts caused by any number and types of causes and their combinations.

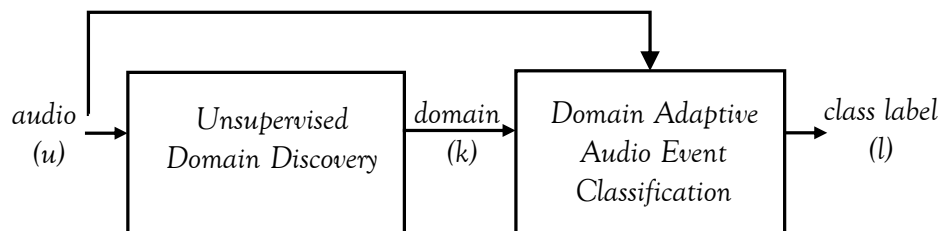


Figure 3.5: Sound-Adapter’s training pipeline

Figure 6.3 shows Sound-Adapter’s two-step acoustic processing pipeline for training a domain-adaptive audio classifier. The first step consists of a *Unsupervised Domain Discovery* module which infers the source domains of the input samples. The second step consists of a *Domain-Adaptive Audio Event Classifier Model* which takes the inferred domains along with the audio samples, and outputs the audio event’s class label. Both of these modules are trained during the training phase. However, the domain discovery module is not employed during the inference phase as the target domain is often known (e.g., a developer often knows the device and/or the environment where the

¹An acoustic event [70] is defined as any limited-duration sound clip that has a semantic name, e.g., laughter, lightening, honks, and so forth.

application will run). The details of these two modules are described in Section 6.10 and Section 4.6, respectively.

3.3.2 Specialty of Sound-Adapter

Unlike previous works in audio domain adaptation [13, 12], Sound-Adapter is able to adapt domains in multiple-source domain scenarios – which does not require domain labels during training. Furthermore, the proposed approach is scalable, i.e., it does not require us to learn domain transformation for all pairs of source and target microphones.

Not requiring the domain labels during the training phase is particularly useful in creating robust classifiers, when a developer has access to many audio clips and/or audio datasets (e.g., crowd-sourcing [71] scenarios) but the recording meta-data is not available. As there can be various recording heterogeneity (e.g., microphones, acoustic processing software, and acoustic environments) and they can create numerous domains separately or in combination – the total number of domains can be very large. Hence, a scalable solution to domain adaptation is highly desirable.

During inference, Sound-Adapter receives audio data from the target domains. These target domains may be completely different from the source domains. However, it is fair to assume that the ground-truth target domain labels are available to the developer. In the inference step, the test samples are passed through the Domain-Adaptive Audio Event Classifier (which has been trained during the training phase). Finally, Sound-Adapter classifies the audio events from the target domains.

3.4 Unsupervised Domain Discovery

Sound-Adapter’s domain discovery is an unsupervised process whose goal is to cluster audio samples according to their domains – given only the audio clips and their class labels. Note that, class labels are used only during the training. Once we get the cluster centroids we do not need the class labels to infer source domain.

To address this, we propose a novel algorithm that systemically removes event descriptors from the audio features and clusters them according to the domain descriptor. This is achieved by employing a *Multi-Task Neural Network* [72] that breaks down the domain discovery problem into two steps, i.e., *Event-Agnostic Audio Representation* and *Domain-Wise Clustering*, and simultaneously solves them. Figure 3.6 shows the design of the neural network.

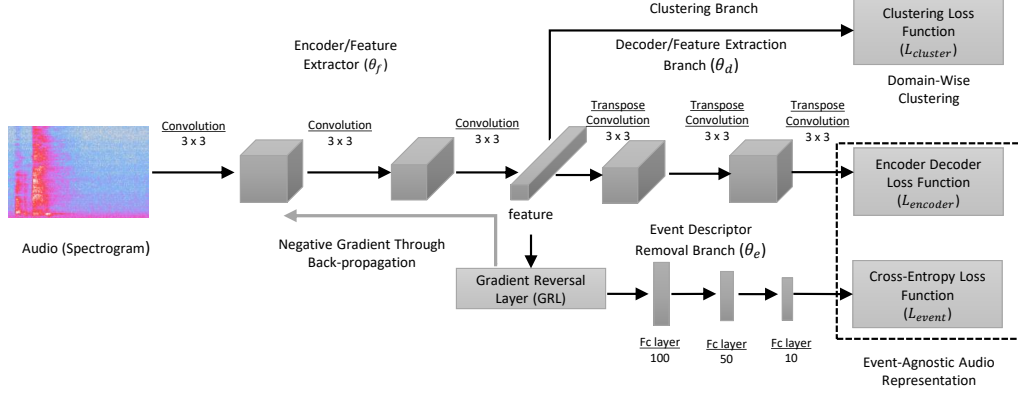


Figure 3.6: The Domain Discovery Network.

3.4.1 Event-Agnostic Audio Representation

The goal of this step is to nullify the effect of event descriptors in order to obtain an event-agnostic audio representation. This representation is used in the next step where we obtain the domain-wise clusters of the audio clips.

- *Baseline Audio Representation:* Prior to describing the event descriptor removal operation, we introduce the baseline audio representation upon which the removal operation is performed. For this, we use a standard *Auto-Encoder* [73] to obtain a lower-dimensional feature representation of the input audio. The benefit of this approach is that it does not require labeled data.

The auto-encoder is shown as the middle branch in Figure 3.6. It consists of an encoder CNN (G_{θ_f}) that encodes the input audio (frequency domain [74]) x of length N to obtain a lower dimensional representation, $G_{\theta_f}(x)$; and a decoder CNN (G'_{θ_d}) that tries to reconstruct the input audio x . To train the auto-encoder, we use the *Mean Squared Error (MSE)* [75] between the actual input (encoder’s input) and its reconstructed version (decoder’s output) as the loss function:

$$L_{encoder} = \frac{1}{N} \sum_{i=1}^N \left(x_i - G'_{\theta_d}(G_{\theta_f}(x_i)) \right)^2 \quad (3.1)$$

After the training, the audio representation we obtain is $G_{\theta_f}(x)$ – which is a lower-dimensional representation of the input audio, but it still contains both the event descriptor as well as the domain descriptor.

- *Event Descriptor Removal:* To remove the event descriptor from the feature representation obtained in the previous step, we employ another branch (the bottom branch) in the multi-task

neural network in Figure 3.6. The key idea behind the event descriptor removal process is to first, train this branch of the network to learn to classify audio events like a regular audio classifier, and then negatively feed this learned knowledge back to the encoder – so that the encoder learns to *forget* or nullify event descriptors when it generates the audio representation.

In our implementation of this branch, we use three fully-connected layers – forming a multi-layer perceptron network [76] – whose objective is to classify the audio events. To train this branch, we utilize the class labels of the audio clips and use a *Cross-Entropy* loss function [76] which ensures that the probability distribution of the output of our network branch matches with the ground-truth.

$$L_{event} = - \sum_{c=1}^M y_c \log (p_c) \quad (3.2)$$

where, M is the total number of audio events in the training dataset; y_c is a binary indicator variable denoting whether a sample belongs to a particular class c (which we get from the ground-truth class labels); and p_c is the probability of the sample belonging to class c that we obtain from this branch.

To enable the negative feedback, we use a *Gradient Reversal Layer (GRL)* [61] whose main objective is to remove traits from feature representation which is responsible for loss minimization. During back-propagation, it multiplies the gradient from its following layer with a negative scalar value λ and then passes it to its preceding layer. During the forward pass, this layer acts as an identity transformation, so that it does not affect the inference process.

- *Why it Works?* Back-propagation of the gradients through the network forces weight updates that minimizes the loss function. In a general scenario, the network would update its weights such that the event descriptors are dominant in the feature representation, and it achieves high accuracy in the audio event classification task. Inserting a gradient reversal layer in between the fully-connected network and the feature from the auto-encoder reverses the gradient during back-propagation. As a result, we are able to form a mini-max situation where the fully-connected layers try to reduce the loss function while the gradient reversal layer makes sure that the preceding layers do not succumb to this. Hence, in effect, the fully-connected layers are extracting the event descriptors from the feature representation while the gradient reversal layer is removing them.

3.4.2 Domain-Wise Clustering

With the removal of event descriptors, we obtain an audio representation that contains primarily the domain descriptors. We perform clustering on these representations to get the desired domain-

wise clusters of the input audio clips. In Sound-Adapter, we employ a *Deep Embedding Clustering* [77] approach to perform the clustering – which requires the number of clusters as the input.

Existing domain discovery algorithms [14, 60] assume that the number of domains is known ahead of time. While this simplifies the problem, this assumption is generally not true as a dataset may contain audio samples from an unknown number of domains (i.e., environments and microphones). Hence, prior to applying clustering operation, Sound-Adapter estimates the number of domains present in the dataset.

- *Determining the Number of Domains:* To deduce the number of latent domains in our dataset, we exploit the class labels in the training data. For each subset of audio samples belonging to the same class, we perform k -means clustering and measure the quality of clustering using a metric (i.e., a weighted sum of Silhouette Coefficient and Calinski-Harabasz Index [78, 79]). This process is repeated for different values of $2 \leq k \leq k_{max}$ to search for the number of clusters k that results in the best-quality clusters.

The above domain counting technique has two caveats: first, since we train the auto-encoder and the clustering branch simultaneously, we cannot use the auto-encoder generated representation in this clustering step while it is still in the making. Hence, we employ a traditional acoustic features, i.e., MFCC [80], to represent audio for this step. Second, this technique yields the most accurate result when we have data from all domains, for all audio events. When the number of domains vary across different classes of audio, the above algorithm outputs the most prevalent number of domains across all classes of audio.

- *Clustering:* In the multi-task neural network of Figure 3.6, a dedicated branch (the top branch) is employed to perform the domain-wise clustering operation. This type of clustering algorithm is known as *Deep Embedding Clustering* [77]. The clustering branch simultaneously updates the feature space to form high-quality clusters and performs clustering on that feature space.

More specifically, this branch performs two main tasks: a) Updating the cluster assignment, and b) Updating the feature space and cluster centroids. A brief description of these steps is the following:

- a) *Update Cluster Assignment:* Following [77], we use *soft assignment* (i.e., a probabilistic assignment) to assign data samples to clusters. The assignment is expressed as probability distributions, i.e, the probability of a data point belonging to different clusters. This probability is derived

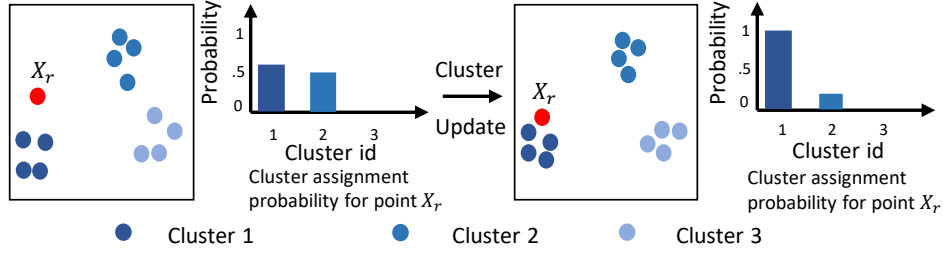


Figure 3.7: Illustration of cluster update step.

using Student’s t-distribution [77] which measures the similarity between a sample X_i ’s feature representation, $h_{X_i} = G_{\theta_f}(X_i)$ and a cluster’s centroid, M_k :

$$q_{ik} = \frac{(1 + \|h_{X_i} - M_k\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{k'} (1 + \|h_{X_i} - M_{k'}\|^2/\alpha)^{-\frac{\alpha+1}{2}}} \quad (3.3)$$

Here, q_{ik} is the probability of X_i belonging to cluster k , and α is the degrees of freedom of Student’s t-distribution.

b) *Update Feature and Cluster Centroids*: Since each audio clip belongs to exactly one domain, ideally, the probability distribution $\{q_{ik}\}$ obtained via the cluster assignment step should have only one peak, i.e., very high probability for one of the domains and near-zero probabilities for the rest. However, due to weak audio representation (which is weak because it is still forming), the cluster assignment step described above does not guarantee single-peaked distributions. To fix this problem, we must update the audio representation and the cluster centroids to reshape the distributions $\{q_{ik}\}$ to its desired shape. Figure 3.7 illustrates how the clusters as well as the cluster assignment (distributions) look before and after the update.

Because there is no ground-truth domain labels to update the feature and the cluster centroids, following self-training techniques [81, 77], we use an update mechanism that relies on its own high-confidence predictions. Specifically, we match the soft assignments, $\{q_{ik}\}$, with an auxiliary target distribution, $\{p_{ik}\}$, where the target distribution puts more emphasis on data points assigned to clusters with high confidence. Furthermore, it balances the cluster sizes by normalizing the loss

contribution of each cluster:

$$p_{ik} = \frac{q_{ik}^2 / \sum_i q_{ik}}{\sum_{k'} (q_{ik'}^2 / \sum_i q_{ik'})} \quad (3.4)$$

To match this target distribution, we use Kullback-Leibler divergence [77] as the loss function – which minimizes the difference between q_i and p_i :

$$L_{cluster} = KL(P||Q) = \sum_i \sum_k p_{ik} \log \frac{p_{ik}}{q_{ik}} \quad (3.5)$$

Once the network reaches saturation in terms of minimizing the clustering loss ($L_{cluster}$) function, we stop the network updating phase and we have our cluster assignments, i.e., the domain labels for the audio samples. For discovering domains for each dataset, the network has to be trained to reach the optimal state.

3.4.3 Putting it All Together

Using the three loss functions corresponding to the three branches of the multi-task neural network of Figure 3.6, i.e., $L_{encoder}$, $L_{cluster}$, and L_{event} , we train the parameters of the neural network using back-propagation. The trainable parameters for encoder, decoder and event descriptor removal, i.e., θ_f , θ_d , and θ_e , are updated as follows:

$$\theta_f \leftarrow \theta_f - \Lambda \left(\frac{\partial L_{encoder}}{\partial \theta_f} + \frac{\partial L_{cluster}}{\partial \theta_f} - \lambda \frac{\partial L_{event}}{\partial \theta_f} \right) \quad (3.6)$$

$$\theta_d \leftarrow \theta_d - \Lambda \frac{\partial L_{encoder}}{\partial \theta_d} \quad (3.7)$$

$$\theta_e \leftarrow \theta_e - \Lambda \frac{\partial L_{event}}{\partial \theta_e} \quad (3.8)$$

Here, Λ is the learning rate. Notice that, during updating θ_f , the gradient of L_{event} is reversed due to the gradient reversal layer. For updating θ_e , the gradient sign of L_{event} is kept intact.

3.5 Domain Adaptive Classification

Using the domain labels² from our proposed domain discovery model, we design a domain-adaptive audio event classifier that fixes the domain shifting problem by normalizing the domain shifts prior to classifying.

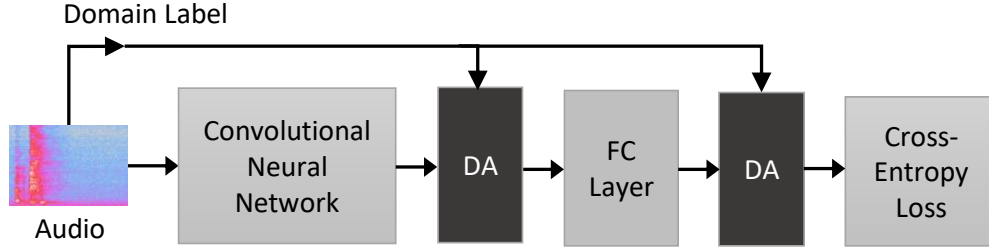


Figure 3.8: Retrofitting an existing audio classifier by inserting domain adaptation layers.

3.5.1 Domain Adaptation Layer

Instead of proposing a completely new network architecture for audio classification, we propose a modular *Domain Adaptation Layer* – which can be used to retrofit popular neural network architectures for audio classification tasks such as VGG [82], Inception [82], AlexNet [83], MobileNet [84] and Resnet [82]. These layers are similar to network adaptation layers recently proposed in computer vision literature [15], however, a major difference is that Sound-Adapter uses the domain labels inferred by the domain-discovery step as opposed to weighted average of all domains to collect more accurate statistics of the neural network layers caused by domain shifts for more effective domain adaptation. Besides, [15]’s domain inference algorithm is semi-supervised, thus relies on training data for inferring domains. On the other hand, Sound-Adapter’s domain discovery algorithm avails the domain adaptation layer to adapt model to different domains without any ground truth domain labels for domain discovery.

Figure 3.8 shows an example of a domain-adaptive audio classifier where we insert two *domain adaptation* (DA) layers to retrofit a neural network architecture having a CNN, a fully-connected layer, and a softmax layer.

²The unsupervised domain discovery module (Section 6.10) provides us generated domain labels such as “ D_1 ”, “ D_2 ”, ... “ D_k ”.

3.5.2 Domain Adaptation Process

The goal of domain adaptation layers is to reverse the effect of domain-shift. Inside a neural network, the effect of domain shift is observed at the activations (i.e., output values) of each layer. Input data from different domains cause different amounts of shifts in the activations of the same layer of the neural network. In order to normalize these shifts, for each input domain, we collect the running means and variances of the activations. These statistics are used to shift and scale the activations to match with a reference distribution having a zero mean and unit standard deviation. Figure 3.9 illustrates the domain adaptation process where activations of a particular layer of a neural network results in two distinct distributions having different means and standard deviations. After applying the domain adaptation layer, both distributions are normalized to a reference distribution having $\mu = 0$ and $\sigma = 1$.

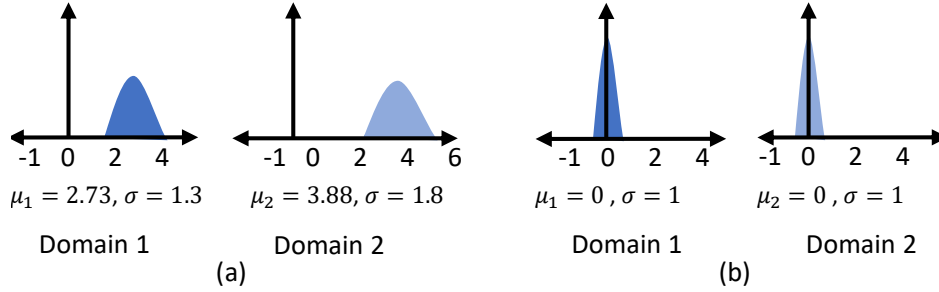


Figure 3.9: Distribution of neural network activations (a) before and (b) after domain adaptation.

3.5.3 Mathematical Formulation

For each batch of input data, at first, the domain membership is inferred using the domain discovery module. Then, for each domain, the neural network activations are normalized by shifting the distribution to the reference distribution with zero mean and unit standard deviation.

Let us assume that for a domain D_i , a hidden layer's activation, A_i follows a distribution, S_i^A . To shift this distribution towards the reference distribution, the domain adaptation layer normalizes it as follows:

$$DA(A_i; \mu_i, \sigma_i) = \frac{A_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \quad (3.9)$$

Here, μ_i and σ_i^2 are the mean and variance of the distribution S_i^A ; and $\epsilon > 0$ is a small constant that helps avoid divide-by-zero error. The value of μ_i and σ_i^2 are estimated for each batch. During back-propagation, gradient passes through the domain adaptation layers to their corresponding

predecessor layer without any alteration.

3.5.4 Neural Network Architecture

Recent works [74, 83] have shown CNN based neural network’s efficiency for acoustic event recognition. Following these earlier works, Sound-Adapter employs a neural network by incorporating CNN and fully connected layers. Each layer of the CNN extracts a feature map from the input data. We empirically determine that three layers of convolutional operations extract adequate information for the later steps of Sound-Adapter. The filters have a size of 3×3 and we increase the depth from 16 to 64 channels for the three layers. We use *Rectified Linear Units* (ReLU), $\sigma(a) = \max\{a, 0\}$ as the non-linear activation function. The output of the third convolutional layer is fed into two *dense* layers [85] with a ReLU activation layer in-between them. Note that, the proposed domain adaptation layers are placed following the CNN layers and each of fully connected layers.

3.6 Evaluation

We conduct a number of experiments to evaluate Sound-Adapter with collected as well as public datasets. In this section, we describe the datasets, followed by component-wise and end-to-end evaluation of Sound-Adapter.

3.6.1 Datasets

- *Empirical Dataset:* To construct a multi-domain dataset, we record audio clips using five types of microphones listed in Table 3.1. The microphones are interfaced with a Raspberry Pi [86], a Laptop [87], and a Desktop Machine to further introduce system induced variability in the data. We play all sound clips from the ESC-10 dataset [67] on an omni-directional speaker, keeping all five microphones equidistant from the speaker. The ESC-10 contains 10 types of acoustic events: *sneezing, clock, dog, crying, rooster, rain, waves, fire, helicopter, chainsaw*. Each audio event has 40 samples of 5 seconds duration. The data collection is demonstrated in Figure 3.2. The microphones are synchronized to record the audio simultaneously. While the synchronous recording minimizes the environment-induced variability, the recorded audio clips are still affected by different multi-paths due to the relative positions of the microphones with respect to the room. In our experiments, we consider a combination of both the device and multipath-induced heterogeneity as domains.

We also collect a dataset containing audio clips recorded with the same device from different environments. We use this dataset to evaluate Sound-Adapter’s ability to discover and adapt

Table 3.1: Device List

Microphone	Interfacing Device
JOUNIVO USB Mic [88]	Raspberry Pi
TONOR USB Mic [89]	Raspberry Pi
eBerry USB Mic [90]	Lenovo Laptop
Connectland 3.5mm Mic [91]	Desktop machine
Matrix Voice [92]	Raspberry Pi

domain shifts due to environmental heterogeneity. We record the ESC-10 dataset in four different environments. A brief description with the environments are presented in Table 3.2. We use a Nexus 5 phone to record data from all four environments, and thus the recorded data is affected by environment-induced heterogeneity.

Table 3.2: Environment Description

Environment	Description
Office room 1	A small office room with only a chair and a table present.
Office Room 2	A mid sized office room with 2 sets of chairs and tables along with a cabinet present.
Bedroom	A mid sized bedroom with one bed, one chair, one table and one bookshelf present.
Lab	A large lab with 3D printer, 4 tables and 4 chairs present.

- *DCASE Dataset*: To complement empirical dataset-driven experiments and to confirm our results further, we use DCASE-2018 (Task 1B) [62] audio dataset that has synchronous recordings from three different microphones. The dataset contains recordings of 10 different types of acoustic scenes such as: *train station*, *public square*, *street*, and *shopping mall*. In Table 3.3, we provide statistics of both datasets.

Table 3.3: Dataset Statistics

Dataset	# Domains	# Classes	# Samples
Empirical Dataset	5	10	2,000
DCASE	3	10	2,160

3.6.2 Performance of Domain Counting

We evaluate the performance of domain counting algorithm with the empirical dataset. To evaluate the performance for different number of domains, we vary domain counts in the dataset by

exhaustively selecting all-possible (i.e., $5C2 + 5C3 + 5C4 + 5C5 = 26$) subsets of the five domains and verify if Sound-Adapter predicts the number of domains correctly. In Figure 3.10 we report the confusion matrix that compares the performance of the algorithm with the ground truth.

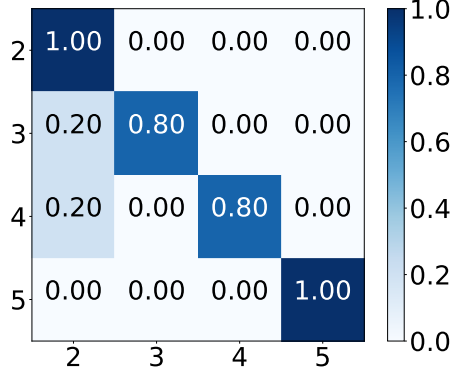


Figure 3.10: Performance of domain counting

Considering the number of times Sound-Adapter predicts the exact number of domains as in the ground truth, its accuracy is 88.46%. However, often in the domain discovery problems, two or more domains (e.g., microphones with similar acoustic response) are so similar that practically they belong to the same domain. Hence, a perfect count of the number of domains is not a strict requirement for domain adaptation. As long as prominent domains are separated by the clustering algorithm (i.e, microphones or acoustic environment with distinct characteristics), domain adaptation works as expected. In Section 3.6.4, we report the end-to-end performance of Sound-Adapter where the effect of domain counting on domain adaptation is reported.

3.6.3 Performance of Domain Discovery

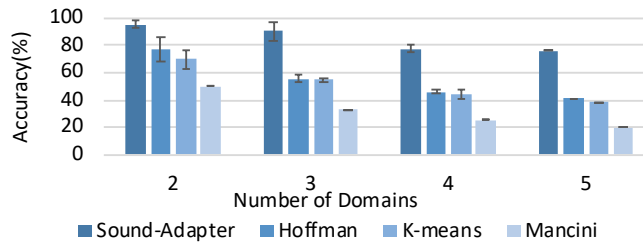


Figure 3.11: Sound-Adapter has superior performance in domain discovery than the baselines for our empirical dataset.

The goal of this experiment is to evaluate Sound-Adapter’s performance of domain discovery under different scenarios and datasets. We use 3 baseline algorithms for comparison: (1) k-means

clustering algorithm – which is popularly used as a baseline for many machine learning algorithm’s evaluation [93], (2) the state-of-the-art Hoffman et al.’s algorithm [14] for domain discovery in image datasets – which we re-purpose for audio datasets, and (3) the state-of-the-art Mancini et al.’s algorithm [15] which is a semi-supervised deep neural network based domain inference algorithm for images. Note that, [15] is not proposed for unsupervised scenario which is the target of our system. We report its result to show that their proposed semi-supervised algorithm is not applicable to unsupervised case .

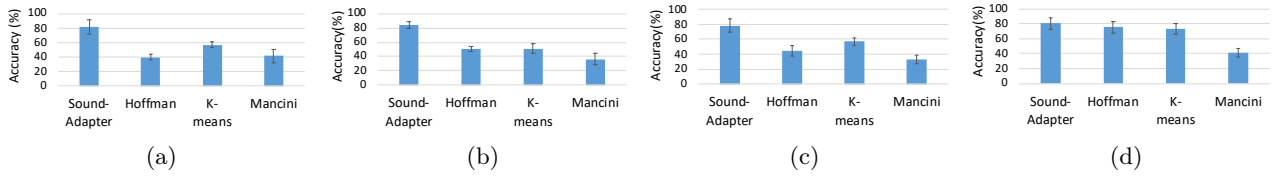


Figure 3.12: (a) Sound-Adapter’s domain discovery algorithm outperforms baselines in scenarios with equal number of audio event and domain. (b) Sound-Adapter’s superior performance also retains for cases with mutually exclusive audio samples across domains. (c) Sound-Adapter beats the baseline algorithms for unseen events. (d) Sound-Adapter’s domain discovery has better performance on DCASE than the baselines.

Varying the Number of Domains Increasing the number of domains makes it harder for an unsupervised algorithm to discover latent domains. We vary the number of domains from two to five by randomly selecting subsets of domains and report the mean accuracy and standard deviation of the accuracy of domain discovery. We use the empirical dataset in this experiment.

In Figure 3.11, we observe that Sound-Adapter shows the best performance among all the approaches. For two domains, Sound-Adapter achieves an accuracy of 96% while Hoffman, K-means and Mancini have 77%, 70% and 50% accuracy, respectively. As the number of domains increases, Sound-Adapter outperforms the baselines by larger margins. For three domains, Sound-Adapter discovers domain with 91% accuracy, whereas Hoffman, k-means and Mancini have a maximum accuracy of 56%. Sound-Adapter’s performance drops to 76% for 5 domains – which is still more than 30% higher than the best of the three baselines’ performance. A random algorithm’s accuracy for 5 domains is $1/5 * 100 = 20\%$.

Overall, Sound-Adapter has a mean accuracy of 87% across all cases. On the other hand, the best of the baselines (i.e., Hoffman) has a mean accuracy of 58% – which is about 30% lower than Sound-Adapter, and the performances of the baselines diminish more drastically than Sound-Adapter

as the number of domains increases. The algorithm proposed by Hoffman relies on handcrafted feature (for audio: MFCC, Spectrogram) to form global cluster among local clusters of domains. However, as the result shows for acoustic data, their algorithm is not well suited, as the domain variance is very subtle in acoustic feature and leads to poor clustering performance. Semi-supervised algorithm proposed by Mancini has almost random accuracy for all cases as their algorithm relies on labeled domains which is unavailable in our case.

Performance with Equal Number of Class and Domains For unsupervised domain discovery, the problem becomes harder when number of domains and events are equal as the clustering algorithm tends to group the data according to the events as opposed to the domains. To show Sound-Adapter’s performance in such cases, we randomly select a subset of events to match the number with domains. For example, when we select 2 domains, we randomly pick 2 audio events for these domains, and perform domain discovery. We vary the number of domains and events from two to five and report the mean accuracy of all possible combinations. In Figure 3.12a, we observe that the mean accuracy of Sound-Adapter for equal number of domains and events is 82%. Hence, the accuracy drop for Sound-Adapter is only around 5% with respect to previous experiment in Section 3.6.3. On the other hand, the performance of Hoffman is particularly worse in this scenario with only 40% accuracy, whereas k-means and Mancini have an accuracy of 51% and 41%, respectively. Hence, unlike the baselines, Sound-Adapter is robust to scenarios where the number of domains and audio events are equal.

Performance with Mutually Exclusive Audio Samples across Domains Our dataset is collected in a way that all the devices record the same audio without any temporal variance. To see the performance in a scenario where the domains have mutually exclusive audio samples, we randomly select subset of samples from each domain such that the domains do not share the samples among them. For this experiment, we also vary number of domains from 2 to 5 to see its effect and report the mean accuracy over all possible combinations. In Figure 3.12b, we observe that Sound-Adapter has a mean accuracy of 84%, whereas Hoffman and k-means show an accuracy of around 50%. Mancini’s accuracy drops to 36% for this case. Thus, Sound-Adapter outperforms the baseline algorithms by atleast 34% for the scenario where the domains have mutually exclusive audio

samples.

Performance with Samples from Unseen Audio Events Although our problem formulation requires audio event labels during training of domain discovery stage, Sound-Adapter’s clustering process is generalizable to samples beyond the audio events that it sees during training (*unseen event*). Once the domain discovery network is trained, Sound-Adapter is capable of domain inference for samples from unseen audio events using Equation 3.3 which requires only the cluster centroids achieved from the training stage. To test Sound-Adapter’s generalizable performance beyond the audio events it is trained with we undergo an experiment where we train the domain discovery network with samples from a subset of ESC-10 audio events. Then, during inference stage we use samples from the audio events that are not present during the training stage i.e., unseen class. We randomly select audio events for training and exclude the rest of them for testing. We vary the unseen classes from 1 to 5 out of the 10 classes of ESC-10 i.e., there are upto 50% audio events not present in the training stage. Following the earlier experiments we also vary number of domains from 2 to 5 and report the mean accuracy here. From Figure 3.12c, we see that Sound-Adapter has almost 79% accuracy for the unseen case which is only a drop of 8% than the seen case. Whereas, Hoffman, k-means and Mancini have accuracy of 44%, 56% and 32%, respectively. Thus, Sound-Adapter is capable of inferring domains for classes for which it does not have training data during domain discovery.

Performance on Public Dataset There are three domains in the DCASE dataset. We conduct experiments with all $3C2 + 3C3 = 4$ combinations of these domains and report the mean accuracy of domain discovery for all the algorithms. In Figure 3.12d, we observe that Sound-Adapter has an accuracy of 81%, whereas Hoffman, k-means and Mancini show 75%, 73% and 42% accuracy, respectively. As the maximum number of domains is 3 in this case, the difference in performance is not as significant as we have seen for the empirical dataset. However, Sound-Adapter still demonstrates superior performance with 6% higher accuracy than the baselines.

Performance on Different Environment In this experiment, we evaluate Sound-Adapter’s domain discovery algorithm’s performance when the cause of domain shift is environmental heterogeneity. The environments are described in Table 3.2. From Figure 3.13, we observe that

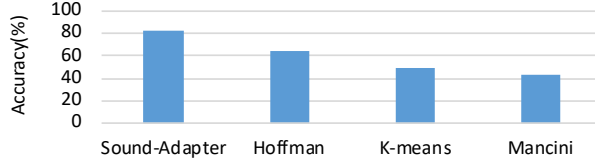


Figure 3.13: Sound-Adapter’s domain discovery algorithm is extendable to domains caused by environment effect.

Sound-Adapter has an average accuracy of around 82%, whereas Hoffman, k-means and Mancini achieves an average accuracy of 64%, 49% and 43%, respectively. This reflects the strength of Sound-Adapter in discovering domains caused by environmental changes during recording.

3.6.4 Performance of Domain Adaptation

In this section, we evaluate Sound-Adapter’s Domain Adaptation (DA) layer’s performance in acoustic event recognition. Here, the audio classifier is trained to classify the acoustic events.

All accuracy numbers for Sound-Adapter, reported in this section refer to the end-to-end performance of Sound-Adapter. To evaluate Sound-Adapter in an end-to-end manner, we execute each step of the system. Sound-Adapter’s domain counting and domain discovery steps are performed in this experiment to infer domain labels of the audio samples. Then the domain adaptation layers are applied to remove the domain bias from the classifier. Therefore, the reported accuracy reflects the performance of Sound-Adapter’s complete audio processing pipeline where an error in the domain discovery step is propagated through the pipeline. Moreover, for Sound-Adapter, we keep the source and the target domains mutually exclusive, i.e., the source and the target domains are different, which is the desired application scenario for our system.

Sound-Adapter is compared with an audio classifier that has the same network architecture with the exception of Sound-Adapter’s domain adaptation layers (i.e., Standard Conv Net). Note that, this system is trained with all of the source domain’s data. This performance signifies the accuracy drop of traditional classifiers when the source and target domains are different.

Here, we also report a system’s performance where the ground truth domain label for each audio sample is known and refer it as *Oracle*. The only difference between Sound-Adapter and this oracle system is the oracle uses the ground truth domain labels for the domain adaptation layers. This system’s accuracy signifies the performance of Sound-Adapter’s domain adaptation layers for domain adaptation. Moreover, the overall impact of domain counting and domain discovery is reflected with

the comparison between Sound-Adapter and oracle.

We also compare with the scenario where the training and testing samples are from same domain (i.e., Same Source - Same Target). This scenario is reported as an ideal result which enjoys the opportunity of the source and the target domains being same. This scenario signifies the performance recovery achieved through the proposed Sound-Adapter.

To compare with current state of the art systems, following Mic2Mic, we train a Cycle-GAN [94] to translate the target domains into source domains. Then we use a standard convolutional neural network with same architecture as Sound-Adapter excluding the DA layer. Note that, we use ground-truth domain labels for Cycle-GAN as it does not have any domain discovery step in its pipeline. We do not include past baseline [12] which relies on data augmentation technique as [13] reports Cycle-GAN’s significant improvement over them.

The source and target domains are picked randomly, and we divide the training and the testing data in five folds as in the ESC-10 dataset. Each experiment is repeated for five iterations for five different combinations of the source and the target domains.

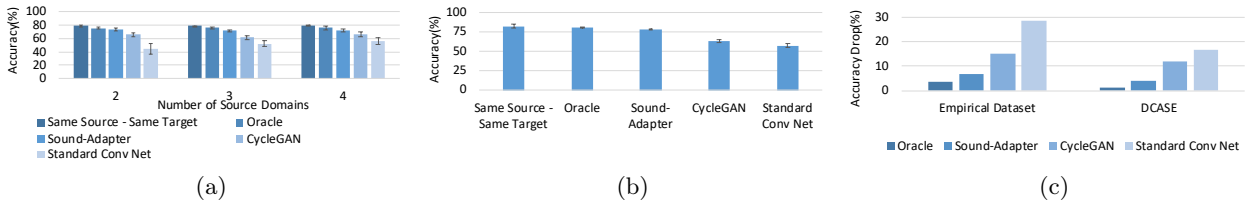


Figure 3.14: (a) Sound-Adapter’s domain adaptation has more than 20% accuracy in average than the same neural network without domain adaptation layer. (b) Sound-Adapter’s domain adaptation is able to improve the accuracy in multi-target domain scenario. (c) Sound-Adapter recovers maximum accuracy drop in comparison with the state of the arts

Performance with Different Number of Source Domains. In this experiment, we vary the number of source domains and evaluate the audio event recognition accuracy of Sound-Adapter for a different target domain. For each of the five domains, we make it a target domain and make 2–4 of the remaining domains the source domains. In Figure 3.14a, we observe that the Same Source - Same Target yields the highest accuracy in all cases as the domains are the same (i.e., no adaptation is required). For two source domains, Sound-Adapter and the Oracle have almost identical accuracy of 75%, which is only 3% lower than the Same Source- Same Target scenario. This is consistent

with our domain discovery algorithm’s accuracy for two domains, which is very high (96%). For Cycle-GAN the accuracy is 65%, which is lower than Sound-Adapter. Without domain adaptation, the performance of a standard convolutional network (Standard Conv Net) is only 45%.

As the number of source domains increases to three and four, we see that the gap between Sound-Adapter and Oracle’s performance increases by 4%. This is due to incorrect predictions of domain membership by Sound-Adapter. However, the performance does not degrade drastically ($\sim 72\%$) as Sound-Adapter has satisfactory performance in domain discovery for all cases. With increased number of source domains, Cycle-GAN and Standard conv net has a maximum of 63% accuracy. Therefore, Sound-Adapter’s accuracy is higher on average due to the use of domain adaptation layers.

Performance with Multiple Target Domains. In Figure 3.14b, we report Sound-Adapter’s performance for multi target scenario, i.e., data from multiple target domains are present during the inference phase. Note that, we assume that the target domain labels are available to us, since typically a developer knows where their application will be deployed (e.g., the device and the environment). We vary the number of target domains from 2 to 3 and report the mean accuracy of acoustic event recognition. We do not use more than three target domains as we have to have at least two source domains to create scenarios having multiple source domains.

As expected, the Same Source- Same Target case has the maximum accuracy of 82%. The oracle and Sound-Adapter achieve 80% and 78% accuracy, respectively. On the other hand, Cycle-GAN and Standard Conv Net (without any domain adaptation layer) achieves a mean accuracy of 63% and 57%, respectively. Therefore, it is evident that even in multi-source multi-target scenarios, Sound-Adapter sacrifices only 2% – 4% accuracy when compared to the Same Source - Same Target and the Oracle. Moreover, Sound-Adapter improves the accuracy of a traditional convolutional neural network by more than 20%. In case of Cycle-GAN, the performance drops than a single target scenario as multiple target domains being translated to source domains cause additional error. The slight improvement in Sound-Adapter’s accuracy compared to the previous experiment (Figure 3.14a) is due to random domain partitions for source and target.

Overall Accuracy Gain. In Figure 3.14c, we report the loss of accuracy of Sound-Adapter and the three baseline algorithms with respect to the Same Source - Same Target scenario. A lower accuracy drop indicates a better domain adaptation capability. In Figure 3.14c, we observe that the Oracle has an accuracy drop of 3.7% and 1.3% for the empirical and the DCASE dataset, respectively. For Sound-Adapter, we observe a drop of 6.7% and 4.1% for these datasets. On the other hand, Cycle-GAN has 15% and 12% accuracy drop. The standard convolutional neural network has an accuracy drop of over 28% and 16% for the two datasets. Therefore, it is evident that Sound-Adapter recovers 12.6% – 21.7% loss of accuracy through its domain adaptation capability.

3.6.5 Beyond Acoustics

Sound-Adapter’s domain discovery algorithm does not rely on handcrafted features and thus it can be applied to other sensing modality such as image data. Out of curiosity, we conduct an additional experiment to evaluate Sound-Adapter’s domain discovery algorithm’s performance on image datasets. We use two datasets: MNIST [75] and MNIST-M [61], which are standard datasets for evaluating DNNs on image processing tasks. MNIST contains 55,000 examples of handwritten digits. These samples are all in black and white format. MNIST-M consists of MNIST’s digits blended with random color patches from the BSDS500 dataset [61]. Thus, these two datasets create two image domains having a total of 10 classes and 110,000 samples.

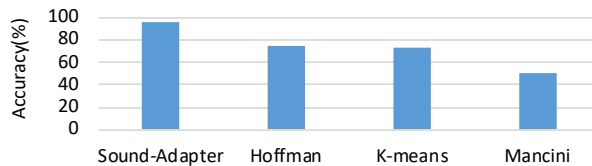


Figure 3.15: Sound-Adapter’s domain discovery algorithm is extendable to image modality.

In Figure 3.15 we observe that Sound-Adapter has an accuracy of 96%, which is 22%-23% higher than the state-of-the-art Hoffman, standard k-means algorithms and Mancini. Hence, we conclude that Sound-Adapter’s domain discovery algorithm is promising to other sensing modalities beyond audio such as images. We leave an in-depth investigation of this as our future work.

3.7 Discussion

- *Scalability of .* An alternative to ’s way of multi-source domain adaptation would be to first discover source domains using the proposed domain discovery algorithm in this chapter and

then repeatedly apply existing single-source to single-domain adaptation algorithms. However, this approach is not scalable and impractical since there are so many different devices, microphones, and environments that obtaining pair-wise training examples for all possible pairs of source and target domains is hard to collect as it calls for a large training dataset ($\mathcal{O}(k^2)$ times larger for k domains). Furthermore, even for a repeated application of single-source to single-domain adaptation, ’s domain discovery algorithm is necessary to find the source domains.

We evaluate ’s unsupervised domain discovery algorithm’s performance for up to five domains. To the best of our knowledge, has been evaluated for the highest number of domains in the literature – including the computer vision literature. For instance, previous supervised and unsupervised approaches [14, 15, 95, 96] to domain discovery in image datasets report results using only two domains in the dataset. Note that, when the training set contains data from numerous sources or domains, the model is not biased to any particular set of domains, rather it becomes more generalized. Therefore, it is more imperative to find and remove the effect of domains when the number of domains are limited.

- *Portability of* . To the best of our knowledge, the maximum classification accuracy of any algorithm on ESC-10 dataset is 92% [97], which is higher than any of the classifiers presented in this chapter. The reason behind this is that in [97], the model is trained and tested on the same original dataset — without considering any environmental noise, and there is no domain shift between the training and the testing data. On top of that, [97] uses transfer learning from visual data to train the classifier. In our empirical dataset, the audio clips from ESC-10 are recorded with microphones – which adds an extra layer of environmental noise to the data. Moreover, we do not perform transfer learning. Therefore, even for the same-source same-target scenario, our result is not as high as [97]. However, ’s proposed domain adaptation layers are modular, and can be integrated to any network architecture including [97].

- *Need for Class Labels*. ’s domain discovery algorithm requires class labels as the input during the event descriptor removal step inside the domain discovery module. This requirement is, however, only for the training phase. Once we train a domain adaptive model using our domain discovery and adaptation algorithm, during inference we feed the ground truth domain labels into the proposed domain adaptation layers for audio event classification.

3.8 Related Work

- *Domain Adaptation:* Recent works have explored single source to single target domain adaptation for audio data. [12] proposes data augmentation for training machine learning models with induced sensor heterogeneity effect. Their approach is limited to heterogeneity database which is collected by recording synchronized audio with different microphones. There are hundreds of different microphones available and gathering heterogeneity information in such precise synchronized way for all possible sensors is not feasible. [13] explores cycle-consistent GAN architecture to translate target domain data into source domain data. [?] proposes speech adaptation for cases where labels mismatch between source and target domain. To this end, solves the problem of multi source domain adaptation problem for audio for the first time by proposing unsupervised domain discovery algorithm and domain-adaptation layers for multi-source domain.

For image data, there have been many works in single source to single target domain adaptation [98, 99]. Recent deep learning based approaches [100, 101] propose deep neural networks to extract domain-invariant feature. All these works on image data work with single source to single target domain adaptation. [15, 61] propose deep neural network based solution for multiple source domain adaptation, but they assume to have access to partial or full domain labels as input. In we consider a much difficult scenario for domain adaptation where domain information is not available to us.

- *Domain Discovery:* Device fingerprinting based on audio data has been explored in [63, 102]. These works are based on supervised model i.e., unlike they assume in the training data they have access to device information as label. [14, 103] proposes unsupervised domain discovery for image data with handcrafted feature. Their approach relies on handcrafted feature which is not dominant by domains rather they are mostly embedded with semantic categorical information (Figure 3.11). [15, 96] propose semi-supervised approach to discover domains for image data. However, their approach requires at least partial domain labels from all the domains for domain prediction. In we propose the first deep neural network architecture for unsupervised domain discovery that does not rely on any handcrafted feature. Moreover, is the first system to find domains from audio data in unsupervised manner.

3.9 Summary

This chapter presents Sound-Adapter– the first system for multi-source domain adaptation model for acoustic data. We propose a novel multi-task neural network architecture for discovering

domain information in unsupervised manner. We propose a domain-adaptation layer to domain-wise normalize the neural network activations to align them to match a reference distribution. We collect data from multiple domains and evaluate Sound-Adapter with both collected and publicly available datasets. We compare our system with the state-of-the-art algorithms for domain discovery and domain adaptation by varying the number of source and target domains. We show that Sound-Adapter’s domain discovery algorithm has a mean accuracy of 87% for up to five domains and its domain adaptation step improves the classification accuracy by 21% from a traditional machine learning model for scenarios having multiple source domains.

CHAPTER 4

Domain Adaptation for WiFi-Based Activity Recognition

4.1 Introduction

With the increase in personal computer, smartphones and Internet of things (IOT), WiFi has become an inseparable part of our daily life. Besides communication, the availability of signal characteristics such as the channel state information (CSI) in commodity WiFi chipsets make WiFi an attractive technology for pervasive human activity sensing. While camera-based pervasive monitoring system can detect human activities reasonably well, they pose numerous privacy concerns [104]. On the other hand, wearables such as smartwatches and activity trackers are less effective due to their usage adherence issues and erroneous performance [105]. Therefore, WiFi-based sensing and inference systems have become an alternate non-intrusive solution for activity monitoring, whose feasibility has been demonstrated in applications such as home activity monitoring [16], sleep monitoring [17], controlling devices using gestures [18, 19], and tracking health vitals [20].

Past works [106, 107, 108] on WiFi-based activity recognition assume there is no environment change between training and inference stage i.e., the activity recognition model is trained and tested on data collected from the same environment. However, this assumption makes the proposed solutions hard to scale for real life applications, where data from testing environment is not available during training stage in most scenarios. For example, in a patient activity monitoring system, the developers have to collect large volume of data from all the patients in their nursing homes to train the model for reasonable performance, which is not feasible. Very recent works [109, 110] propose solutions to build environment agnostic activity classifier from WiFi data. However, these solutions either require considerable amount of labeled examples (≈ 50) of all the activities from large number of domains (≈ 20) or require multiple WiFi access points (≈ 6).

To the best of our knowledge, we are the first to propose a WiFi-based activity recognition system WiDeo, that is able to adapt to new environment (*domain*) with a minor calibration step and is capable of working with only one WiFi access points. Our proposed system requires only one

labeled example per activity in the new environment which relieves the developers and the users of the system from painstaking task of collecting and labeling large scale training data for every new environment. Moreover, our proposed system’s reliance on only one access point makes it an ideal choice for existing infrastructure as most indoor environments are equipped with one WiFi access point.

To achieve this, we propose a novel framework for extracting representation from raw WiFi signal with the supervision of visual data to model the variance in the WiFi signal that correlates with body part movements. In the training phase, it is fair to assume the developers have access to camera to collect video data along with WiFi signal. However, during inference stage, the visual data is not needed once the feature extraction network is trained and only WiFi signal is used for activity classification. Moreover, our proposed framework is the first to apply transformer based self-attention architecture in the context of WiFi signal. Then, we also propose a adaptation framework for updating the representation learning network using only one labeled example for each activity. This yields the first one-shot environment adaptation for WiFi-based activity recognition.

We develop WiDeo using Intel Network Interface Card (NIC) 5300 [25] which captures the WiFi CSI data. To develop the machine learning models, we collect training and testing data from four volunteers in five different environments. We show that, using the proposed domain adaptation method WiDeo is able to recover 28% accuracy than state of the art algorithms when the training and testing environment is different. Moreover, we show that the proposed RF-visual joint feature improves general activity recognition’s performance.

The contributions of the chapter are the following:

- We propose a novel representation learning for WiFi signal using video supervision to create a joint feature space, that maps the variance between WiFi signal and change in body part locations.
- To the best of our knowledge, we are the first to propose an algorithm for environment adaptive WiFi-based activity classifier that is able to adapt to new environment with only one labeled examples per activity class.
- We collect data from five different environments from four volunteers for activity recognition and report the performance of our proposed algorithm using the collected dataset.

4.2 Preliminary Study

In this section, we undergo some preliminary studies to determine the effect of environment change in WiFi signal and the overall effect on activity classifier.

4.2.1 Effect on CSI

In this experiment we perform experiments to find the effect of environment on WiFi signal. More specifically, we present the difference in channel state information (CSI) signal for two different rooms for the same activity (*raise*). We collect WiFi data while a participant performs the same activity in two different room (an office room, a large lab). The details of the room dimensions are presented in Section 7.7.

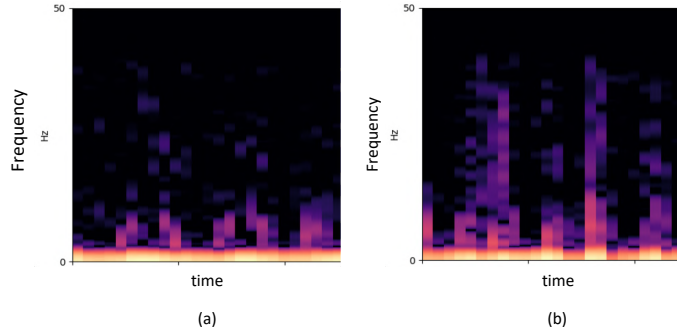


Figure 4.1: Spectrogram of same activity in 2 different room

In Figure 4.1, we present the spectrogram of the CSI signal for the two environments. We see that for the same activity performed by the same person in two different environments produce different frequency response. This is due to the fact that different rooms have different dimensions and furniture. Therefore, the multipaths received at the receiver change with different environments and result in variance in the WiFi signal.

4.2.2 Effect on Activity Classifier

In this section, we show the effect of the variance caused in WiFi signal due to environment change. We train an activity recognition model [23] with training data collected from one environment (office room) and tested it with inference data collected from a different environment (large lab).

From Figure 4.2, we find that when the trained model is tested with data collected from the same environment the classifier’s accuracy is around 86%. However, the same model tested with data collected from different environment results in a massive accuracy drop to around 27%. This is due to the change in WiFi signal because of the environment change. In this figure, we also report

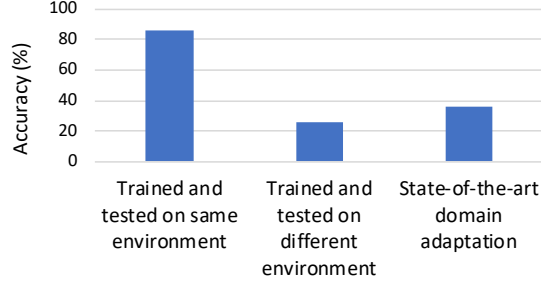


Figure 4.2: WiFi-based classifier’s performance drops when environment changes between training and testing phase

[109]’s performance of domain adaptation when it has only one training data from data collected from lab, as our primary target is one-shot domain adaptation. We find that [109]’s performance is also not satisfactory (37%) as the algorithm proposed in [109] requires large number of labeled examples from the target domains.

4.3 WiDeo System Design

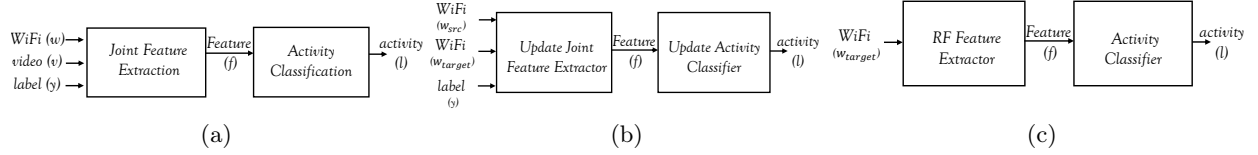


Figure 4.3: (a) Training pipeline for source domain (b) Pipeline for domain adaptation (c) Pipeline for inference

WiDeo is a WiFi-based activity classifier that is capable of adapting to new environment with minimum supervision (one labeled example per class). We denote the environments used for training data collection as *Source domain* and the environments used for inference stage as *Target domain*. WiDeo takes a short-duration WiFi CSI stream as the input, and processes the CSI stream through a signal processing pipeline to classify it as one of activity types present in training phase. During training phase, the first step is to train a RF-visual joint feature extraction network for generating representation from WiFi signal where we use both video and WiFi data as input for source environment. The next step is to train a classifier for activity recognition using the extracted representation from joint feature mapping step. These two steps are illustrated in Figure 6.15(a).

For target domain, we propose a domain adaptation step to update the parameters of the joint feature extractor using training data from both source and target domain. In this step no visual

data is needed. With the updated joint feature mapping we fine-tune the parameters of the classifier for an improved performance. These steps are depicted in Figure 6.15(b).

During inference, WiDeo only requires WiFi data from the target environment to recognize the activity. Using the domain adapted feature representation and classifier our proposed system is able to infer activities with very few labeled examples from the target environment. The inference pipeline is depicted in Figure 6.15(c).

4.4 Rf-Visual Joint Representation

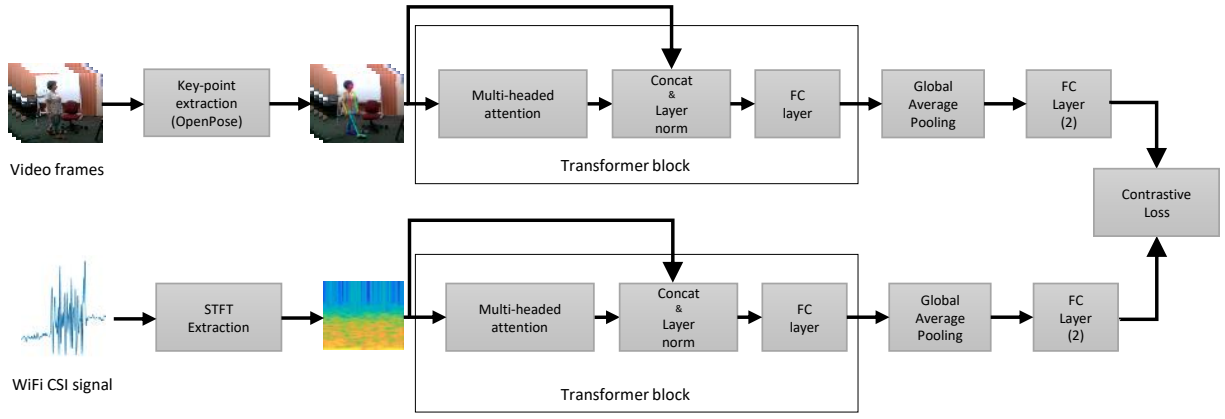


Figure 4.4: Joint Representation Learning using video and WiFi signal

WiFi signal gets reflected from the user’s body as well as from the objects present in the environment while travelling from the transmitter to receiver. Therefore, a change in the environment results variation in the received WiFi signal, which causes variance in the feature extracted from the WiFi signal, even for the same movement performed by a user. Therefore, while modeling human movement from WiFi signal it is difficult to capture the signal variance caused only by user’s movement. Recent development in computer vision literature [111, 112, 113, 114] proposes deep neural network model for extracting body key-points’ (wrist, hand, leg, ankle etc.) locations from only RGB image. Using the models proposed in [113], we aim to extract feature from WiFi data that effectively models the variation caused by human body parts only. The main idea of our proposed representation is to project feature extracted from visual domain (RGB image) and WiFi domain in a joint latent space. In this way, the feature extraction module learns representation for the WiFi signal that correlates with the changes of body part locations during an activity from the

visual domain. To achieve this goal, we propose the following steps:

4.4.1 Video Feature Extraction

The main objective of this step is to extract feature from video (RGB) frames that models the change of body parts of the subject while performing any gesture or activity. To do this, we need to find the locations of user’s body parts and then model the movement of body parts across consecutive video frames. This step yields feature that captures the body movements that constitute the activities. The two steps are described below:

Body Key-point Extraction For extracting co-ordinates of human body parts from RGB image we use OpenPose [113] which is a realtime 2D pose estimator based on deep neural network. Using OpenPose it is possible to jointly detect human body and as well as body part co-ordinates. For our purpose, we extract 25 body key-points from each frame of the video that includes the 3D co-ordinates of shoulder, wrist, hip, ankle, heels etc. This body key-point extraction is crucial for environment invariant activity representation learning.

Activity Modeling Sequential position change of human body parts constitute different activities. For example, to perform the activity of waving, a person has to raise his or her hand and then move it sideways. On the contrary, for walking both hand and leg movement is required. Therefore, to model different activities it is necessary to model the sequential nature of body part movements. For that purpose, we use the time series of multi-dimensional co-ordinates of body key-points extracted in the previous step. In this chapter, we propose to use *self-attention* module to encode the contextual information from the time-series input signal. Recent works on natural language problems [115, 116] showed the benefits of using self-attention based transformer [51] networks over other sequence model such as recurrent neural networks and its variants (LSTM, GRU). Following their work we use self-attention to extract contextual embedding for each input data at each timestep that contains temporal information from all the timestep’s data. We use multi-headed attention which improves over single-headed attention by focusing on different timesteps and generating multiple representation subspaces. These contextual embeddings are passed through layer normalization [117] layer to remove distribution shift. After that, we put a fully connected layer which is then fed into a global average pooling layer [118] to extract a single vector of embedding from all the time steps.

This vector contains contextual information from across the time dimension which is then fed into another two layer of fully connected neural networks to further extract information that captures the activity’s unique feature. The whole pipeline is depicted in Figure 4.4.

4.4.2 RF Feature Extraction

Similar to video feature extraction the RF (WiFi) feature extraction goes through two steps. First, the raw CSI signal is converted into time-frequency domain and then a sequence modeling step is proposed to extract the movement feature from the WiFi signal. The two steps are described below:

Frequency Domain Representation The raw CSI data are not practical for direct use in the training or in the classification steps, as CSI is usually sampled with high sampling frequency (200Hz [119] to 2.5KHz [120]). Therefore, the dimension of the input vector becomes too large to process efficiently. Moreover, raw CSI signals do not explicitly reflect the intrinsic characteristics of the high-level activity. Hence, like most sensing and inference systems, we extract the time and frequency domain properties of the signal based on spectrogram [121] to obtain S as follows:

$$S = |\text{STFT}(X)|^2$$

where, $\text{STFT}(\cdot)$ denotes *Short-Time Fourier Transform* [121], which estimates the short-term, time-localized frequency content of input CSI signal X .

Activity Modeling Following the video feature extraction module, we also use transformer architecture inspired self-attention module to capture the temporal nature of CSI variance caused by different movement pattern of human body for different activities. To the best of our knowledge, we are the first to propose to use self-attention for representation learning for activity recognition from WiFi signal. We follow the similar architecture to the video feature extraction with fine-tuned hyper-parameters for WiFi-based activity modeling. The network components are illustrated in Figure 4.4.

4.4.3 Joint Space Feature Mapping

In this step, we project the activity representation extracted from both the WiFi and the video input onto the same feature space. The objective of this step is that, the feature extracted from

WiFi and video for the same activity is projected together, whereas for different activity the feature extracted from WiFi and video are projected far from each other by ensuring a certain margin between them. This is achieved by using *contrastive loss function* during the training phase, which is discussed later in Section 4.4.4.

We train the two feature extraction networks for video and WiFi by providing them with pairs of video and WiFi data as the input and a binary target that denotes whether or not they belong to the same class as the output. To sample pairs of data for training the joint space feature mapping network, we randomly select a batch of WiFi signal from the training data. Then, to form the pair we select a video data from the same activity or a different activity based on a uniform random probability. More specifically, for a batch of size N we form $\frac{N}{2}$ positive pairs by selecting WiFi and video from the same activity and $\frac{N}{2}$ negative pairs by mixing WiFi and video data from different activities. During training, the joint feature mapping network learns parameters to project the positive pairs in close proximity and to project the negative pairs far from each other in the feature space.

Once the training finishes, we take the output vector from the last fully connected layer of the WiFi feature extractor—which becomes the desired robust, inter-class similarity aware representation of an activity.

4.4.4 Loss Function

We denote $G_\theta^w(X_w)$ and $G_\gamma^v(X_v)$ as the projections from the video and WiFi feature extraction networks where, G_θ^w , G_γ^v are the feature extraction networks and X_w , X_v are the input WiFi and video data, respectively. θ and γ denote the learnable weights of the networks. For a pair of wifi and video data input, X_w and X_v , we set their distance score, $Y = 0$ when they belong to same class, and $Y = 1$ when they belong to different classes. Finally, we define a contrastive [122] loss function as follows:

$$L(\theta) = (1 - Y) \frac{1}{2} \left| G_\theta^w(X_w) - G_\gamma^v(X_v) \right| + Y \frac{1}{2} \left\{ \max \left(0, \Delta - \left| G_\theta^w(X_w) - G_\gamma^v(X_v) \right| \right) \right\} \quad (4.1)$$

The first term in Equation 5.1 minimizes the distance between a pair of wifi and video from same class whereas the second term maximizes the distance between a pair of wifi and video from

different classes. The term Δ represents the distance margin that the network maintains between data from different classes.

4.5 Activity Classifier

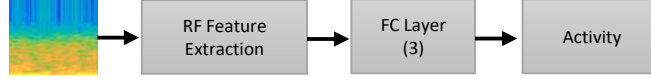


Figure 4.5: WiFi-based activity classifier

We extract feature from WiFi data using the network G_{θ}^w trained in the joint representation step. To classify activity based on this representation we use a neural network consisted of 3 layers of fully connected layer. The last layer outputs the probability of prediction for the sample belonging to a particular activity class. Note that, during this step we freeze the feature extraction network’s G_{θ}^w parameters and only update the fully connected classifiers trainable parameters. To train this classifier, we utilize the class labels of the WiFi data and use a *Cross-Entropy* loss function [76] which ensures that the probability distribution of the output of our network branch matches with the ground-truth.

$$L_{classifier} = - \sum_{c=1}^M y_c \log(p_c) \quad (4.2)$$

where, M is the total number of activities in the training dataset; y_c is a binary indicator variable denoting whether a sample belongs to a particular class c (which we get from the ground-truth class labels); and p_c is the probability of the sample belonging to class c that we obtain from this branch.

4.6 Domain Adaptation

The steps described in Section 4.4 and Section 4.5 are executed during the training phase for the source environment. For the target environment, we propose the domain adaptation step. The goal of this step is to adapt the feature extraction network’s parameters to match the extracted representation of target domain with source domain. Due to the change in environment, the CSI value from the WiFi signal reflected from the user’s body exhibit variance even from the same activity from another environment. Therefore, the feature extraction network does not produce the same representation for the same activity performed in two different environments. To compensate

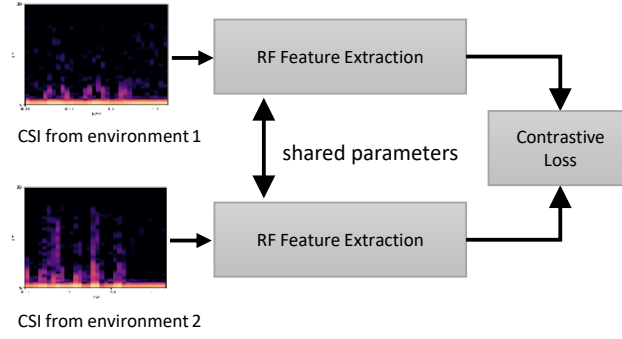


Figure 4.6: Domain adaptation for WiFi-based representation extraction

for this factor, we fine-tune the parameters of the feature extraction network so that same activities performed in different environments produce similar feature.

4.6.1 Update RF Representation

To update the parameters of the WiFi signal’s feature extractor, we propose to use *siamese architecture* [74, 123]. A siamese network is a twin neural network architecture where two identical deep neural networks share their weights. In our case, the siamese neural network takes two WiFi CSI signal as the input and projects a representation based on their class similarity. We initialize the shared weights from the parameters learned during the joint representation step described in Section 4.4. The objective of this network is to learn a representation of the training data which projects similar data points closer while ensuring a certain margin between data points from different classes. This is achieved by using the same *contrastive loss function* during the training phase, which is discussed in section 4.4.4. In the training stage, we have one labeled examples for each activity from the target environment. To train the siamese network we form pairs of positive samples by incorporating random samples of the same class from the source environment with the target environment’s sample. We also generate pairs of negative samples by sampling mismatched pair of samples from the source and the target environment. In this way, we are able to create batches of training data with a few labeled examples from the target domain to fine-tune the RF feature extraction network. The architecture is presented in Figure 4.6.

4.6.2 Update Activity Classifier

Once we update the RF feature extraction network in the previous step, we further fine-tune the classifier’s parameters for the target environment using the labeled examples. More specifically,

we train the classifier with the representation generated from the environment-adjusted RF feature extractor network using cross-entropy loss function as described in Equation 4.2. The fully connected layers in the classifier are initialized with the parameters trained for the source environment. This step further adapts the end-to-end classifier pipeline for the target environment.

4.7 Experimental Setup

In this section, we describe our implementation steps and experimental setup details including the environments we used for data collection.

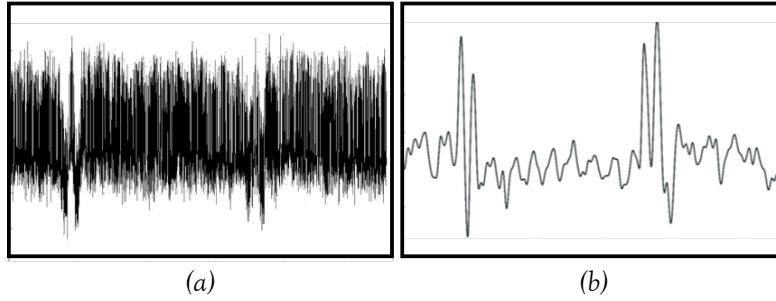


Figure 4.7: (a) Before denoising CSI stream has dominant noise. (b) After denoising all high frequency noise components are removed.

4.7.1 WiFi CSI Collection

We implement WiDeo on an Intel NUC [124] mini PC. The mini PC is interfaced with an Intel NIC 5300 [25] and three omni-directional antennas with 6 dBi gain. To extract CSI values from the NIC card, we use the tool developed by Halperin et. al [125]. This tool provides a modified firmware for NIC 5300 and an open source Linux driver with user space tools to collect CSI information of the received packets. We use a Netlink router as an access point (AP). The mini PC pings the AP periodically at a certain interval and the CSI is extracted from the packet that is received by the mini PC as an echo from the AP. The sampling rate of ping is set at 100 sample per second. Therefore, according to Nyquist theorem [2] our system detects activities with a maximum of 50 Hz, which is sufficient for human activities [120].

We collect data from five different environments with different room size and furniture orientations. Moreover, the position of the user in each room is different with respect to the transmitter and the receiver. The data collection was done by four volunteers. The list of activities are presented in Section 6.7.5. The list of activity classes are: hand raise, clap, swipe, wave, walk. The details of the environments are presented in the Table 4.1. In Figure 6.15 we see different volunteers performing

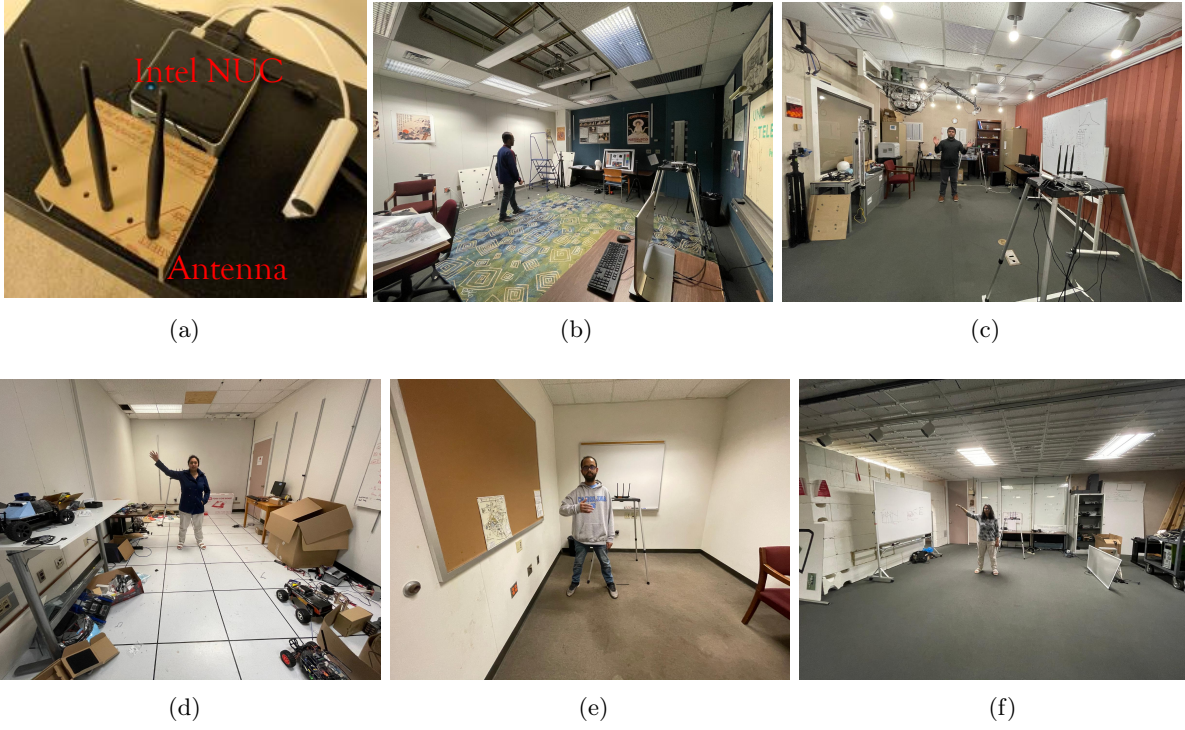


Figure 4.8: (a) Intel Nuc with antennas. Volunteers performing different activities at (b) Lab 1 (c) Lab 2 (d) Lab 3 (e) Office (f) Lab 4.

activities at different data collection rooms.

Table 4.1: Data Collection Environment

Environment	Dimension
Office Room	$3.7m \times 2.8m$
Lab 1	$5.5m \times 5m$
Lab 2	$8.5m \times 5.7m$
Lab 3	$6.5m \times 3.3m$
Lab 4	$10.8m \times 9m$

4.7.2 Noise Reduction

CSI data extracted from the NIC card suffer from dominant noise, which makes gesture recognition difficult. To remove noise, we follow [120]’s PCA-based denoising technique. Since the Channel Frequency Response (CFR) for different subcarriers is a linear combinations of the same set of waveforms with different initial phases, when a gesture is performed, the changes of CFR value in different subcarriers are correlated. To extract the changes in CFR values caused by the movement in different subcarriers, we apply PCA on the CSI stream collected from 30 subcarriers for each

TX-RX antenna pair. The ordered principal components give us the most variances experienced across all sub-carriers. We discard the first principal component stream as it contains dominant effect of noise. The second and the third components contain clear effect of gestures and less effect of noise. We choose the third stream as it has shown better noise immunity. Although the selected stream is less prone to noise, it still contains some effect of unwanted high-frequency noise. Hence, the stream is passed through a Butterworth filter [126] to remove static noise. We set a cut off frequency of 50Hz as human activity is usually below 50Hz in frequency. Thus, we obtain a de-noised CSI stream from all sub-carriers.

In Figure 4.7(a) we see the noisy raw CSI stream of a user performing a gesture repetitively. After performing the denoising step we get a filtered CSI stream in Figure 4.7(b).

4.7.3 Gesture Segmentation

In order to detect the start and the end of a gesture, we instruct the volunteers to perform gestures with a brief pause between them. As the peaks in a CSI stream depends on the initial position of the user, which is variable for different gestures, we cannot segment gestures by directly applying a threshold on the CSI values. Instead, we follow [119] and take the first order difference of the stream which is stable during the pause and fluctuates during gestures. We apply a threshold-based peak detection technique on the first order difference to detect gestures. The segments on the first order difference that are above a predefined threshold are detected as gestures. We refer [119] to the readers for more details on pre-processing of CSI stream.

4.8 Experimental Results

We conduct a number of experiments to evaluate WiDeo with the collected dataset. In this section we present the experiments to report WiDeo's performance.

4.8.1 Performance of RF-visual joint feature

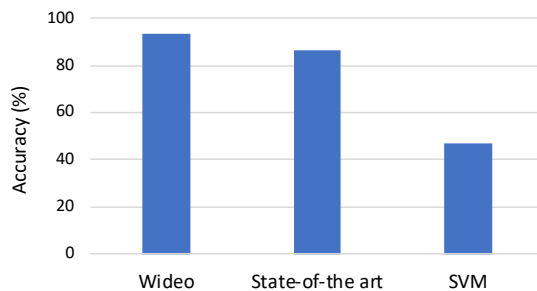


Figure 4.9: Performance of RF-visual joint feature

In this section, we report the performance of our proposed joint feature mapping’s performance for WiFi-based activity recognition. For this experiment, we show that for the same environment training and testing scenario, our proposed video assisted feature extraction model outperforms state-of-the-art WiFi-based activity recognition systems. In Figure 4.9 we report activity recognition’s performance for the proposed joint feature mapping. For comparison, we report the performance of state of the art [23]’s performance. As a baseline we also report non-deep neural network based classifier SVM’s performance with STFT as input feature. From Figure 4.9, we find that WiDeo achieves an accuracy of 93% for activity recognition. The baseline algorithms [23] and SVM achieves 86% and 47% accuracy, respectively. Therefore, it is evident that WiDeo’s RF-video joint feature mapping outperforms the state of the art WiFi activity recognition systems by improving the accuracy with 7%. The joint feature mapping explicitly learns the relationship between movement of body parts with WiFi CSI signal from the supervision of visual data. Moreover, our proposed transformer based representation extraction network is able to extract contextual information from the whole temporal sequence. Therefore, WiDeo is able to achieve higher accuracy than state-of-the-art baselines in activity recognition for same environment training-testing scenario.

4.8.2 Performance of environment adaptation

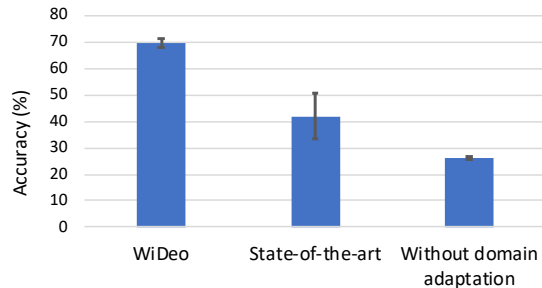


Figure 4.10: Performance of environment adaptation

In this section, we report WiDeo’s performance for environment adaptive WiFi-based activity recognition. More specifically, here the source and target environments are not the same and we have only one labeled samples per activity for the target environment. In Figure 4.10 we report the average accuracy of activity recognition when source and target environments are different. As a baseline, we report [109]’s performance, as to the best of our knowledge it is the only literature for single access point environment adaptation method for WiFi-based activity recognition. We

also report the classifier’s performance when no domain adaptation technique is applied. From the Figure 4.10, we find that WiDeo is able to achieve an accuracy of around 70% in this case. On the other hand [109]’s accuracy is around 42% as their proposed method is not suitable for the scenario where only few labeled samples are present from the target environment. Therefore, it is evident that WiDeo is able to recover almost 28% accuracy in comparison with current state of the art solution for environment adaptive WiFi-based activity classifier. For the classifier without domain adaptation the accuracy drops to around 28%. Thus WiDeo outperforms the classifier without domain adaptation by a margin of 42%.

4.8.3 Performance of person-environment adaptation

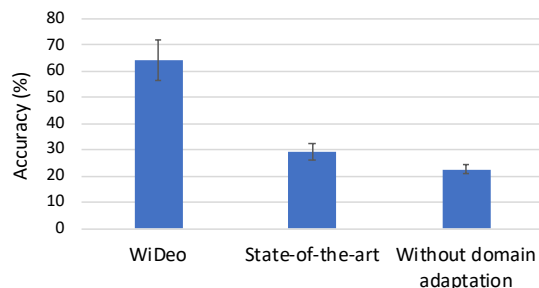


Figure 4.11: Performance of domain adaptation’s performance for mismatched person and environment

In this section, the performance of the proposed system is evaluated for a scenario where source and target domains have different environment and person between them i.e., the participants and environments used in the training and testing dataset are mutually exclusive. In Figure 4.11 we report the average accuracy of WiDeo, [109] as state-of-the-art and a classifier without domain adaptation. We find that WiDeo has an accuracy of around 65% whereas the accuracy of [109] drops to 30%. Moreover, the classifier without domain adaptation achieves an accuracy of 23%. Therefore, WiDeo has superior performance than other state-of-the-arts for the scenario where both the subject and environment changes between training and testing scenario.

4.9 Related Work

- *RSSI based*: WiFi-based sensing have opened the doorway for device-free activity monitoring in the last couple of years. Researchers have used wifi signal characteristics such as signal strength (RSSI) and channel state information(CSI) for activity recognition [127]. RSSI based activity recognitions have been proposed in [18, 128, 129, 130, 131]. However, RSSI based gesture recognition

system have limitations in detecting fine-grained gestures. Moreover, all of these works require training examples to detect a particular activity class. [18, 129] uses pattern matching algorithm to find the best match between the target gesture with pre-defined gestures. Our target in this chapter, is classifying activities and gestures without training examples and our framework can be ported to RSSI based systems.

- *Special Device based:* There have been several works in gesture and activity recognition using specialized devices and radars. [132, 23, 133] use FMCW [134, 135] radio to monitor user activity. [23] trains a CNN classifier to detect human motion. [132] trains a hidden markov model using time-velocity feature for activity recognition. [136] uses a 5-antenna receiver and a single-antenna transmitter to perform gesture classification, in the presence of three other users performing random gestures. They use doppler shifts to match with pre-defined gestures for an incoming test data. It is evident that none of these specialized device based sensing system deals with ctivity recognition without prior examples. They all require labelled training examples for activity recognition.

- *CSI based:* With the availability of CSI from network interface cards, multiple works [137, 138, 139, 140] have emerged which exploit CSI information for gesture and activity recognition. [16] proposes a signal profile matching technique to detect loosely defined daily activities that involve a series of body movements over a certain period of time. [120] proposes correlation between CSI amplitude value and gesture speed to build model for gesture recognition. [141] uses variations in the Channel State Information (CSI) to classify gaits of humans. [119] uses translation based data augmentation technique to make gesture classification models robust to user orientation. [142] proposed multi-person gesture recognition system by generating virtual gesture samples and combining them to create an exhaustive template matching algorithm. Recent works such as [106, 107, 108] use deep learning based techniques such Convolutional Neural Networks to recognize activities from CSI. [143, 106] proposes recurrent neural network based activity classifier , however ours is the first work to model the micro-activity or state transition which constitute an activity. [143] predicts label for each segment of the CSI stream.

- *Environment Aware:* [109] proposed an adversarial network to learn environment independent signal characteristics from gestures. Their work requires considerable amount of labeled examples from multiple environments to adapt to new domain. Very recent work [110] proposed velocity profiles as environment independent feature to solve the problem of environmental effect on CSI.

However, their work relies on multiple WiFi access points to achieve this goal, which is not a feasible solution in many real world scenarios. In WiDeo, we propose solution for a more difficult and practical problem where there is only one access point available to collect the WiFi signal and only one labeled example per activity for the target environment.

4.10 Summary

In this chapter, we propose a novel methodology to adapt WiFi-based activity recognition classifier for new environment with only one labeled sample per activity. To this end, we propose for the first time an algorithm to extract feature from WiFi signal that explicitly models the correlation between the movement of body parts and WiFi signal. To evaluate our proposed method we collect real data from four volunteers from five different environments and report that our proposed method is able to adapt activity classifier to new environment with superior performance than current state-of-the-arts through elaborate set of experiments.

CHAPTER 5

Acoustic Event Recognition Through External Knowledge Transfer

5.1 Introduction

With the ubiquity of acoustic sensing systems, the diversity in their sound types as well as their intended purposes are evolving—making it hard for us to acquire a large corpus of labeled training data for all possible types of acoustic events that an application wants to detect. For example, in applications like sleep monitoring, home intrusion detection, and health monitoring, we may want to collect multiple instances of rare, inconvenient, and unwanted acoustic events such as snoring, breaking in, sneezing, and wheezing. Such requirements create challenges to developing personalized and environment-specific acoustic event recognition systems which primarily rely on user-contributed labeled data for training the classifier. Hence, the research community for long has sought after a solution that can classify audio using few or no training examples.

Audio classification has been a well-studied problem for decades. Dealing with the lack of training data has remained an open and active research problem since then. Recent work [144, 145, 146] in the audio recognition literature that address the problem of *learning from limited training data* mainly apply *data augmentation* [147] and *classifier fusion* [148] techniques. However, these techniques require a certain amount of representative training data to begin with. Furthermore, all existing audio recognition algorithms limit themselves to a fixed set of sound types, and thus when an audio clip from a completely new sound type is presented to them, they have no built-in mechanism to guess the correct label for that unknown sound. For instance, if a classifier learns only two classes {microwave oven, vacuum cleaner} and is tested with a sound of a {blender}, it will never be able to output anything other than {microwave oven, vacuum cleaner} or "sorry".

In this chapter, we study the problem of *learning without examples* in the context of audio inference. The problem is popularly known as the *zero-shot learning* [149, 150], which is an active area of research in computer vision and image classification. To the best of our knowledge, we are the first to apply zero-shot learning to solve the general-purpose audio event classification problem. The

central idea of zero-shot learning is to exploit information or learned knowledge from other sources such as textual descriptions. For example, to describe a *tiger* to a person who knows *cats*, instead of showing them a tiger, we can describe that ‘a tiger is just like a cat but 20 times bigger.’ In this work, we exploit the context-aware representation of English words as an additional source of knowledge when classifying audio clips. Through an advanced audio to text domain projection algorithm, we blur the difference between an audio and its corresponding English word/phrase by representing them in the exact same vector space. We exploit the semantic relationship between English words to classify an audio clip that the system has never heard before. We combine these algorithmic steps to develop an advanced acoustic event classification system that we name—Sound-Semantics.

Developing a system like Sound-Semantics that projects audio representation onto the space of text representation is non trivial and poses several challenges that are addressed in this chapter. First, we propose a new way to learn robust audio features by explicitly learning the similarity (and dissimilarity) between different sound types. We show that such a representation is robust, beneficial to later steps, and yields better features for acoustic data. Second, we propose a neural network architecture to merge text and audio domain so that acoustic data is embedded with the distributional characteristics of English words. This results in the first context-aware audio embedding and paves a way for classification without training data for audio. Third, we propose a two-way classifier that is capable of classifying audio clips from previously *heard* as well as *unheard* sound types. This makes Sound-Semantics a generalized system for classifying all types of acoustic events.

To demonstrate the performance of the proposed algorithms in real-world scenarios, we develop a smartphone application that implements the end-to-end pipeline of Sound-Semantics. A user of the application simply inputs a list of sound types along with zero or more training examples for each type, and the system is capable of learning and accurately classifying all sound types in real-time. We deploy the Sound-Semantics app in two real-world scenarios (kitchen and street sound recognition) and show that the system is able to detect acoustic events with or without training samples with an accuracy of up to 90%. We also evaluate Sound-Semantics’ performance on a popular empirical dataset and report the performance of the proposed algorithms.

The contributions of this chapter are the following:

- We describe Sound-Semantics, which is the first system that addresses the problem of learning without examples for general-purpose acoustic event classification.
- We propose a new approach to learn robust audio representation to preserve better semantic relationship between similar sounds. This representation by itself can be used in developing robust acoustic event classifiers when we have training data for all the classes.
- We propose the first algorithm that exploits semantic knowledge in text to enable classification of audio clips of sound types for which a classifier has never been trained for.
- We develop a light-weight mobile application that executes the audio classification algorithms in real-time and conduct empirical and real world evaluations to test the algorithms as well as the end-to-end performance of the mobile application.

5.2 Concept Applications

Sound-Semantics is an advanced acoustic event recognition system that beats the state-of-the-art in scenarios where we want to recognize a wide variety of sounds, but collecting enough audio clips for each type of sounds is not feasible. In the machine learning literature, this is known as the *class imbalance* problem [151], where the number of training examples varies significantly from one class to another. Especially, in rare event detection problems, such as anomaly detection, fraud detection, and medical diagnosis, often we do not have enough training examples of the primary event of interest. This results in classifiers that suffer from high false negative rates and misclassify critical events. Sound-Semantics solves this problem in a unique way, and accurately classifies acoustic events even if some of the sound types have a few or no training examples. This capability of Sound-Semantics enables many indoor and outdoor applications that were not possible before.

5.2.1 Indoor Usage Scenarios

There are many indoor audio sensing and recognition scenarios such as sleep monitoring [6], bathroom activity monitoring [9], home appliance monitoring [152], asthma severity detection [11], music genre recognition [153], and context recognition for voice assistants [2], where using Sound-Semantics can dramatically improve the performance as well as the intelligence of an application. Because each indoor acoustic environment is different, these systems typically rely on user-contributed audio clips to train the classifiers. However, collecting a large number of audio clips for each type

of sound is an inconvenience to an end-user. In such cases, Sound-Semantics is able to recognize sounds even if a user provides no sample audio for some of the sound types.

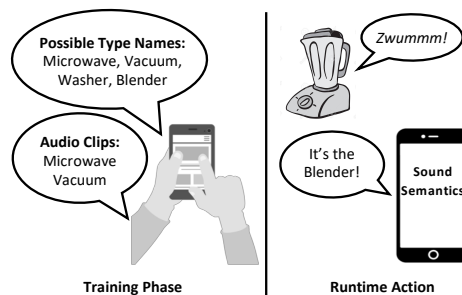


Figure 5.1: Kitchen activity monitoring application.

For instance, consider the simplified *kitchen activity monitoring* scenario in Figure 5.1 where a user wants Sound-Semantics to monitor four types of kitchen appliances or activities such as {microwave oven, vacuum cleaner, dish washer, blender}. The user provides Sound-Semantics the names of these four types of sounds. Sound-Semantics asks the user to record one or more audio clips for each sound type. However, the user only records and provides audio clips of two of these types, i.e., {microwave oven, vacuum cleaner} and does not provide any audio clip for the other two, i.e., {washer, blender} as collecting dish washer and blender’s sound requires more work from our busy user. Despite the lack of sample audio clips, Sound-Semantics is able to recognize all four types of sounds at runtime, including the blender—for which Sound-Semantics was not provided any audio samples during the training phase.

5.2.2 Outdoor Usage Scenarios

Like indoors, Sound-Semantics has the potential to make outdoor sound recognition applications far more effective than they are today. There are many outdoor acoustic event recognition applications, such as street activity monitoring [154], violence detection [155], and pedestrian safety applications [80], where the number of sound types involved in the application is large, and the quality of the sound varies due to distance, background noise, and acoustic environment. It is not feasible to collect audio samples of all types of sounds under all possible combinations of distance, background noise, and acoustic context. The use of Sound-Semantics simplifies the task of training acoustic event classifiers as it only requires a list of possible sound types and example audio clips of some. For instance, in a *street monitoring* application, it is relatively easier for a user to list a possible set of sounds that he/she may encounter (e.g., car engine and tire noise, honks, sounds of

buses and trucks, human chatter and footsteps) than to actually record audio clips of all these types of sounds. Collecting training data for this application is not only cumbersome, but it may also raise privacy concerns as audio recording in public places is, in general, illegal. Using Sound-Semantics, a user can record as many audio clips of as many types of sounds as possible, and the system takes care of learning the rest of the sound types automatically.

5.3 Overview of Sound-Semantics

Sound-Semantics is an advanced acoustic event recognition system that takes a short-duration audio clip and a list of possible sound types (i.e., *tag-list*) as input and processes the audio through an acoustic processing pipeline to classify the audio as one of those given sound types. The design of Sound-Semantics is modular. Computationally expensive modules (e.g., one-time offline training of the classifiers) are run on a server, while the end-to-end audio classification pipeline – from sensing to classification – runs on embedded systems such as smartphones, tablets, and voice assistant devices like Amazon Echo and Google Home.

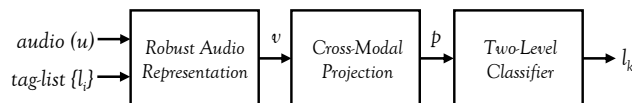


Figure 5.2: Sound-Semantics acoustic processing pipeline.

Figure 6.3 shows the pipeline diagram containing the three processing steps of Sound-Semantics: *Robust Audio Representation*, *Cross-Modal Projection*, and *Two-Level Classifier*. This section provides an overview of each of these steps and defers their algorithmic details to subsequent sections.

5.3.1 Robust Audio Representation

Audio signals from microphones are essentially a time series of numbers. Typically, the sampling rate of a microphone lies between 8 KHz to 48 KHz. This results in 8,000 to 48,000 dimension vectors every second, which is not practical for direct use in training or classification of audio data—especially in CPU and memory constrained embedded systems. Hence, the first step in any audio classification system is to map raw audio data to a lower dimensional vector, commonly known as the *feature* vector, based on its time and frequency domain properties. A wide variety of audio features have been proposed in the literature. The list includes classic time and frequency domain features such as zero crossing rate, pitch, cepstral coefficients, spectral entropy, energy, flux, and roll-off [21]; as well as recent practices where a layer of a pre-trained neural network is considered as

the learned features [156]. A limitation of all these features is that although they can distinguish examples from different classes, they do not preserve the inter-class relationship.

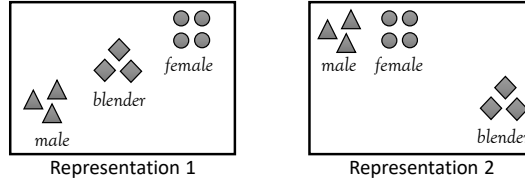


Figure 5.3: Preserving semantics in audio representation.

Figure 5.3 illustrates this concept using a simple three-class problem where two 2D feature representations of examples from the classes {blender, male voice, female voice} are shown. Both representations are able to distinguish examples from different classes. However, the left one does not consider the fact that male voice and female voice are semantically similar, and thus, they should be closer to each other. The representation on the right is a better representation in this regard as it ensures class separability and preserves inter-class semantic relationship.

In Sound-Semantics, we employ a special type of neural network, namely the *siamese network*[123], to learn a robust audio representation that positions similar types of audio signals closer to each other and vis-à-vis. Such a representation helps Sound-Semantics in the next step down the audio processing pipeline where we leverage semantic knowledge from English text to enable classification of unknown sounds. The details of robust audio representation is described in Section 5.4.

5.3.2 Cross Modal Projection

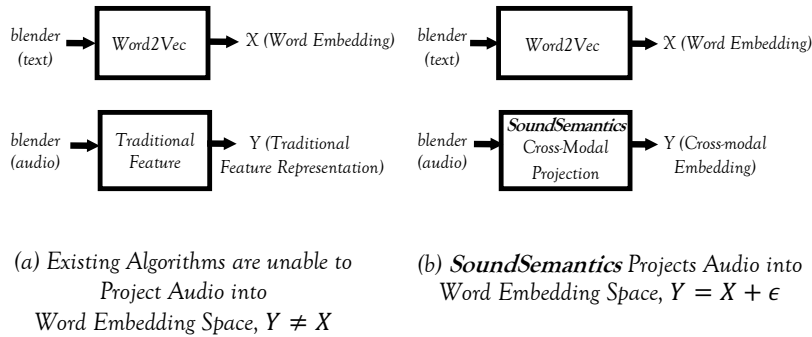


Figure 5.4: Cross-modal Projection maps audio to its corresponding label in word embedding space.

The secret recipe behind Sound-Semantics’ ability to classify completely unknown sounds is the *cross modal projection* step. Through this step, we blur the difference between an audio clip and its corresponding English word, and make them (almost) equal in their feature representation. In other

words, if an English word (e.g., the word ‘blender’) has a known vector representation (e.g., obtained using Google’s word-2-vec [24] on a large dataset like Wikipedia [157]), the goal of this step is to generate the exact same vector when Sound-Semantics is presented with an audio clip of that word (e.g., the sound of a blender). The benefit of this *projection* from audio to text representation is that there are over 150 thousand English words for which researchers in the natural language processing field have created semantically aware vector representations, called the *word embedding*. Such an embedding preserves the contextual relationship among words and puts two words that are similar in meaning or are often used in the same context closer in the representation space. By projecting the audio representation to the word embedding space, Sound-Semantics is able to generate meaningful and context-aware representation of any audio clip – irrespective of whether or not it has heard example audio clips of the same class before.

In Figure 6.8, we illustrate cross-modal projection using the example of a *blender*, whose word embedding is X . The audio representation of blender is passed through the cross-modal projection to obtain Y . In Sound-Semantics, we make sure that $X \approx Y$. In traditional acoustic feature representation, however, the projection is different ($X \neq Y$), and hence unlike Sound-Semantics, it is not possible for traditional features to guess the word label of an unheard sound.

5.3.3 Two-Level Classifier

There are two classes of sound types that Sound-Semantics may encounter at runtime: *heard* and *unheard* classes. The *heard* classes refer to those sound types for which Sound-Semantics has audio clips for training. The *unheard* class, on the other hand, refers to sound types for which Sound-Semantics does not have any training audio clip. For example, in the kitchen monitoring scenario of Figure 5.1, {microwave, vacuum} are heard classes and {blender, washer} are unheard classes. During the classification step, at first, Sound-Semantics employs a *novelty detection* algorithm to determine whether the input audio belongs to a heard class or an unheard class. After this determination, a *nearest neighbor* classifier is used to find the most probable sound type for the input audio. The use of a two-level classifier significantly improves the accuracy of Sound-Semantics, because we propose a holistic approach to classify both *heard* and *unheard* classes together. The details of this two-level classifier is described in Section 6.6.

5.4 Robust Audio Representation

The goal of this step is to obtain a robust feature representation of audio which explicitly learns the similarity and dissimilarity of examples from different classes.

5.4.1 Basic Approach

We employ a *Siamese* neural network [123] that takes two audio clips as the input and determines whether or not they are similar. A siamese network is a twin neural network architecture where two identical deep neural networks share their weights. The objective of this network is to learn a representation of the training data which projects similar data points closer while ensuring a certain margin between data points from different classes. This is achieved by using an *loss function* during the training phase, which is discussed later in this section.

We train the network by providing it with pairs of audio clips as the input and a binary target that denotes whether or not these clips belong to the same class as the output. Once the training finishes, we take the output vector of one of the last few layers—which becomes the desired robust, inter-class similarity aware representation of an input audio clip.

A siamese network has advantages over other types of commonly used neural networks such as the convolutional neural networks (CNNs) as it learns the semantic representation better by explicitly learning a similarity metric. Furthermore, siamese networks are less susceptible to *class imbalance* [158, 123] problem, which is one of the main challenges that we address in this chapter.

5.4.2 Analytical Formulation

We denote $G_\theta(X_i)$ as the projection of an audio clip by one of the twin networks G_θ , where X_i is a standard frequency domain representation (e.g., FFT or MFCC [21]) and θ denotes the learnable weights of the network. For two audio clips, X_1 and X_2 , we set their distance score, $Y = 0$ when they belong to same class, and $Y = 1$ when they belong to different classes. Finally, we define a contrastive [122] loss function as follows:

$$\begin{aligned} L(\theta) = (1 - Y) \frac{1}{2} \left| G_\theta(X_1) - G_\theta(X_2) \right| \\ + Y \frac{1}{2} \left\{ \max \left(0, \Delta - \left| G_\theta(X_1) - G_\theta(X_2) \right| \right) \right\} \end{aligned} \quad (5.1)$$

The first term in Equation 5.1 minimizes the distance between a pair of audio from same class whereas the second term maximizes the distance between a pair of audio from different classes. The

term Δ represents the distance margin that the network maintains between audio clips from different classes.

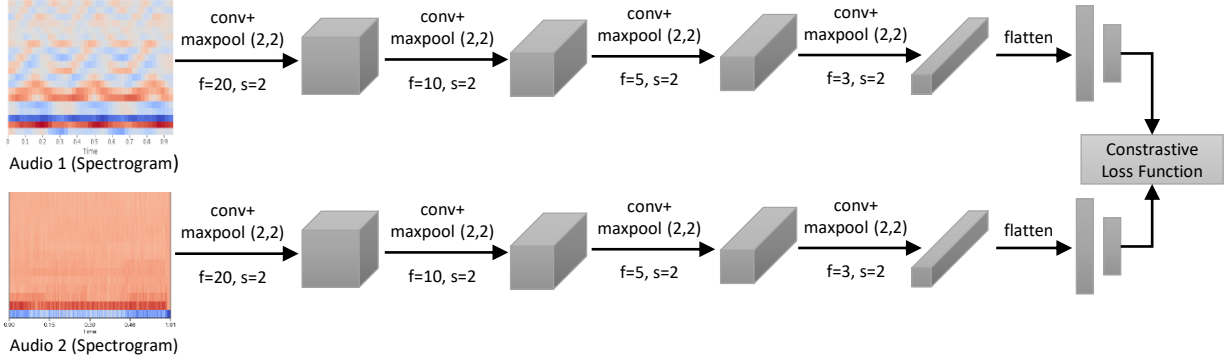


Figure 5.5: The architecture of the Siamese network used in robust audio representation. Here, f denotes the size of the filters and s denotes the stride length of the convolutions. Two audio clips are given as input and the last layer is fed into the contrastive function to learn the similarity between them.

5.4.3 Neural Network Architecture

As the architecture of G_θ , we use four layers of convolutional layers with max pooling and batch normalization. Each layer extracts feature map from the input data and as we go deeper into network more number of filters extract more information. We decided to go with four layers as we found that these layers were able to get adequate informatin for the later steps. For nonlinear activation, we use *relu* activation function. The output of the fourth convolutional layer is fed into two layers of dense layers which have *relu* activation in between them. We keep the last dense layer with 1000 neurons which is our feature dimension.

In Figure 5.5, we start with a filter size of 20×20 for the first convolutional layer. We keep the size slightly bigger to capture more information from both the time and frequency domain. We have experimented with larger sized filters and find that they result in slower convergence. We use pooling layers along with the conv layers with a stride of 2 and a kernel size of 2×2 . This is a common choice for pooling layer to sub-sample from layers to reduce the size as neighbor nodes are usually highly correlated. We also find that batch normalization boosts the accuracy as it handles different scaling factors of the input.

Using the trained siamese network, we obtain the robust audio representation of the audio sample by forward propagation through any of the twin networks. Since the networks have identical weights, choosing any of them yields the same representation.

5.5 Cross Modal Projection

In the previous section, we proposed a feature learning algorithm that learns to map audio data onto a feature space that projects similar data close and dissimilar data far. This gives us a robust audio representation which by itself can directly be used to train robust audio classifiers. However, that representation can not be used to perform zero-shot learning as it does not borrow knowledge from an external source. To achieve this, we propose an algorithm that integrates semantic knowledge from the text domain to the audio representation. This is done by projecting the audio representation onto the word embedding space such that each audio representation is mapped to the word embedding of its corresponding text.

The difficulty in this step, however, is that a word is spelled or written in exactly one way, whereas, its corresponding audio clip may have a lot of variations. In Sound-Semantics, we solve this problem using a neural network to learn the projection from audio to text.

5.5.1 Neural Network Architecture

We use a two-layer fully connected neural network for cross-modal projection between the two data modality. The first layer has 500 neurons which are connected to the input layer with a dimension of 1000. For the first layer, we apply $\tanh()$ to model the non-linearity in input data. The output of the first layer is then fed into the last layer which outputs the projected embedding of the audio onto the word space. The number of neurons in the output layer is the same as the dimension of the word embedding we use. In this case, it is set to 50 to match with the dimensions of word embedding of [157].

For an audio sample, A and its robust representation, A_{feat} , the final output of the neural network is defined by the following equation:

$$A_{out} = \theta^2 g(\theta^1 A_{feat}) \quad (5.2)$$

Here, θ^l denotes the layer of neurons and l denotes the number of layers. If A_{feat} has $M = 1000$ dimensions, the number of neurons in the first layer is h^1 , and the number of dimensions of word embedding is d , then the layers have the following dimensions: $\theta^1 \epsilon^{h^1 \times M}$ and $\theta^2 \epsilon^{d \times h^1}$. Note that, g denotes the non-linear function $\tanh()$.

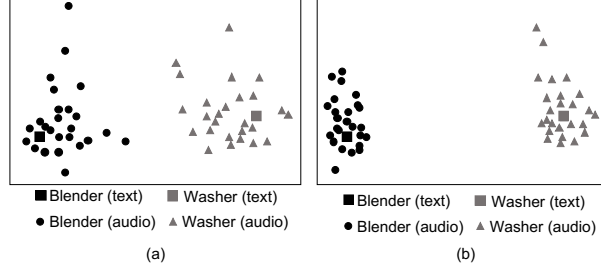


Figure 5.6: (a) MSE Loss fails to separate the cross-modal embedding of adjacent audio class *blender* and *washer* in word embedding space. (b) Neighbour-aware Loss projects the audio clips close to their corresponding labels in word embedding space and also keeps separation from adjacent classes.

5.5.2 Loss Function

The target of cross-modal network is to project audio as close as possible to its corresponding label’s word embedding. Therefore, when an audio sample’s robust representation passes through the neural network, the target is to minimize the euclidean distance between the output of the neural network and the audio’s class label’s word embedding.

- *Mean Square Error (MSE) Loss*: Mean Square Error(MSE) [159] is a good choice of loss function for our network as we can use it to minimize the distance between projected audio embedding and its label. Lets represent the pre-trained word embedding of a word W as $E(W)$. For an audio sample A , and its class label y , following the MSE loss function we want to minimize:

$$L(\Theta) = \|E(y) - A_{out}\| \quad (5.3)$$

Here, $\|.\|$ is the Euclidean norm and $\Theta = \{\theta^1, \theta^2\}$. We have omitted the division with vector size here for easy notation. One drawback of MSE loss is it does not consider its neighbour word labels when minimizing the distance between the audio projection and word embedding. This may lead to poor separability of embeddings between two audio classes which have close class label word embedding. In Figure 5.6(a), we demonstrate this fact by projecting audio of *blender* and *washer* in word space. We see from the figure that, MSE loss minimized the distance between the corresponding projection of audio and their class labels. But MSE loss failed to get a fair separation between the projected embedding of *blender* and *washer* audio clips.

- *Neighbour-Aware Loss*: To consider nearby labels, we propose a new loss function to project audio into word embedding space. The proposed loss function has two objectives. First, it minimizes the distance between the projected audio embedding and its corresponding audio class label. Second,

it ensures that the distance between the projected audio embedding and its corresponding audio class label is less than the distance between the projected audio embedding and other class labels by a certain margin.

The second objective makes sure that the projected audio embedding of a class y is closer to $E(y)$ by a margin of α than other class labels $y' \neq y$. To formally define, we want the following condition to hold after projection:

$$\|E(y) - A_{out}\| + \alpha \leq \|E(y') - A_{out}\|, \quad y' \neq y \quad (5.4)$$

Hence, the objective function becomes:

$$\begin{aligned} & \underset{\Theta}{\text{minimize}} \quad \|E(y) - A_{out}\| \\ & \text{subject to} \quad \|E(y) - A_{out}\| + \alpha \leq \|E(y') - A_{out}\|, \quad y' \neq y \end{aligned}$$

Using this optimization function, we propose our *Neighbour-aware* Loss function as following:

$$\begin{aligned} L(\Theta) = & \|E(y) - A_{out}\| + \frac{1}{|adj|} \times \sum_{y' \in adj(y)} (\|E(y) - A_{out}\| \\ & + \alpha - \|E(y') - A_{out}\|) \end{aligned} \quad (5.5)$$

Here, $adj(y)$ denotes the neighbour or adjacent labels that our proposed loss consider. $|adj|$ denotes the number of classes in the $adj(y)$ set. We experimented to set $adj(y)$ in two ways: a) keeping all other labels other than y in the set, b) keeping the nearest n neighbours of y in terms of euclidean distance in embedding space. We find that the former gives a slight improvement over the later. But the second option yields faster training when the number of classes are high. Both MSE and the proposed neighbor-aware loss are convex functions. However, since the neural network itself is non-convex, $L(\Theta)$ is non-convex.

In Figure 5.6(b), we show the effect of neighbour-aware loss with our previous example of *blender* and *washer*. We see that, the proposed neighbour-aware loss is able to minimize the distance between projected audio embedding and their corresponding class label's embedding in word space and also maintain a margin of distance with audio embedding of neighbour classes. Thus, we get adequate separability between the projected embedding of *blender* and *washer* audio clips as desired.

5.6 Two-Level Classifier

Sound-Semantics employs a two-level classifier to infer the most likely sound type for an input audio clip. The first level classifier determines whether the input audio belongs to *heard* or *unheard* classes. The second level classifier makes the final determination of the most probable sound type for the input audio.

5.6.1 Heard vs. Unheard Determination

Recall that Sound-Semantics is trained on examples from the *heard* classes only and the goal of the cross modal projection step is to create a projection of an input audio which is as close as possible to its corresponding textual label. Hence, the projection of audio clips from an *unheard* class are unlikely to be close to any class labels that are in our training data.

Using this hypothesis, we devise a simple threshold-based decision algorithm to determine whether an input audio clip belongs to an *unheard* class. For an input audio, u , it belongs to an *unheard* class if the following condition is true:

$$\min_{h \in H} \|E(h) - \rho(u)\| > \Pi \quad (5.6)$$

where, $\rho(u)$ is the audio embedding, H is the set of *heard* classes, Π is an empirically determined threshold, and $\|\cdot\|$ is the Euclidean norm. If this condition is false, u belongs to a *heard* class.

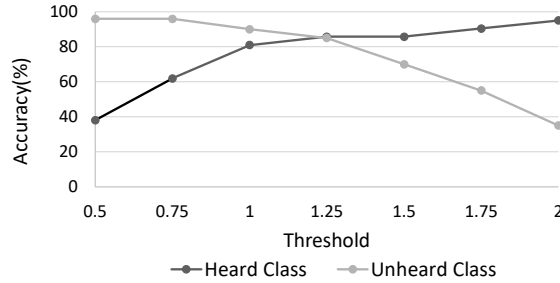


Figure 5.7: The accuracy of *heard* and *unheard* class detection depends on the threshold Π 's value.

The threshold, Π is empirically determined to maximize the accuracy of *heard* vs. *unheard* class detection over an empirical dataset. In Figure 6.14, we plot the accuracy for $\Pi \in [0.5 - 2.0]$. We observe that setting a high threshold fails to detect any class as *unheard* and the accuracy drops to almost zero for *unheard* classes. On the other hand, setting the threshold too small leads to poor results for *heard* classes as it determines every audio sample as *unheard*. Hence, there is a trade-off

between the *heard* and *unheard* class detection accuracy. The optimum threshold is 1.25, for which, the classification accuracy of both the *heard* and *unheard* classes are around 85%.

5.6.2 Classification

Once an audio clip is determined to be from a *heard* or an *unheard* class, we use a nearest neighbor classifier to get the class label. If the audio is detected as *heard* category, we consider labels from only the *heard* categories, and select the label that has the minimum Euclidean distance from the audio’s projected embedding. Likewise, if the audio is detected as an *unheard* type, we consider labels from only the *unheard* categories, and select the label that has the minimum Euclidean distance from the audio’s projected embedding.

5.7 Implementation Notes

- *Mobile Phone Application.* We develop a mobile phone application to run Sound-Semantics. The application records audio and classifies them using the models we have designed. We use Tensorflow Lite [160] to build the application to run our pre-built models. In Table 5.1, we list the run-time for inference of the two models. We see that the siamese network having four convolutional layers takes 112 ms on average for inference, which is about 4 times higher than the inference time for cross modal projection model having only two fully connected layers.

Table 5.1: Runtime of our algorithms

Operation	Execution Time(ms)
Siamese network inference	112
Cross Modal network inference	39

- *Training.* We train our models on a server machine and use the pre-built models in the mobile application. We use mini-batch with Adaptive-Moment-Estimation (Adam) [161] to train the models. We use Adam as it adapts its learning rate for faster convergence than an optimizer with a fixed learning rate algorithm such as Stochastic-Gradient-Descent (SGD).

5.8 Empirical Evaluation

In this section, we empirically evaluate the proposed algorithms with well-known public datasets and report the effectiveness of different parts of Sound-Semantics’ pipeline. Note that our evaluation is primarily focused on Sound-Semantics’ capability in inferring acoustic events that do not have training examples. The data we use in the evaluation have common environmental noise and

reverberation effects, and thus our neural network models and their evaluation implicitly included these artifacts. However, environmental effects vary in different acoustic environments and impacts any audio classification algorithm negatively. To deal with this, additional pre-processing steps such as noise removal, reverberation compensation, and augment training should be applied in conjunction with Sound-Semantics.

5.8.1 Datasets

We use three datasets in the empirical evaluation: ESC-50, ESC-10, and AudioSet [162, 83].

ESC-50 and ESC-10 are two popular datasets for audio event recognition and have been used in several recent audio classification papers [162, 97]. These two datasets have labeled collection of 2000 and 400 environmental audio recordings, respectively. There are a total of 10 and 50 classes with 5 seconds long 40 examples per class for ESC-10 and ESC-50 dataset, respectively. The datasets are provided with five folds for easy benchmarking across all algorithms. The ESC-50 also divides the dataset into categories such as *Animals*, *Natural Soundscapes*, and *Interior*.

AudioSet is primarily used to learn priors for transfer learning [162] to recognize acoustic events on datasets like ESC-10 and ESC-50. AudioSet is weakly labeled and most of the files are multi-labeled. On average, there are five labels per file [83]. This makes AudioSet unsuitable for our system’s evaluation as we focus on single label classification problems. Hence, we select a subset of the classes and files from AudioSet which have at least 50 single-labelled files. This results in a dataset that has over 1600 audio clips from 33 classes.

5.8.2 Performance of Robust Audio Representation

In Section 5.4, we use a contrastive function (Equation 5.1) to learn representation of audio. Contrastive function directly learns to map similar audio closer and dissimilar audio farther in feature space. This gives our proposed Siamese network based robust audio representation advantage over traditional classifier network based feature learning. Here, we compare Siamese based robust audio representation with a traditional classifier network based feature learning, by evaluating the clusters generated by mapping audio samples in feature space. Better representation results in more separability between clusters of different classes and less variance in a cluster of same class. For this comparison, we used ESC-10 data set. We generated feature for each of the five folds using Sound-Semantics’ robust audio representation pipeline and convolutional neural network based classifier. Both of the networks have same number of layers and same parameters for fair comparison.

Moreover, they were trained with same number of epochs.

Table 5.2: Our proposed Robust Audio Representation preserves better clustering properties than feature extracted from a convolutional neural network based classifier.

Algorithm	Intra-cluster Centroid Distance	Inter-cluster Variance
Convolutional Network	0.36	0.029
Sound-Semantics' Robust Audio Representation	0.59	0.017

We compared the features learnt by the two algorithms using two metrics. First, the mean distance between the centroids of the clusters for different classes. Second, the inter-cluster variance among each class. From Table 5.2, we find that the mean distance of the centroids for Sound-Semantics is 0.59, on the other hand the mean distance for convolutional network is 0.36. This shows that features generated by Sound-Semantics is able to differentiate between the classes more efficiently than features generated by convolutional network. Then, we compare the quality of the clusters by comparing that the inter-cluster variance for Sound-Semantics and convolutional network are 0.017 and 0.029, respectively. Therefore, we conclude that, our robust audio representation algorithm learns better representation than traditional convolutional networks.

5.8.3 Performance of Proposed Neighbour-Aware Loss

We compare our proposed neighbour-aware loss with MSE loss by comparing their accuracy in classifying audio events with training examples. We used ESC-10 dataset and ESC-50 dataset. For ESC-10 we did the experiment for all the folds. Moreover, we did the experiments with the categories listed in ESC-50 and reported the mean accuracy with variance as error bar in Figure 5.8. We refer the category based ESC-50 dataset as ESC-50 Categorized. Note that, in a category the class labels are much closer in embedding space than a general ESC-10 dataset. For example, in Animal category some of the classes are : dog, cat, pig. They all are very close to each other in word embedding space which makes Sound-Semantics' cross-modal projection based classification harder. In Figure 5.8, we see that our proposed loss has an accuracy of 83%, whereas MSE loss has an accuracy of 82% for ESC-10 dataset. However, for categorized ESC-50 dataset where the labels are closer to each other in embedding space our proposed neighbour-aware loss has an average accuracy of 76% and MSE has an average accuracy of 72%. So, we see a performance boost of 4% here. As

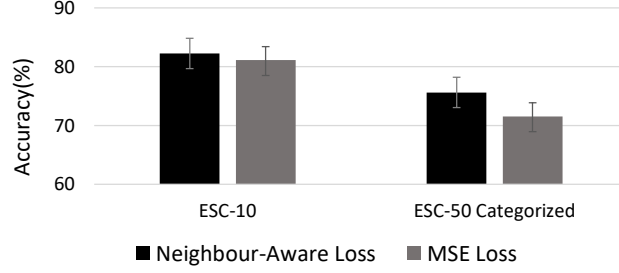


Figure 5.8: Sound-Semantics’ proposed neighbour-aware loss yields better result in classifying heard classes. The effectiveness of neighbour-aware loss reflects more in ESC-50 Categorized dataset. In categorized dataset the class labels are closer in embedding space which makes the scenario harder for MSE loss. Sound-Semantics’ proposed loss function comes as winner because it is aware of the neighbor words.

our proposed loss considers neighbour words when projecting audio to word embedding space, it performed better when the labels in the dataset are closer to each other compared to MSE loss.

5.8.4 Performance on *Unheard Class*

Here, we report Sound-Semantics’ performance on unheard classes. These are the classes that Sound-Semantics did not have any examples during training phase. As, current state of the arts are not able to classify audio data without training examples, here we only report the performance of Sound-Semantics with varying number of unheard classes. To evaluate Sound-Semantics’ performance, we used a subset of ESC-50 and AudioSet as dataset, where we select classes based on the acoustic and semantic similarity.

For evaluation, we consider five scenarios. The scenarios have 2, 3, 4 and 5 unheard classes respectively. Each scenario was evaluated atleast three times with different combinations of classes in both heard and unheard classes. For ESC 50 dataset, the unheard classes were selected from each category by making sure that the category has some classes in the heard category. For AudioSet we picked the unheard classes by selecting from the closest labels of the classes in the heard category.

In Figure 6.12, we show the accuracy of Sound-Semantics for different number of unheard classes. We observe that for two unheard classes, the mean accuracy of Sound-Semantics is around 90% for the ESC dataset and 76% for the Audioset. The drop of accuracy for Audioset is due to weak labeling and presence of noise in the audio clips. For both datasets, the accuracy drops as the number of unheard classes increases. The reason is that the accuracy of unheard sound detection is dependent upon the heard classes. An unheard sound is detected successfully only when there are heard sounds which are similar in acoustic properties as well as their labels’ contextual properties

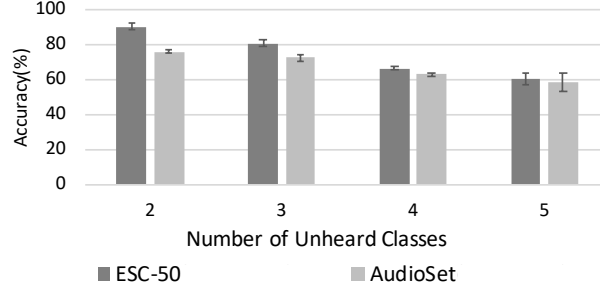


Figure 5.9: Sound-Semantics’ accuracy in unheard audio event classification decreases from 90% to 60% for 2 to 5 unheard classes in the ESC dataset. For Audioset, due to weak labeling and noise, the accuracy drops to 76% for 2 unheard classes.

in the text domain. The heard classes act as representatives of the corresponding unheard classes. As more classes fall into the unheard category, their representative classes in the heard category decreases. Therefore, we see that the accuracy drops to 60% when five classes are in the unheard category.

5.8.5 Performance on *Heard* Class

Sound-Semantics is a generalized acoustic event recognition system that is expected to encounter examples of both *heard* and *unheard* classes at runtime. To test Sound-Semantics’ accuracy on *heard* classes, we compare its performance with baseline algorithms such as Convolutional Neural Network (CNN) and Random Forest. Both of them are trained with same number of epochs. We use more epochs when using only CNN as it does not have the extra step of cross-modal projection. Note that all of these algorithms have the same number of layers and same design of network for fair comparison. We use the ESC-10 dataset in this experiment and perform a five-fold cross validation using the folds that the dataset comes with. We also report the performance of the best classifiers for the dataset, i.e., EnvNet [163] and ConvRBM [68].

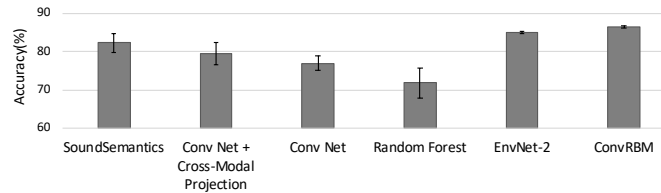


Figure 5.10: Sound-Semantics is able to classify heard audio events better than baseline algorithms because of its robust audio representation and cross-modal projection.

In Figure 6.13, we see that the mean accuracy of Sound-Semantics for heard classes is 82.25% which is higher than that of a CNN with our cross-modal semantic projection (79.4%), a regular CNN (77%), and random forest(71.8%). EnvNet and ConvRBM achieves 84.9% and 86.5% accuracy, respectively. Note that both algorithms use raw audio as the input to the network unlike Sound-Semantics which feeds time-frequency domain representation as the input to the network. These papers [163, 68] learn the frequency filterbanks from raw audio data, and thus learns better representation and eventually attains a higher accuracy. Sound-Semantics’ Siamese network-based robust representation and cross-modal projection steps are generic and can be used with the techniques proposed in these papers to achieve similar or higher accuracy in heard class detection.

5.8.6 Performance with *Distractor Words*

While Sound-Semantics performs well in detecting unheard audio classes it requires the unheard label names beforehand. In this experiment, we wanted to see Sound-Semantics’ performance when the unheard labels are not only limited to the expected classes, but has some extra random labels also. We refer these extra words in the unheard labels as *distractor words*. We performed two sets of experiments for a scenario with two unheard classes. In the first, experiment the distractor words were chosen randomly and in the second experiment the distractor words were chosen from the k-nearest neighbours of the expected class label in word embedding space. For example, if we had *cat* sounds in unheard class then some of the nearest neighbour words were: *dog*, *rabbit*, *pet*, *mouse* etc. Note that, both of these experiments were done with multiple pairs of unheard class and only the average is reported here.

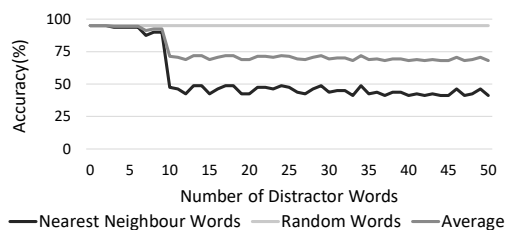


Figure 5.11: Random word injection in unheard class labels does not have effect on Sound-Semantics’ performance in unheard class detection. Adding closest words with respect to unheard labels in embedding space causes a certain drop in Sound-Semantics’ accuracy.

In Figure 5.11, we see the result with random words and nearest neighbour words as distractor words. With random words, the accuracy does not change even with 50 distractor words. This proves that the words have well spanned embedding and Sound-Semantics is robust with random

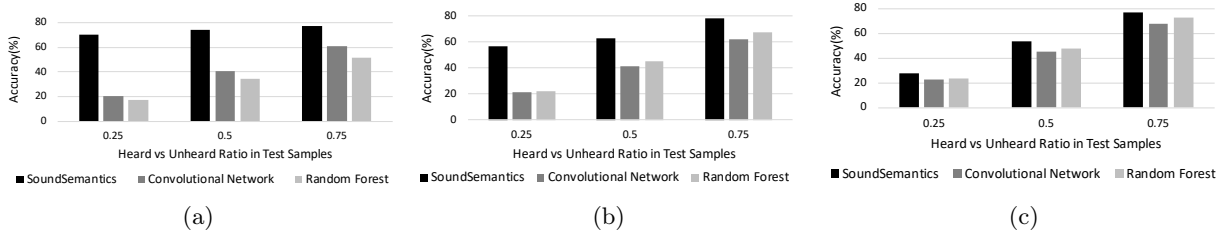


Figure 5.12: Sound-Semantics achieves better accuracy than baseline algorithms in monitoring kitchen for all scenarios with all proportion of heard vs unheard class in test samples. (a) Eight out of 10 class are in heard category. Sound-Semantics’ accuracy remains over 70% for all cases. (b) For six out of 10 class in heard category, Sound-Semantics’ accuracy drops from 75% to 58% with increasing proportion of test samples coming from unheard classes, whereas the best baseline’s accuracy drops to 23% . (c) For, two out of 10 class in heard category, Sound-Semantics achieves 77% accuracy when majority of test samples are from heard category but drops to 28% with more test samples from unheard category.

Table 5.3: Dataset for Kitchen Monitoring.

Category	Examples	Count	Length
Appliances	dryer, utensil, basin, pouring, plate, blender, vacuuming, heater	320	1600 s
Pet	dog, cat	80	400 s

words injection. However, with nearest words added the accuracy drops around 10 distractor words. This is because, the distractor words are too close to the unheard class labels in the embedding space. We notice that from 10 to 50 distractor words, the accuracy does not dip very much and the reason behind this is further inclusion of nearest distractor words do not have any effect as they are far enough from the unheard class labels in embedding space.

5.9 Real world evaluation

We deploy Sound-Semantics phone application in two real world scenarios involving both indoor and outdoor usages and evaluated our system’s performance with other state of the art acoustic event classification algorithms. In this section we report the results from these two real world applications.

5.9.1 Kitchen Monitoring

We use Sound-Semantics in an indoor scenario for kitchen monitoring. In this application, a user monitors his/her kitchen using Sound-Semantics. The user provides a total of 10 acoustic event names which are listed in Table 5.3. The number of samples and total duration of audio are also listed in the table.

We consider three different training scenarios to stress out Sound-Semantics’ performance under different conditions. First, we consider that the user is giving 8 out of 10 classes’ audio examples

to Sound-Semantics during training. So the number of classes in heard and unheard categories are 8 and 2, respectively. Second, we consider where 6 out of 10 classes' audio examples are given to Sound-Semantics during training. And, the last scenario is the hardest, where Sound-Semantics has only 2 classes' audio samples during training, that makes 8 out 10 classes in unheard category. For each scenarios, we randomly picked the classes for heard and unheard categories but making sure that classes from both *Appliances* and *Pet* have been selected. In Figure 5.12, we report the performance of Sound-Semantics along with two baseline algorithms : convolutional neural network(CNN) and random forest for all three aforementioned scenarios. For each scenario, we consider three cases varying the ratio between samples from heard and unheard class in test dataset in the following ways: a) $\frac{\#heard}{\#unheard} = .25$, b) $\frac{\#heard}{\#unheard} = .5$ and c) $\frac{\#heard}{\#unheard} = .75$. Here, # represents the number of samples.

In the first scenario (Figure 5.12(a)) where only 2 classes were in unheard category, we see that Sound-Semantics' accuracy is over 70% for all cases as there are enough representative classes in training samples. For the 1st case where only .25 fraction of test samples are from heard category baseline algorithms fall behind by a long margin (less than 20% accuracy) as they can not classify any of the classes from unheard category. The situation gets better for the baselines as the fraction of heard classes increase in test samples. The maximum accuracy of convolutional network approaches around 60% for the case with .75 fraction of test classes in heard category. For this case, Sound-Semantics' accuracy(77%) still beats baseline algorithms by more than 15%. For the second scenario (Figure 5.12(b)), with 4 classes in unheard category the scenario becomes harder as unheard category classes have increased. Still, Sound-Semantics was able to classify around 57% of the test samples even in the extreme case with only quarter fraction of test samples from heard category. In this case, the baseline algorithms accuracy are around 22% and 23%. The third scenario (Figure 5.12(c)), is the most challenging one as it deals with only two classes in the heard category. When the fraction of test samples which belongs to heard category is .25 we see that Sound-Semantics' performance drops to 28% as it can not classify most of the unheard classes with such limited training data. Although it still outperforms conv-net(23%) and random forest(24%). With less proportion of test samples from unheard category, Sound-Semantics' performance gets better as expected and reaches 77% when there are three-quarter of the test samples from heard category.

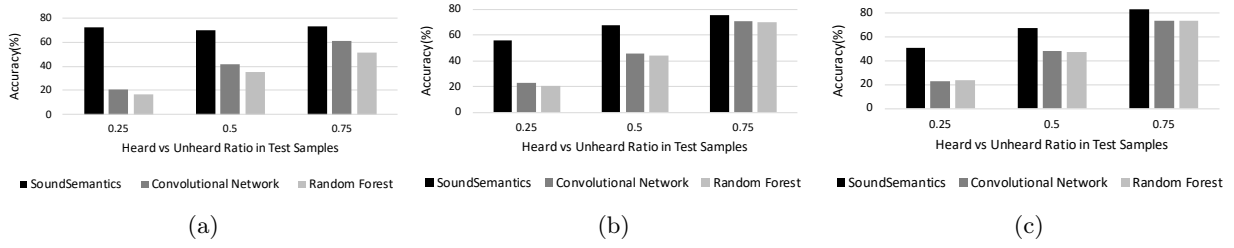


Figure 5.13: Sound-Semantics performs better than baseline algorithms for all cases in street monitoring. (a) When four out of six class are in heard category, Sound-Semantics has more than 70% accuracy for all cases. On the other hand, baseline algorithms’ accuracy drops below 20% for some cases. (b) For three out of six class in heard category Sound-Semantics outperforms all baseline algorithms for different cases. (c) When two out of six class are in heard class, for .25 fraction of test samples coming from heard category Sound-Semantics’ performance drops to 51% which is still two times more than baselines’ performance.

5.9.2 Street Monitoring

Sound-Semantics was also deployed in outdoor scenario for street event recognition. Here, we showed the usecase of Sound-Semantics in sreet event recognition using audio such as: car engine sound, honk or people talking etc. There are total 6 classes in this scenario and in Table 5.4, we described the data for this application.

Table 5.4: Dataset for Street Monitoring

Category	Examples	Count	Length
Vehicle	car, airplane, honk, siren	160	800 s
Noise	talking, shouting	80	400 s

For evaluation in outdoor environment, we again consider three scenarios. Scenario 1: out of 6 classes, 4 classes were in heard category and the rest of the 2 classes are in unheard category. Scenario 2: both heard and unheard categories have 3 classes each. Scenario 3: heard category has only 2 classes whereas unheard category has 4 classes. For each scenario we again vary the proportion of heard vs unheard classes to .25, .50 and .75 respectively.

In Figure 5.13, we report the result of Sound-Semantics for all the cases in the three scenarios we mentioned. In Scenario 1 (Figure 5.13(a)), where only 2 classes are in unheard category, we find that Sound-Semantics has accuracy of over 70% for all the cases. On the other hand, the baseline algorithms’ accuracy drops under 20% when most of the samples are coming from unheard category. In scenario 2 (Figure 5.13(b)), for Sound-Semantics we see an accuracy of 75% for case 3 with majority of the samples in test cases coming from heard classes. However, when the ratio of heard classes in test data get decreased in case 1 and 2, the accuracy drops. Yet, Sound-Semantics’

performance is better than baselines by a margin of greater than 30%. For scenario 3 (Figure 5.13(c)), where only two classes are in heard category, the accuracy for the case where 75% of test samples are from heard classes reaches up to 83% for Sound-Semantics. For heard category the classification has become a binary problem, and thus causes boost in accuracy for this case. However, for case 3, the accuracy drops and gets to 51% where 75% of the test samples are from unheard categories. For this case, baselines achieve a maximum accuracy of 23%.

5.10 Related Work

Audio classification has been a well known problem for decades. There are some approaches to deal with limited training data for audio classification. Some of the past works dealt with limited training data problem by generating new training examples with the help of perturbation acoustic characteristics of existing examples, e.g., frequency warping [2], modifying tempo [164], and adding simulated reverberation [145], and noise [165]. These works fall in the category of *data augmentation*. Another direction of works use classifier fusion [166], where outputs of multiple classifiers are combined to improve the overall accuracy. These techniques vary from simple voting and averaging [167], to ranking [168], to learning decision templates [168]. Very recent works [162] deal with the problem of limited data with transfer learning [169] from other large scale audio dataset. Another branch of work [170] proposes classification algorithm for weekly-supervised dataset where the presence or absence of an audio event in a clip is marked but the absolute position of the audio event is not labelled. However, our problem is significantly different from these works as these papers do not consider the scenario when absolutely no training examples are present. Ours' is the first work to deal with zero shot learning for audio data by leveraging semantic knowledge from textual domain.

In image domain, learning from limited or no data has been explored recently. [123, 171] focus on learning suitable image representation to classify images from very few or only one examples. These works are totally different from ours as we want to recognize audio events without any training data. The other branch of learning with limited data focuses on zero shot learning for image classification. The earlier practices of zero shot learning [149, 172] for image classification problem infer the labels of unseen classes using a two step algorithm. First, the attributes of the sample is inferred and then the class label is predicted from an attribute database, which has most similar attributes with the image. Recent works [173, 174] have explored the mapping between image features and semantic

space by projecting image features into word embedding space. Similar line of works [175, 150, 176] project image feature into semantic space then searches the nearest class label embedding using ranking loss. Recent works such as [177] uses auto-encoder to generate embedding for image features by reconstructing the images from the projected embedding. Although these papers propose zero shot learning method for images, none of them addresses the problem for audio domain and the associated challenges with it such as audio representation learning. Our work is the first system to propose a zero shot learning method for audio classification where we overcome the challenges for cross-modal learning between text and acoustic domain. To the best of our knowledge, we present the first system with capabilities of inferring acoustic events without training examples and report extensive evaluation of such system under varying scenarios.

5.11 Discussion

- *Zero-Shot Learning Assumption.* We assume that audio classes with similar acoustic properties have semantically similar class labels. We use this assumption to learn the non-linear mapping function to project audio signals onto the word embedding space. When there are no semantically similar training examples corresponding to an unheard class, a zero-shot learner fails to recognize examples from that unheard class. However, in a large dataset, the chances that there is no semantically similar training example to an unheard class is relatively low. This is why zero-shot learning performs better on larger and diverse datasets than smaller and skewed ones.

- *Necessity of User-Provided Tag-List.* The user-provided additional labels do not have any influence on the training phase. They are only used in the classification step after the training has been completed. If we do not have these additional labels, then the search space becomes too large (i.e., as big as having all the words in our database for a language) in the classification stage. We have demonstrated this in Section 5.8.6 where we added extra labels (i.e., distractor words) along with user-provided additional labels. We find that the inclusion of similar semantic words with additional labels causes a dip in the accuracy of unheard audio class detection.

- *Heard-Unheard Synergy.* Sound-Semantics' accuracy on unheard sound classification depends on the characteristics of heard classes. If the training classes have representative acoustic and semantic representations of rare audio events, Sound-Semantics is able to infer it. Hence, as long as this assumption is satisfied, Sound-Semantics can recognize rare, inconvenient, or unwanted events. For example, to detect the sound class *waves* that has no training examples, we need to have some class

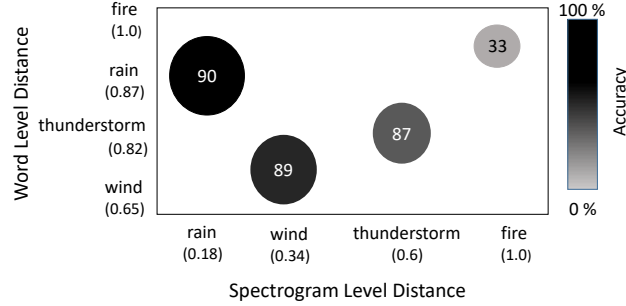


Figure 5.14: Sound-Semantics’ performance in recognition of an unheard audio event is contingent upon the acoustic and word embedding similarity between that event and classes in heard category.

in the heard category that is close to waves in terms of acoustic properties as well as word semantics, e.g., *wind*. However, without the presence of such classes in the heard category, the performance of *waves* detection without examples drops. To visualize this, we conduct an experiment where we want to classify the unheard sounds of *waves*. For training, we use one class from the following set interchangeably in the heard category: $\{rain, wind, thunderstorm, fire\}$. We want to see the effect of these classes for *waves*’ zero-shot classification. We use the mean Euclidean distance between the MFCC features of the audio samples of the classes as a metric for the *Spectrogram-level distance* and the Euclidean distance between word embeddings as *Word-level distance*. We sort the classes $\{rain, wind, thunderstorm, fire\}$ in terms of both spectrogram-level distance and word embedding-level distance from *waves*. In Figure 6.16, we show the sorted classes on the X and the Y axes based on the spectrogram-level and word embedding-level distances. All distances are normalized by their maximum. We see that, as the spectrogram level distance increases, the classification accuracy for *waves* drops. We find that, for *fire*, the accuracy drops to 33% as it is very distant from *waves* in both the spectrogram and the word level.

5.12 Summary

This chapter presents the first system for acoustic event classification without training data. We propose a novel context-aware audio representation embedded with semantic knowledge from textual domain. Using this novel feature mapping, we created a classification algorithm capable of inferring acoustic events with or without training examples. We evaluated our proposed system Sound-Semantics with public audio dataset and presented the benchmarking for different parts of the whole pipeline in the chapter. We also implemented a mobile application and deploy our system in two real world applications. We compared our system’s performance with other state of the art









audio classification algorithms and showed that our system is capable of inferring acoustic events without any examples with an accuracy of 60% – 90% for different scenarios.

CHAPTER 6

WiFi-Based Activity Recognition Through External Knowledge Transfer

6.1 Introduction

The widespread adoption of WiFi in indoor spaces, and the accessibility of WiFi signal characteristics such as the signal strength (RSSI) and channel state information (CSI) in commodity WiFi chipsets, make WiFi an attractive technology for human activity monitoring. Alternate solutions that use wearables like smartwatches and activity trackers are becoming less attractive due to their usage adherence issue, and systems that use cameras to monitor home activities raise serious privacy concerns. WiFi sensing, on the other hand, is device-free, non-intrusive, and less privacy-invasive. Hence, in recent years, we have seen an increase in WiFi-based sensing and inference systems whose feasibility has been demonstrated in application scenarios such as home activity monitoring [16], sleep monitoring [17], gesture-based smart device interaction [18, 19], and health vitals tracking [20].

	Training Phase		Testing(target) Phase	
Existing Systems	 walk	 drink	✓  drink	✗  run
Wi-Fringe	 walk	 drink	✓  drink	✓  run

✓ : able to detect ✗ : not able to detect

Figure 6.1: Unlike existing systems, WiFringe is able to recognize *run* in the testing phase, even though it did not see any training example of *run* in the training phase.

A vast majority of WiFi-based activity recognition systems employ either template matching algorithms or machine learning classifiers — ranging from traditional classifiers like support vector machines [22] to advanced deep convolutional neural networks [23]. These algorithms require a decent number of training examples for each class of activity in order for the system to accurately classify them. Furthermore, the capability of these systems are fundamentally limited by the number of activity classes for which the system has been trained for. When these systems are presented with a completely new type of activity, there is no built-in mechanism to make an educated guess about

the possible class label for that unseen example.

Figure 6.1 illustrates this scenario. When a system is trained to recognize only {walk, drink}, but is presented with an example of an unseen activity, e.g., run, it is likely to detect the activity as either walk (based on the closest match) or it will determine that it is an unknown category (based on a distance threshold). At present, existing systems have no inherent mechanism to infer that the activity could be run, as these systems have no prior knowledge of how an activity called run might be.

A naïve way to deal with the above problem is to train a system with examples of all possible activities that it may ever encounter. However, this is not feasible for several practical reasons. First, despite recent efforts in environment-invariant activity classification [109], existing WiFi-based sensing systems are prone to environment changes due to RF signals’ dependency on surroundings for reflection, refraction and scattering. Hence, it puts the burden on the end-users to provide the training data — which is time-consuming, error-prone, and in general, an inconvenience. Second, even with environment-invariant techniques, due to the diversity of human activities, it is not feasible to build models for all possible activities for users of all demographics. This motivates us to devise a WiFi-based sensing system that recognizes activities and gestures without prior examples.

In this chapter, we propose the first system, called the *WiFringe*, which can infer activities from WiFi data without requiring prior training examples for all of its activity classes. The principle behind WiFringe is popularly known as the *zero-shot learning* [149, 150], which is an active research topic in computer vision and image classification fields. Recently, zero-shot learning has also been applied to general-purpose sound recognition problem [74]. These techniques are not directly applicable to RF-based gesture recognition problems, since gestures require tracking sequential properties of the signal and external knowledge about the attributes that defines an activity. To the best of our knowledge, we are the first to apply zero-shot learning in RF-based device-free activity recognition problem. The core idea of zero-shot learning is to exploit information or learned knowledge from other sources such as textual descriptions, rules, and logic. For example, to teach the concept of run to a system that has already learned to recognize walk, instead of training it with many examples of run, we can add a rule into the system, e.g., “run is just like walk but it’s 3 to 5 times faster.” At runtime, the system will use this additional information to classify a run activity correctly.

In WiFringe, to embed such rules between *seen* classes (i.e., explicitly trained) and *unseen*

classes (i.e., not explicitly trained) in an RF sensing system, we exploit attributes and context-aware representation of English words as the additional source of knowledge. Through a RF-domain to text-domain projection algorithm, we blur the difference between an activity’s RF signature and its corresponding English word/phrase by representing them in the same vector spaces, i.e., *word embedding* [26] and *word attribute* [31] spaces. We exploit the attributes and semantic relationship between English words as the background knowledge to classify an activity from WiFi that the system has never seen before. We combine these algorithmic steps to develop an activity recognition system that we name—WiFringe, as it can detect previously unseen *fringe* activities beyond what it is trained for.

The intuition behind WiFringe is that the WiFi signature of an activity correlates with the corresponding verb’s semantic and attribute information. Like two similar activities perturb the WiFi signals similarly, when we describe these two activities in English sentences, we see a similar likeness between the sentences. We generalize this notion for an arbitrary number of activities represented in both RF and text domains, and strive to find a projection between the two representations from the RF domain to the text domain. By learning this projection, we gain the ability to find the corresponding English word from the RF representation of any arbitrary activity.

Projecting RF signals onto the space of textual representation is non-trivial and poses several challenges that are addressed in this chapter. First, we propose context-aware RF features by explicitly learning the transition of *states* (i.e., micro-activities) in an activity . We show that such a representation is robust and yields better features for activity recognition in general. Second, we propose a neural network architecture to merge text- and RF-domain representations of activities so that WiFi CSI data are mapped to the attributes and distributional characteristics of English words. This results in the first *cross-modal RF embedding* work, and paves the way for device-free WiFi-based activity classification without requiring training data for all activities. Third, we propose a two-level classifier that is capable of classifying both *seen* and *unseen* activity types. This makes WiFringe a generalized system for classifying a wide variety of human activities.

We develop WiFringe using Intel Network Interface Card (NIC) 5300 [25] which captures the WiFi CSI data. To develop the machine learning models, we collect training and testing data from four volunteers in a multi-person apartment as well as in an office building for 20 activity classes — which is, to the best of our knowledge, the largest collection of activities used in any WiFi-based

device-free gesture recognition system till date. We show that WiFringe is able to recognize activities with 62%-90% accuracy, as we vary the number of unseen classes from six to two.

The contributions of the chapter are the following:

- We describe WiFringe, which is the first system that addresses the problem of learning without examples for WiFi based device-free named activity recognition.
- We propose a new approach to learn state-aware RF representation to preserve the transitional characteristics of states in an activity. This representation by itself improves the classification performance in supervised learning.
- We propose the first algorithm that exploits semantic knowledge and attributes in text to enable classification of activities from WiFi data without prior training examples.
- We collect data from four volunteers, from two environments, for 20 activities, and show that WiFringe beats state-of-the-art WiFi-based activity recognition algorithms by 30%.

6.2 Example Usage

WiFringe relieves the developers and end-users from the burden of extensive data collection and training and is able to classify new types of activities for which no training examples are provided.

There are many WiFi-based activity recognition systems such as elderly monitoring [178], home activity recognition [16], gesture based media controller [18], and gesture based gaming [179] where the use WiFringe can significantly improve the performance as well as the intelligence of an application. We describe one such scenario to illustrate the capability and usage of WiFringe.

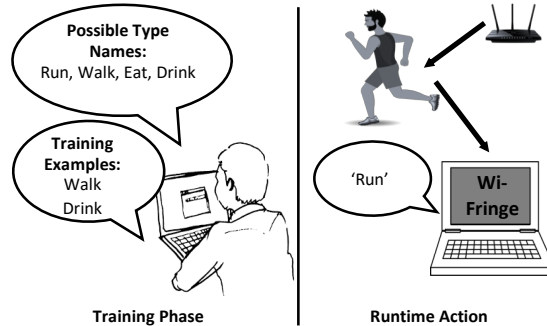


Figure 6.2: WiFringe is able to recognize activities for which it has not been explicitly trained for.

Consider a simplified home activity recognition scenario which is shown in Figure 6.2. The user of WiFringe wants to monitor four activities: {run, walk, eat, drink}. At first, they input this list of

four activities (only the names) to WiFringe. WiFringe then asks the user to provide zero or more training examples for each activity. However, our user provides training examples for two out of the four classes: {walk, drink} and does not provide any example for the remaining two classes: {run, eat}.

Despite the lack of training examples, WiFringe is able to recognize the run activity at runtime by combining its knowledge of walk (from training) with its knowledge of the relationship between run and walk (which it acquired from external sources such as textual embedding).

6.3 WiFringe System Design

WiFringe takes a short-duration WiFi CSI stream (e.g., 5–8 seconds) and a list of possible activity types (i.e., a list of tags) as the input, and processes the CSI stream through a signal processing pipeline to classify it as one of those given activity types. The design of WiFringe is modular. Computationally expensive modules such as the one-time offline training of the classifiers are run on a server, while the end-to-end activity classification pipeline—from sensing to classification—is runnable on embedded systems such as smartphones and tablets¹.

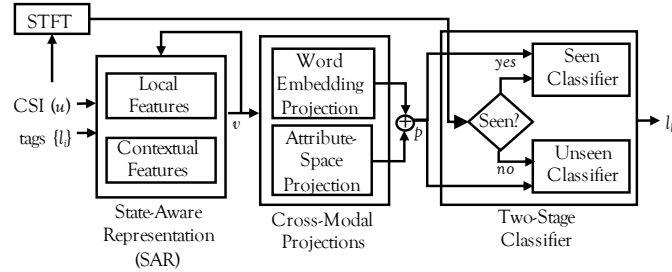


Figure 6.3: WiFringe signal processing pipeline.

Figure 6.3 shows the signal processing pipeline of WiFringe, which consists of three main steps: *State-Aware Representation*, *Cross-Modal Projections*, and *Two-Stage Classifier*. The State-Aware Representation step extracts local and contextual features from the CSI stream. These features are projected onto the word and attribute spaces to incorporate external knowledge from the text domain in the Cross-Modal Projections step. The two-stage classifier determines if the input belongs to a seen or an unseen class and then classifies it accordingly.

¹Recent developments [180] have shown how to extract CSI on smartphones. In this chapter, we conduct experiments using an Intel NUC [124].

There are two classes of activities that WiFringe may encounter at runtime: *seen* and *unseen* classes. The *seen* class refers to those activity types for which WiFringe has labeled CSI streams for training. The *unseen* class, on the other hand, refers to activity types for which WiFringe does not have any training CSI stream. For example, in the home activity monitoring scenario of Figure 6.2, {walk, drink} are seen classes and {run, eat} are unseen classes.

The next three sections describe the algorithmic details of these three components of WiFringe.

6.4 State-Aware Representation

The goal of this step is to obtain a state-aware representation of WiFi CSI values corresponding to an activity which encodes both the *local* features as well as the *contextual* features of an activity. The local features refer to the frequency response of an activity at a particular time-step. In contrast, the contextual features learn the contextual relationship among the local features. At the end of this step, we obtain an information-rich, lower-dimension vector representation of a CSI stream that captures both local and contextual features of an activity. This state-aware feature is generic, and can be used directly with an off-the-shelf supervised classifier to recognize *seen* activities². In WiFringe, we use this representation as an intermediate output, which is used by the later stages of the processing pipeline to enable recognition of both *seen* and *unseen* activities.

6.4.1 The Need for State-Aware Representation

The CSI stream is a time series of complex numbers, sampled at a rate of between 200Hz [119] to 2.5KHz [120]. The raw CSI data are not practical for direct use in the training or in the classification steps, as with high sampling frequency, the dimension of the input vector becomes too large to process efficiently. Moreover, raw CSI signals do not explicitly reflect the intrinsic characteristics of the high-level activity, and generally, they contain artifacts of environmental factors that are not related to the target activity. Hence, like most sensing and inference systems, we map raw data to a lower dimensional vector, commonly known as the *feature vector*, based on the time and frequency domain properties of the signal.

A wide variety of features have been proposed in the literature for activity recognition from WiFi CSI data. The list includes classic time and frequency domain features such as cepstral coefficients

²In Section 6.8.2, we conduct an experiment to demonstrate its performance.

and spectral energy [16], as well as recent practices where a layer of a trained deep neural network is considered as the learned features [23]. However, none of these techniques consider the sequential nature of an activity when converting raw CSI values to feature vectors.

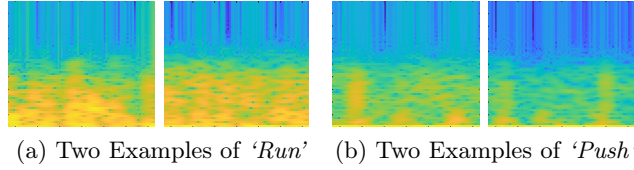


Figure 6.4: Different activities have different state sequences.

Human gestures and activities are time-series of micro-activities that we call *states*. For example, in Figures 6.4, we show four spectrograms of CSI values where the first two correspond to two examples of the run activity, and the last two corresponds to two examples of the push activity. We observe that the pattern how the frequency response changes over the duration of an activity is similar (if not quite the same) for the same activity, and dissimilar for the two different activities. Hence, the key to recognize an activity is to model the sequential change in its frequency spectrum.

To capture the sequential properties of gestures and activities, [120] uses a *Hidden Markov Model* (HMM) [181] where traditional time-frequency features (*Discrete Wavelet Transform* (DWT) [120]) are used to represent the states rather than learning representation that embeds the sequential properties of states. A fundamental limitation of this and any other classical HMM-based approaches is that the Markovian assumption, i.e., a state depends only on the previous state, does not hold for most activities. These models do not work when the number of states is large and variable, and an activity has longer term dependencies between the micro-activities that constitute it.

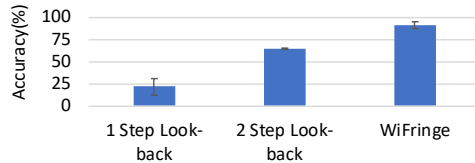


Figure 6.5: Looking back deeper in time improves modeling accuracy.

To demonstrate this, we conduct an experiment to predict the next state of an activity based on the previous states. We use five activities {push, pull, run, walk, eat} and divide each sample into five segments that represent the states that constitute an activity. To test the Markovian assumption about the longer term dependencies among the micro-activities, we consider two variants: (a) *One*

step Look-Back: a five-state Markov model resulting in a 5×5 dimension transition matrix, A , where each entry, A_{ij} represents the transitional probability of going from state i to state j , and (b) *Two Step Look-back*: a 25-state Markov model resulting in a 25×25 dimension transition matrix, A , where each entry, A_{ijk} represents the transitional probability of going from state ij to state k . In Figure 6.5, we observe that using the one step look-back, we get an average accuracy of 20% for predicting the next state. We observe an improvement of prediction accuracy (63%) when we use the two step look-back variant, which conditions on previous two states to predict the next state. This tells us that a strong Markovian assumption that the current state only depends on previous state does not hold strongly for activity's sequence of states and looking back deeper in time improves the modeling accuracy.

To overcome the limitations of existing activity modeling techniques, we propose *state-aware* feature representation of CSI streams that captures complex, non-linear dependencies between micro-actions that constitute an action—without requiring a strict Markovian assumption or a predefined, fixed set of states. To achieve this, we employ a state-of-the-art deep neural network that capture both the local (static) as well as contextual (sequential) properties of CSI spectrograms. Our proposed state-aware model is able to predict the next state of an activity with 90% accuracy (Figure 6.5), which is 30% – 70% higher than Markovian models.

6.4.2 Rationale Behind Deep Neural Networks

In WiFringe, a *Convolutional Neural Network* (CNN) [85] and a *Recurrent Neural Network* (RNN) [85] are used in tandem to capture the local and the contextual features, respectively.

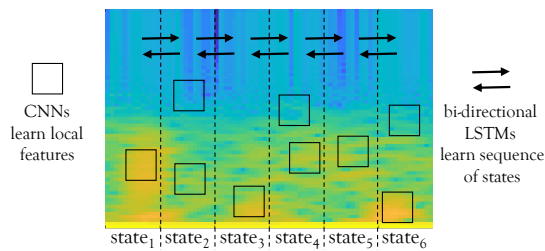


Figure 6.6: CNNs learn local patterns that characterize each state, whereas RNNs (LSTMs) learn sequence of states.

The CSI spectrogram exhibits rich, informative, and distinguishable patterns for different states within an activity. It contains temporal information, for which, a CNN is the most suitable choice [85]. CNNs contain a hierarchy of filters, where each filter's job is to detect the presence of a particular

pattern in a small region on the spectrogram. In Figure 6.6, we use rectangular boxes on the spectrogram which are recognized by the different convolutional filters of the CNN shown on the left.

To model the sequential variation of states within an activity, we use an RNN. We choose an RNN over classical approaches such as a Hidden Markov Model (HMM) since unlike HMMs, RNNs do not require a strong Markovian assumption [182]. With RNNs, a model can learn long-term dependencies among the states. Furthermore, neural networks are in general better at learning complex patterns within the data than shallow models, and RNNs do not need fixed states like an HMM does. In Figure 6.6, we use slices of a spectrogram to denote states and use arrows to illustrate the inherent dependencies between nearby states.

To construct the RNN, we choose *Long Short Term Memory* (LSTM) as the recurrent component since LSTMs are better at learning long-term dependencies between states [85], which makes them suitable for capturing relationship between the past states with the recent states. To preserve contextual information from both the future and the past states, we use a bi-directional LSTM model. A unidirectional forward LSTM only captures the state transition from past to recent states. But a bi-directional LSTM learns how future states influence past states as well. In many cases, it is important to bring information from the last few states to model an activity and bi-directional LSTMs have this feature.

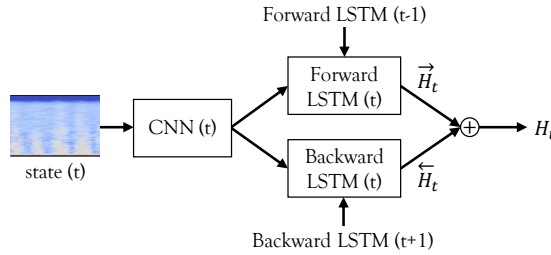


Figure 6.7: Network architecture for state-aware representation.

6.4.3 Detailed Algorithmic Steps

Figure 6.7 shows the integrated neural network architecture that takes CSI spectrogram as the input and produces the state-aware vector representation through a sequence of processing steps.

- *CSI Processing:* For a given CSI stream, X that corresponds to an instance of an activity, we divide the CSI values into n equal segments $\{X_1, X_2, \dots, X_n\}$, which are the states. Here, n is empirically determined, we used $n = 5$, that corresponds to one seconds of CSI data, in all our

experiments. For each segment X_i , we take the spectrogram [121] to obtain S_i as follows:

$$S_i = \text{STFT}(X_i)$$

where, $\text{STFT}(\cdot)$ denotes *Short-Time Fourier Transform* [121], which estimates the short-term, time-localized frequency content of X_i .

- *CNN Processing:* We use a three-layer CNN, G_θ , where θ are the parameters of the network, which takes S_i as the input and produces a 1000 dimension vector, L_i that represents the local features of the input spectrogram:

$$L_i = G_\theta(S_i)$$

Each layer of the CNN extracts a feature map from the input data. As we go deeper into network, the deeper-layer filters extract more information. We empirically determine that three layers of convolutional operations extract adequate information for the later steps of WiFringe. The filters have a size of 3×3 and we increase the depth from 16 to 64 channels for the three layers. We use *Rectified Linear Units* (ReLU), $\sigma(a) = \max\{a, 0\}$ as the non-linear activation function for their robustness against the vanishing gradient problem [183]. The output of the third convolutional layer is fed into two *dense* layers [85] with a ReLU activation layer in-between them. We keep 1000 neurons in the last dense layer, which is the dimension of the local feature vector.

- *RNN Processing:* For a total of n segments, we obtain n local feature maps $G_\theta(S_i)$ from the CNN. Each state's local feature is fed into a bi-directional LSTM (bi-LSTM) to model the contextual property of the states of an activity. The bi-LSTM network has two unidirectional LSTMs, i.e., a *forward* and a *backward* LSTM. For the forward LSTM, each hidden state \vec{H}_i depends on the previous state H_{i-1} and the input S_i . On the other hand, for the backward LSTM, each hidden state \overleftarrow{H}_i depends on the future state H_{i+1} and the input S_i . By using a bi-directional LSTM, we obtain a representation of an activity that is aware of the transition of states which yields a strong model of the activities. We use 500 neurons in both the forward and the backward LSTMs. Each LSTM has a dropout layer to avoid overfitting.

- *Representation:* The final hidden representation of the bidirectional LSTM is the concatenation of \vec{H}_i and \overleftarrow{H}_i .

6.5 Cross-modal Projections

In the previous section, we proposed a feature learning algorithm that learns to map CSI stream segments onto a feature space that captures the contextual information of sequential states of each activity. This gives us a robust activity representation, which by itself can directly be used to train robust activity classifiers. However, that representation can not be used to perform zero-shot learning as it does not borrow knowledge from an external source. In this section, we describe the cross-modal projection step of WiFringe which brings external knowledge from the text domain to enable classification of unseen activities.

6.5.1 The Need for Cross-Modal Projections

Existing CSI-based human activity recognition systems can recognize only a fixed set of activities for which the system has been trained for. When an unseen activity is presented to these systems, the best they can do is to find the class that is closest to the given activity. Due to their reliance on a single source of knowledge (i.e., only WiFi CSI-based training data), these systems by design cannot generate a new class label that describes the unseen example. By having additional sources (or *modes*) of knowledge, e.g, the relationship among words and their attributes in the English text, we can enhance the ability of these systems to recognize previously unseen activities, and to generate class labels that come from the additional knowledge sources (i.e., the text domain).

The secret recipe behind WiFringe’s ability to classify unseen activities is the *cross modal projection*. Through this step, we blur the difference between an activity’s CSI stream and its corresponding English word embedding and attributes, and make them (almost) equal in their feature representations. In other words, if an English word (say, run) has a known vector representation (e.g., obtained using Google’s Word-2-Vec [24] on a large dataset like Wikipedia) and a vector of attributes of the word (e.g., *movement of legs* and *motion of hands*, which is obtained from [31]), the goal of the cross-modal projection is to generate the exact same vectors when WiFringe is presented with a CSI stream of that word (i.e., CSI of running).

Figure 6.8 illustrates cross-modal projection using run as the example activity, whose word embedding and attribute vectors are X and A , respectively. In WiFringe, the activity representation of run is passed through two cross-modal projection steps to obtain Y_x and Y_A , corresponding to the word and the attribute spaces, respectively, and it is made sure that $Y_x \approx X$ and $Y_A \approx A$. In traditional feature representation, however, the projection is different (i.e., $Y \neq X$ and $Y \neq A$), and

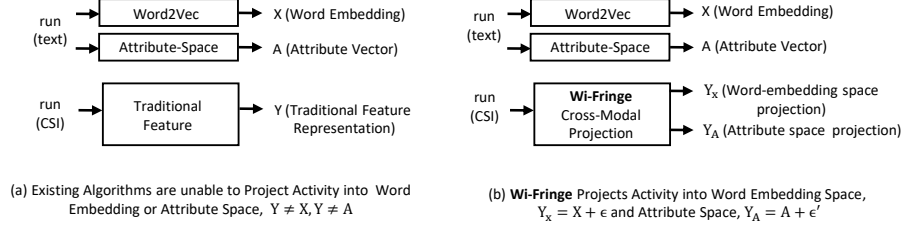


Figure 6.8: Cross-modal projections map a CSI stream to its corresponding label in the word and attribute embedding spaces.

hence, unlike WiFringe, traditional features cannot guess the word label of an unseen activity.

6.5.2 Rationale Behind Multiple Projections

The benefits of cross-modal projection from CSI to two latent spaces, i.e., word and activity-attribute spaces, are as follows:

- *Word Embedding Space:* There are over 150 thousand English words for which researchers in the natural language processing field have created semantically aware vector representations, called the *Word Embedding*. Such an embedding preserves the contextual relationship among words and puts two words that are similar in meaning or are often used in the same context closer in the representation space. By projecting the activity representation to the word embedding space, WiFringe is able to generate meaningful and context-aware representation of any CSI stream, irrespective of whether or not it has seen CSI of the same class before.

- *Activity-Attribute Space:* Human activities and bodily gestures comprise of movements by different body parts, i.e., arms, legs, head, and torso. Activities also involve external objects, e.g., an eating activity may involve the use of spoons and knives. These *attributes* create nuances in different RF-based activity feature representations. Projecting CSI onto the activity-attribute space embeds this information into the projection, as CSI implicitly gets affected by moving different sorts of objects due to their reflections. Using this embedded contextual and attribute information from CSI, WiFringe recognizes activities without any training examples.

WiFringe uses both word embedding and activity-attribute spaces for cross-modal projections to combine the predictive power of both. Combining these two help each other in the final prediction step. For example, in the word embedding space (50 dimensional W2Vec), run is surprisingly closer to pull than walk, since they appear more with similar neighbouring words. However, run and walk

are more similar in terms of activity properties. In the attribute space, therefore, run and walk are closer to each other. In such scenarios, where relying only on the contextual semantic information fails to figure the correct relationship between activities, the attribute information helps correct the misprediction.

Likewise, in many cases, the attributes extracted for two activities are very similar. For example, run and walk have the exact same attribute (binary) vectors when the attributes are constructed from online dictionaries [31]. While we desire their vectors to be closer in the attribute space for these two activities, we do not want them to be exact. This is because if they are the same, the RF projections for these two activities will also be the same. Thus, WiFringe will not be able to distinguish between walk and run if we only rely upon the attributes. In such scenarios, the word embedding helps correct the problem and WiFringe is able to separate the activities in the joint feature space.

6.5.3 Detailed Algorithmic Steps

The goal of cross-modal projection is to map state-aware representation of WiFi CSI streams (Section 6.4) to two latent spaces, i.e., the word embedding space and the activity-attribute space. The difficulty in this step is that although the class label corresponding to an activity has a fixed word embedding and a fixed attribute-vector, the CSI values corresponding to the same activity vary due to the diversity in human motion and physique. This makes the mapping from the CSI domain to the word-embedding and activity-attribute spaces a non-linear projection problem. In WiFringe, we solve this problem by using a neural network to learn the projection from CSI to the latent spaces.

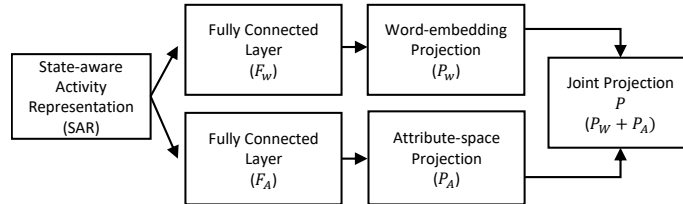


Figure 6.9: State Aware Representation is projected into both Word-embedding and Attribute space which are then aggregated for a joint projection.

The projection operation is illustrated by Figure 6.9. The state-aware representation, i.e., the output of the LSTM from Figure 6.7, is fed to two neural networks having fully connected layers.

The first network projects the state-aware representation onto the activity attribute space, and the second network projects the representation onto the word embedding space. We refer to the last layer of the neural networks that perform attribute space projection and word embedding projection as F_A and F_W , respectively.

- *Projecting onto Activity-Attribute Space.* From the attribute database provided by [31], we obtain a set of binary attributes associated with each activity. Each activity, a_i is represented as an m -dimension vector, $d_i \in \{1, 0\}^m$, where m is the total number of attributes used to define an activity. Each element $d_i^{(k)}$ is a binary indicator of whether the k^{th} attribute is true or false for that activity.

$$d_i^{(k)} = \begin{cases} 1, & \text{if attribute } k \text{ is true for activity } a_i. \\ 0, & \text{otherwise} \end{cases} \quad (6.1)$$

To get the likelihood of a CSI stream being predicted as an activity a_i , we project F_A to the attribute space. We take the dot product of the attribute vector d_i and F_A . The dot product demonstrates the similarity between the projection F_A with attribute vector d_i . For a CSI stream from activity a_i , our model's target is to increase the similarity between d_i and F_A . The similarity which denotes the likelihood of F_A 's probability of belonging to activity a_i in attribute space is calculated as a dot product:

$$P_A^i = d_i \cdot F_A \quad (6.2)$$

- *Projecting onto Word-Embedding Space.* For an activity, a_i whose word embedding is w_i , we want to project the CSI-based state-aware representation as close as possible to w_i . Therefore, for a CSI stream of an activity a_i , our target is to project F_W to be close to w_i in vector space. This results in a higher value of dot product between F_W and w_i . Therefore, the similarity in word embedding space is defined as following:

$$P_W^i = w_i \cdot F_W \quad (6.3)$$

- *Projecting onto the Joint Space.* To obtain a joint projection on both the attribute and the

word embedding space, we employ an ensemble approach to combine the two projections from Equations (6.2) and (6.3) as follows:

$$P_i = P_A^i + P_W^i \quad (6.4)$$

where, P_i carries the confidence of a CSI segment's probability of belonging to class a_i . For $|a|$ number of activities, given F_A and F_W extracted using state-aware representation for a CSI segment X as described in Section 6.4, the probability of $X \in a_i$ is calculated using the following softmax operation:

$$p(a_i|F_A, F_W) = \frac{e^{P_i}}{\sum_{j=1}^{|a|} e^{P_j}} \quad (6.5)$$

6.6 Two Stage Classifier

WiFringe employs a two-stage classifier to infer the most likely activity type for an input CSI stream segment. The first-stage classifier determines whether the input CSI stream segment belongs to a *seen* or an *unseen* class. The second-stage classifier makes the final determination of the most probable activity type for the input CSI stream segment.

6.6.1 The Need for Two-Stage Classifier

Neural networks tend to memorize patterns in data from training. Thus, even for an unseen category, the network tries to project an input close to one of the seen classes in the state-aware representation space. While this serves our purpose of classifying an unseen activity, it affects the classification performance of a generalized system where the system may encounter examples from both seen and unseen classes. Since the unseen classes are projected too close to the seen classes, they will be classified as one of those seen classes. In Figure 6.10(a), we plot the first two principal components of our proposed state-aware activity representation for two activities: run and walk. Here, the state-aware representation is trained with samples from walk. We see that for these two activities, the projections are very close to each other with some overlaps. This results in errors in deciding the samples from run (an unseen category) and they get classified as walk.

To overcome the problem posed by neural network based state-aware representation in distinguishing seen vs. unseen categories, we employ a classifier which is based on the signal characteristics such as the time-frequency representation. To be more precise, while some activities such as run and walk have similarity in their attributes, they still have distinguishable signal characteristics

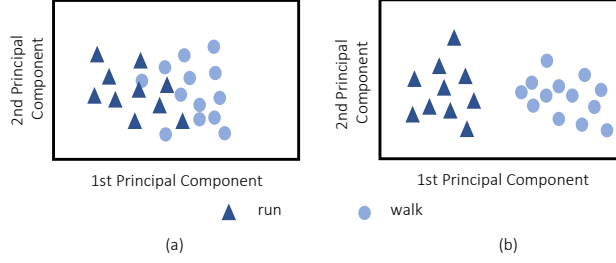


Figure 6.10: (a) State-aware activity representation projects samples of unseen class *run* close to samples of seen class *walk* (b) Spectrogram level feature projects the samples of unseen class *run* far from the samples of seen class *walk*.

that are embedded in WiFi CSI. Past works [120, 16] have proven the distinguishable power of traditional signal-level features such as Short Time Fourier Transform (STFT) and Discrete Wavelet Transform (DWT). We use STFT to detect if a sample is from a seen or an unseen category. STFT gives us the changes in frequency components of the signal along the time axis.

In Figure 6.10(b), we plot the first two principal components of the STFT for the samples *run* and *walk*. We see that they are well separated in the STFT representation, which allows us to determine if an example is from a seen or an unseen category.

6.6.2 Detailed Algorithmic Steps

The two steps of the algorithm are as follows:

- *Seen vs. Unseen Detection.* We devise a simple threshold-based decision algorithm to determine whether an input CSI stream segment belongs to an *unseen* class. We use K-means [184] clustering algorithm to cluster the STFT of the training samples of the seen category classes. This gives us K cluster centers, C_1, C_2, \dots, C_K for K seen classes. For an input CSI stream segment, u , it belongs to an *unseen* class if the following condition is true:

$$\min_{s \in S} \|C_s - STFT(u)\| > \Omega \quad (6.6)$$

where, S is the set of *seen* classes and Ω is an empirically determined threshold that maximizes the accuracy of *seen* vs. *unseen* class detection, and $\|\cdot\|$ is the Euclidean norm. When this condition is false, u belongs to a *seen* class.

- *Classification.* If the CSI segment is recognized as from a *seen* category, only the labels from

seen category are considered and the class label is obtained by applying the following equation:

$$a_i p(a_i | F_A, F_W) \quad (6.7)$$

On the other hand, if the CSI segment is recognized as from an *unseen* category, we exclude all the labels from the *seen* category and only the labels from the *unseen* category are considered and the class label is obtained by applying Equation 6.7.

6.7 Experimental Setup



Figure 6.11: (a) Intel Nuc with Antennas. (b) Experimental Setup.

6.7.1 WiFi CSI Collection

We implement WiFringe on an Intel NUC [124] mini PC. The mini PC is interfaced with an Intel NIC 5300 [25] and three omni-directional antennas with 6 dBi gain. To extract CSI values from the NIC card, we use the tool developed by Halperin et. al [125]. This tool provides a modified firmware for NIC 5300 and an open source Linux driver with user space tools to collect CSI information of the received packets. We use a Netlink router as an access point (AP). The mini PC pings the AP periodically at a certain interval and the CSI is extracted from the packet that is received by the mini PC as an echo from the AP. The sampling rate of ping is set at 500 sample per second. Therefore, according to Nyquist theorem [2] our system detects activities with a maximum of 250 Hz, which is sufficient for human activities [120]. The PC and the AP are placed 12 feet apart from each other. Experiments are done in a student apartment and an office space. Both rooms have typical furniture, e.g., two tables, multiple chairs, and cabinets. Activities are performed by four volunteers (1 female and 3 males). Figure 6.11(a) shows the mini PC Intel NUC connected to three antennas

which we place on a stand during the experiments. Figure 6.11(b) shows the the two rooms with our experimental setup.

6.7.2 Noise Reduction

CSI data extracted from the NIC card suffer from dominant noise, which makes gesture recognition difficult. To remove noise, we follow [120]’s PCA-based denoising technique. Since the Channel Frequency Response (CFR) for different subcarriers is a linear combinations of the same set of waveforms with different initial phases, when a gesture is performed, the changes of CFR value in different subcarriers are correlated. To extract the changes in CFR values caused by the movement in different subcarriers, we apply PCA on the CSI stream collected from 30 subcarriers for each TX-RX antenna pair. The ordered principal components give us the most variances experienced across all sub-carriers. We discard the first principal component stream as it contains dominant effect of noise. The second and the third components contain clear effect of gestures and less effect of noise. We choose the third stream as it has shown better noise immunity. Although the selected stream is less prone to noise, it still contains some effect of unwanted high-frequency noise. Hence, the stream is passed through a Butterworth filter [126] to remove static noise. We set a cut off frequency of 50Hz as human activity is usually below 50Hz in frequency. Thus, we obtain a de-noised CSI stream from all sub-carriers.

6.7.3 Gesture Segmentation

In order to detect the start and the end of a gesture, we instruct the volunteers to perform gestures with a brief pause between them. As the peaks in a CSI stream depends on the initial position of the user, which is variable for different gestures, we cannot segment gestures by directly applying a threshold on the CSI values. Instead, we follow [119] and take the first order difference of the stream which is stable during the pause and fluctuates during gestures. We apply a threshold-based peak detection technique on the first order difference to detect gestures. The segments on the first order difference that are above a predefined threshold are detected as gestures. We refer [119] to the readers for more details on pre-processing of CSI stream.

6.7.4 Data Augmentation

To avoid overfitting and to make our model robust, we use a data augmentation technique to increase the user contributed data size by about 20 times. First, we apply a geometric transformation to the time-frequency representation of the raw CSI signals. Specifically, we use translation to shift

the spectrogram along the time axis. We exclude shifting along the frequency axis, as it does not reflect any physical world phenomenon. Shifting along the time axis reflects the effect of recording the same gesture at different timestamps. Second, We superimpose simulated noise on the signal to augment the dataset further. Each signal is superimposed with a Gaussian white noise [2]. The noise model is generated offline. Injecting noise captures environmental effect and makes the model resilient to noisy environment.

6.7.5 Empirical Dataset

In this section, we describe our dataset collection method. WiFringe leverages knowledge from the text domain using the activity names to classify activities without training examples. For that reason, we need *named* activities that have semantic representation in English language. For example, WiFringe is able to classify *run* if it has seen training examples of *walk* based on the semantic relationship between these two words. However, WiFringe won't be able to classify activities without having semantic meaning in English or any other language.

Based on our study of named activities from [185, 186, 142], we collect 20 most common named activities for our empirical evaluation. To the best of our knowledge, this is the largest dataset in terms of the number of classes considered in WiFi-based activity recognition systems. Existing literature have used 6–8 types of gestures for supervised activity recognition from WiFi [119, 142, 16, 120]. Our dataset contains activities collected from four volunteers in two different rooms with different orientations and furniture. Our dataset is diverse and it stresses out the algorithmic components of WiFringe. In Table 6.1, we provide the list of the 20 activities clustered with major attributes. On average, each class have 100 samples.

Table 6.1: Twenty categories of activities are used in the experiments. Each category has 100 examples on average.

Category	Activities
Freehand Gestures	Point, Raise, Rub, Scratch, Shake, Toss, Circle, Arc.
Object-Human Interactions	Drink, Eat, Push, Pull.
Upper/Lower-Body Gestures	Sit, Stand, Bow, Duck, Kick.
Mobility	Jump, Walk, Run.

6.8 Experimental Results

6.8.1 Accuracy of Unseen Class Detection

In this experiment, we report the accuracy of WiFringe for *unseen* classes. These are the classes for which WiFringe did not have any training examples during training phase. As state-of-the-art systems are not capable of recognizing activities without prior training examples, we are unable to compare them with our solution (i.e., any existing algorithm will have an accuracy of random guess at best). Hence, we report WiFringe’s performance in this section by varying the number of unseen classes and compare it with two variants of our algorithm: (1) projecting State-Aware Representation (SAR) onto only word embedding (W2Vec) space, and (2) projecting State-Aware Representation (SAR) on only activity-attribute space. This comparison shows the performance boost due to joint space projection. Note that WiFringe is able to correctly label an unseen activity only if it has learned the representation of classes that are semantically related to it in terms of movement of body parts and word-level embedding. Therefore, while selecting *unseen* classes, we keep at least one class from the *seen* classes which is close to it in word embedding and attribute space. For example, we select *push* as an unseen class and keep *pull* as a seen class which are semantically related. A *push* is similar to a *pull* in motion of the body parts, e.g., both require movement of hand(s) but legs remain unmoved. Furthermore, the two English words *push* and *pull* are closer in the word embedding space as they appear in similar contexts.

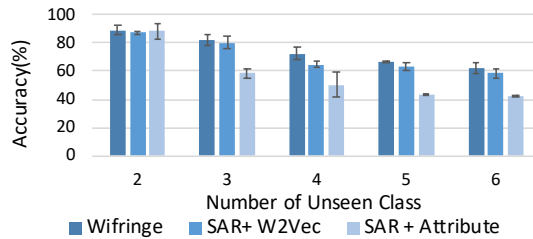


Figure 6.12: WiFringe’s accuracy for unseen activity classification.

In Figure 6.12 we report the accuracy of WiFringe in recognizing the unseen classes of activities when 2–6 types of activities are from unseen classes. We evaluate with different combinations of *seen* and *unseen* activities and present the mean accuracy and variance in the plot. In Figure 6.12, we see that for two unseen classes, we achieve a classification accuracy of around 90%. With only word embedding and attribute space projection, the accuracy is 87% and 88%, respectively. For

three unseen activities, we get an accuracy near 83% with WiFringe. With only word embedding projection the accuracy is around 80%, but with attribute space projection the accuracy drops to 60%. This is due to the similarity of activities in attribute space, which results in very similar attribute vectors. Therefore, projecting only on attribute space makes the classification harder. As the number of unseen classes increase to 4, 5, and 6, the accuracy becomes to 73% , 67% and 62%, respectively. In all the cases, joint space projection boosts the performance in comparison with single space projection. As the number of unseen classes increase, the problem becomes harder since the model has to differentiate between more classes without training data. We report up to six unseen classes in the plot, however, for seven unseen classes, our accuracy is around 53%. With random selection, the accuracy for seven unseen class is 14.28%, so WiFringe is still better by almost 40%.

6.8.2 Accuracy of Seen Class Detection

In this experiment, we evaluate WiFringe’s *seen* class detection performance by keeping all the classes in seen category. We compare WiFringe with other baseline classification algorithms. Recent works [23, 109] have demonstrated the efficiency of using Convolutional Neural Network(CNN)-based classifiers for WiFi based activity recognition. Following these works, we compare WiFringe with a CNN classifier optimized for our dataset with five convolutional layers along with batch normalization and dropout layers. As mentioned earlier, state-aware representation (SAR) by itself can be used as a feature for supervised classification. Therefore, we also report the performance of state-aware representation (SAR) integrated with a softmax layer. In addition, we also show the performance of projecting state-aware representation only to word embedding space and attribute space. We use five-fold cross-validation by randomly selecting training and testing examples each time. We also report the classification performance of a shallow classifier with a traditional handcrafted feature (i.e., STFT). We report the mean and variance of classification accuracy.

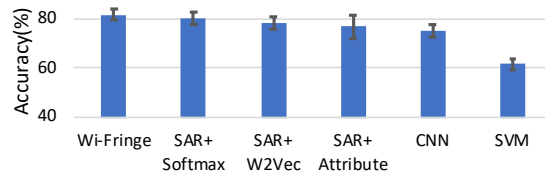


Figure 6.13: WiFringe’s accuracy is higher than baseline algorithms in seen class detection.

In Figure 6.13, we find that WiFringe achieves a mean accuracy of 82%. On the other hand, state-aware representation (SAR) along with softmax layer is able to achieve around 80% mean accuracy. The performance boost of WiFringe is due to the fact that from joint space projection, our model is able to classify activities by integrating knowledge from both word embedding and attribute domain. Projecting state-aware representation onto only word embedding and attribute space yields accuracy of 78% and 76%, respectively. With CNN, we have an accuracy of 74%. Therefore, the proposed state-aware Representation improves the accuracy for seen class detection by 8% than CNN model. This improvement is achieved since the state-aware representation learns the temporal relationship among the states in an activity. With an SVM, we see the accuracy is around 62%. Therefore, it is evident that WiFringe is able to achieve better accuracy than other classifiers in seen class detection. Note that WiFringe’s architecture is modular and generic. We can also integrate other representation to WiFringe if necessary.

6.8.3 Accuracy of Seen vs Unseen Class Detection

In this section, we present the accuracy of our threshold based Seen vs. Unseen detection’s performance. The accuracy is threshold dependent. In Figure 6.14, we plot the accuracy for $\Omega \in [4.0 - 5.25]$. We observe that setting a high threshold fails to detect many class as *unseen* and the accuracy drops for the *unseen* classes. With high threshold, the unseen class samples have to be very far apart from any cluster center of the seen class clusters. On the other hand, setting the threshold too small leads to poor results for *seen* classes as it determines majority CSI stream sample as *unseen*. Hence, there is a trade-off between the *seen* and *unseen* class detection accuracy. The optimum threshold is 4.75, for which, the classification accuracy of the *seen* and *unseen* classes are around 80%.

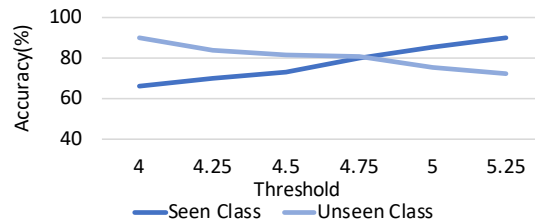


Figure 6.14: The accuracy of *seen* and *unseen* class detection depends on the threshold Ω ’s value.

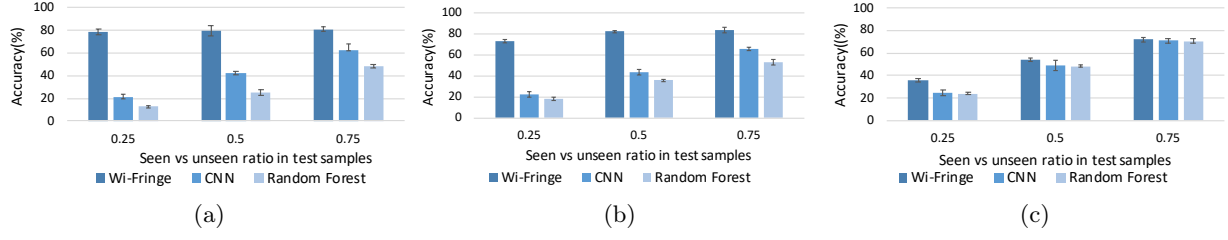


Figure 6.15: WiFringe performs better than baseline algorithms for all cases. a) When 8 out of 10 classes are in seen category, WiFringe has almost 80% accuracy for all cases. On the other hand, baseline algorithms’ accuracy drops below 20% for cases when unseen category dominate in test samples. b) For 5 out of 10 classes in seen category WiFringe outperforms all baseline algorithms for different cases. c) When 2 out of 10 classes are in seen class, for .25 fraction of test samples coming from seen category WiFringe’s performance drops below 40% which is still 1.5 times better than baselines’ performance.

6.8.4 End to End Evaluation

To quantify WiFringe’s end-to-end performance, we report its classification accuracy for an application scenario. We monitor a user’s home activity for ten different classes: {push, pull, run, sit, rub, walk, stand, eat, scratch, drink}. We consider three different training scenarios. First, we consider that the user provides 8 out of 10 activity classes’ examples to WiFringe during training, i.e., the number of classes in seen and unseen categories are 8 and 2, respectively. Second, we consider the case where 5 out of 10 activity classes’ examples are given to WiFringe during training. The last and the hardest test case is a scenario where WiFringe has only 2 activity classes’ samples during training, i.e., 8 out 10 classes are unseen.

In Figure 6.15, we report the performance of WiFringe along with two baseline algorithms: a convolutional neural network (CNN) and a random forest classifier for all three aforementioned scenarios. For each scenario, we consider three cases where we vary the ratio between samples from seen and unseen classes in the test dataset in the following ways: a) $\frac{\#seen}{\#unseen} = 25\%$, b) $\frac{\#seen}{\#unseen} = 50\%$ and c) $\frac{\#seen}{\#unseen} = 75\%$. Here, # denotes number of samples.

In Scenario 1 (Figure 6.15(a)), where only 2 classes are in the unseen category, WiFringe shows an accuracy around 80% for all the cases, whereas the baseline algorithms’ accuracy drops below 20% when most of the samples are coming from the unseen category. Note that the unseen classes are chosen by keeping one of their closest neighbours in the word embedding and attribute space in the seen category.

In scenario 2 (Figure 6.15(b)), WiFringe achieves an accuracy of 84% for case 3 with majority of

the samples in test cases coming from the seen classes. However, when the ratio of seen classes in the test data gets decreased in case 1, the accuracy drops to 73%. Yet, WiFringe’s performance is better than both baselines by a margin of greater than 40%.

In scenario 3 (Figure 6.15(c)), where only two classes are in the seen category, the accuracy for the case where 75% of test samples are from seen classes reaches up to 72% for WiFringe. However, for case 1, the accuracy drops to 36% where 75% of the test samples are from the unseen categories. This drop is due to the fact that most of the classes are now in unseen category and WiFringe has very few classes to learn the mapping function from RF to text domain. For this case, baselines achieve a maximum accuracy of only 24%. Therefore, it is evident that WiFringe’s performance is better than traditional classification algorithms in all the cases.

6.8.5 Execution Time

Although only a few WiFi chipsets support CSI information extraction, we believe that in the future, with new tools and chipsets, WiFringe will be runnable completely on a variety of platforms including smartphones. In this experiment, we report the execution time of the proposed algorithms for a computer (Macbook Pro). Data are read from the file system of the device. In Table 6.2, we list the execution times for inference of the two models. We see that the state-aware representation (SAR), which has convolutional and recurrent layers, takes 12ms on average for inference on a Macbook. On the other hand, the execution time for cross-modal projection, which has two fully connected layers, is around 2ms on a Macbook.

Table 6.2: Inference time of deep networks.

Network	Macbook Pro
State-Aware Representation	12ms
Cross-Modal Projection	2ms

6.9 Related Work

6.9.1 Device-free Sensing

- *RSSI based*: WiFi based sensing have opened the doorway for device-free activity monitoring in the last couple of years. Researchers have used wifi signal characteristics such as signal strength (RSSI) and channel state information(CSI) for activity recognition [127]. RSSI based activity recognitions have been proposed in [18, 128, 129, 130, 131]. However, RSSI based gesture recognition

system have limitations in detecting fine-grained gestures. Moreover, all of these works require training examples to detect a particular activity class. [18, 129] uses pattern matching algorithm to find the best match between the target gesture with pre-defined gestures. Our target in this chapter, is classifying activities and gestures without training examples and our framework can be ported to RSSI based systems.

- *Special Device based:* There have been several works in gesture and activity recognition using specialized devices and radars. [132, 23, 133] use FMCW [134, 135] radio to monitor user activity. [23] trains a CNN classifier to detect human motion. [132] trains a hidden markov model using time-velocity feature for activity recognition. [136] uses a 5-antenna receiver and a single-antenna transmitter to perform gesture classification, in the presence of three other users performing random gestures. They use doppler shifts to match with pre-defined gestures for an incoming test data. It is evident that none of these specialized device based sensing system deals with ctivity recognition without prior examples. They all require labelled training examples for activity recognition.

- *CSI based:* With the availability of CSI from network interface cards, multiple works [137, 138, 139, 140] have emerged which exploit CSI information for gesture and activity recognition. [16] proposes a signal profile matching technique to detect loosely defined daily activities that involve a series of body movements over a certain period of time. [120] proposes correlation between CSI amplitude value and gesture speed to build model for gesture recognition. [141] uses variations in the Channel State Information (CSI) to classify gaits of humans. [119] uses translation based data augmentation technique to make gesture classification models robust to user orientation. [142] proposed multi-person gesture recognition system by generating virtual gesture samples and combining them to create an exhaustive template matching algorithm. Recent works such as [106, 107, 108] use deep learning based techniques such Convolutional Neural Networks to recognize activities from CSI. [143, 106] proposes recurrent neural network based activity classifier , however ours is the first work to model the micro-activity or state transition which constitute an activity. [143] predicts label for each segment of the CSI stream. On the other hand, we need only one label for the whole CSI stream which makes our model to learn the entire sequence of states to make a decision about the activity. Besides, we learn the local and transitional feature in an end to end manner with bi-directional recurrent neural network, which makes our model stronger. Ours SAR is the first model to incorporate local feature of states and their transition in a bi-directional fashion. [109] proposed

an adversarial network to learn environment independent signal characteristics from gestures. In this work, the authors used unlabelled data to improve their model’s performance. But their proposed system is not able to infer an activity without having any training example. Very recent work [110] proposed velocity profiles as environment independent feature to solve the problem of environmental effect on CSI. All of these works rely on provided training examples to classify a particular class of activity. WiFringe deals with classification of activity from WiFi CSI data without any training examples. This is significantly different from current state of the arts.

6.9.2 Zero Shot Learning

In image domain, learning from limited or no data has been explored recently. [123, 171, 187] focus on learning suitable image representation to classify images from very few or only one examples. These works are totally different from ours as we want to recognize activities from WiFi without any training data. The other branch of learning with limited data focuses on zero shot learning for image classification. The earlier practices of zero shot learning [149, 172, 188] for image classification problem infer the labels of unseen classes using a two step algorithm. First, the attributes of the sample is inferred and then the class label is predicted from an attribute database, which has most similar attributes with the image. Recent works [173, 174, 189] have explored the mapping between image features and semantic space by projecting image features into word embedding space. Similar line of works [175, 150, 176, 177] project image feature into semantic space then searches the nearest class label embedding using ranking loss. Although these papers propose zero shot learning method for images, none of them addresses the problem for RF domain and activity recognition. [31, 190, 191, 192] do activity recognition using zero shot learning for RGBD data. [74] proposes zero shot learning for audio event recognition. But, the sequential nature of activity and need for external attribute knowledge makes our problem more challenging. [193] proposes zero shot learning for body-worn IMU based activity recognition. However, our work is the first system to propose a zero shot learning method for WiFi based activity classification where we overcome the challenges for cross-modal learning between text and RF domain. These works on zero shot activity detection use an attribute database of the gestures to recognize unseen types. Our contextual word-embedding based approach does not require the attributes of the unknown gestures or activities beforehand. To the best of our knowledge, we present the first system with capabilities of inferring activities from RF signal without training examples. We propose the first work to integrate textual domain with

RF. We also present novel feature representation for RF based activity monitoring.

6.10 Discussion

- *Synergy Between Seen and Unseen Class* For an unseen activity, WiFringe is able to detect it with high accuracy if there is a seen class with similar class label property, i.e., similar word embedding and attribute vectors. WiFringe assumes that activity classes with similar properties have semantically similar class labels. We use this assumption to learn the non-linear mapping function to project WiFi signals onto the word embedding and attribute spaces. Without semantically similar training examples corresponding to an unseen class, a zero-shot learner fails to recognize examples from that unseen class. However, in a large dataset, the chances that there is no semantically similar training example to an unseen class is relatively low. To demonstrate this, we conduct an experiment. We keep pull as an unseen class and exclusively put the following classes in the seen category in a round-robin fashion: {push, throw, point, kick}. In Figure 6.16, on the X-axis, we put these classes with their corresponding distances in the joint word embedding and attribute space with pull. We see that, as the distance increases from the unseen category, the accuracy of unseen class recognition drops. When push is in seen category, the classification accuracy for pull is more than 90%. As push and pull are close in word embedding and attribute spaces, the presence of push in training data helps recognize the samples from pull. As we go further away from pull in the word embedding and attribute space, the accuracy drops even more. For kick, which is the farthest from pull, we see that the classification accuracy for *pull* is around 60%.

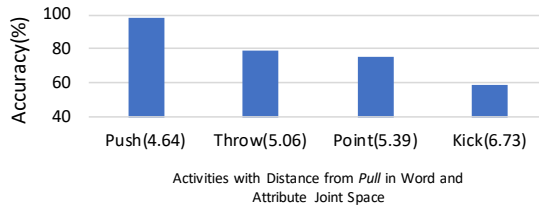


Figure 6.16: WiFringe’s performance in unseen activity recognition is dependent on its relationship with classes in seen category.

- *Necessity of User-Provided Tag-List.* The user-provided additional labels do not have any influence on the training phase. They are only used in the classification step after the training has been completed. If we do not have these additional labels, then the search space becomes too large (i.e., as big as having all the words in our database for a language) in the classification stage.

Therefore, the user provided labels for unseen class is important in getting better accuracy for unseen activities.

- *Environmental Effect.* Our main objective in this chapter is to propose the first cross-modal RF to text projection to enable zero-shot activity classification. The proposed representation learning algorithm does not consider environment or multi-person effect. Recent works [109, 142] have showed promising results on solving these issues. Both State-Aware Representation and Cross modal projections are modular and generic. We can port these solutions to WiFringe to handle these artifacts.

6.11 Summary

In this chapter, we present the first WiFi-based device-free activity recognition system that does not require training examples for all activities. We propose a robust state-aware representation of RF signature associated with activities to preserve their contextual information. We propose a novel way to embed contextual information from the text domain to the RF domain by projecting RF data onto the word embedding and attribute space. We use this cross-modal RF embedding and propose a general classifier to recognize both *seen* and *unseen* activities. We collect WiFi data for 20 different activities from four volunteers and show that WiFringe is capable of inferring activities from WiFi without training examples with 62% – 90% accuracy for 2–6 unseen classes.

CHAPTER 7

Privacy Preserving Acoustic Sensing

7.1 Introduction

Having reached the milestone of human-level speech understanding by machines, continuous listening devices are now becoming ubiquitous. Today, it is possible for an embedded device to continuously capture, process, and interpret acoustic signals in real-time. Tech giants like Apple, Microsoft, Google, and Amazon have their own versions of continuous audio sensing and interpretation systems. Apple’s Siri [1] and Microsoft’s Cortana [194] understand what we say, and act on them to fetch us a web page, schedule a meeting, find the best sushi in town, or tell us a joke. Google and Amazon have gone one step further. Android’s ‘OK Google’ feature [195], Amazon’s Echo [196], and Google Home [?] devices do not even require user interactions such as touches or button presses. Although these devices are activated upon a hot-word, in the process, they are continuously listening to everything. It is not hard to imagine that sooner or later someone will be hacking into these cloud-connected systems and will be listening to every conversation we are having at our home, which is one of our most private places.

Furthermore, there is a recent trend in the IoT world that many third-party, commercial IoT devices are now becoming voice enabled by using the APIs offered by the voice-controlled personal assistant devices like Echo or Google Home. Henceforth, we will refer to these devices interchangeably as smart hubs, home hubs, or simply hubs. For example, many home appliances and web services such as Google Nest thermostats [?], Philips Hue lights [?], Belkin WeMo switches [?], TP-Link smart plugs [?], Uber, and Amazon ordering are now ‘Alexa-Enabled’ – which means, we can send voice commands to an Amazon Echo device to actuate electrical devices and home appliances. Because it enables actuation and control of real-world entities, a potential danger is that they can be activated by false commands (e.g., sounds from a TV) and/or unauthorized commands (e.g., an outsider commands someone’s home hub to control his home appliances, places a large purchase on his Amazon account, or calls a Uber driver). A careful scrutiny of voice commands is therefore a

necessity to ensure safety and security.

Unfortunately, none of the existing voice-enabled home hub devices take any of these vulnerabilities into account while processing the audio. They merely apply standard noise-cancellation [197, 198, 199] to suppress non-speech background sounds in order to improve the signal-to-noise ratio. However, this process alone cannot eliminate acoustic signals from unwanted acoustic sources that happen to be present in the environment and overlap in time and/or frequency with a user’s voice signals. There is also no support for personalization of speech commands in these devices.

In this chapter, we study the ‘overhearing’ problem of acoustic sensing devices, and develop a system that mitigates personal or contextual information leakage due to the presence of unwanted sound sources in the acoustic environment. Instead of developing a special-purpose, application-specific embedded system that works only for voice commands, we address the problem in a generic setting where a user can define a specific type of sound as primary (i.e., a relevant or essential sound type for the application), or secondary (i.e., a non-essential and potentially privacy concerning sound). For example, the voice of a user issuing a command is a primary source for home hubs, whereas any identifiable background noises such as the sounds from appliances, other conversations are examples of secondary sounds. Furthermore, instead of proposing modifications to existing home hubs, we build an independent embedded system that connects to a home hub via its audio input. Considering the aesthetics of home hubs, we envision the proposed system as a smart sleeve or a cover for these home hubs. The proposed system has necessary hardware and software to capture the audio, isolate signals from distinct sound sources, filter out signals that are from unwanted sources, and process the signals to enforce policies such as personalization before the signals enter into an untrusted system like Amazon Echo or Google Home. The device is programmable, i.e., an end user is able to configure it for different usage scenarios.

Developing such an embedded system poses several challenges. First, in order to isolate acoustic sources, we are required to use an array of microphones. Continuously sampling multiple microphones at high rates, at all times, and then processing them in real-time is extremely CPU, memory, and time demanding for resource-constrained systems. Second, to train the system to distinguish primary and secondary sources, we are required to collect audio samples from an end user to create person or context specific acoustic models. To the users, it would be an inconvenience if we require them to record a large number of audio samples for each type of sound in their home. Third, because

no acoustic source separation is perfect, there will always be residuals of secondary sources after the source separation has been completed. These residuals contain enough information to infer personal or contextual information, and hence, they must be eliminated to ensure protection against information leakage.

In this chapter, we address all these challenges and develop a complete system called the *SoundSifter*. The hardware of SoundSifter consists of a low-cost, open-source, embedded platform that drives an array of microphones at variable rates. Five software modules perform five major acoustic processing tasks: to orchestrate the sampling rates of the microphones, to align the signals, to isolate sound sources, to identify primary source, and to post process the stream to remove residuals and perform speaker identification. At the end of the processing, audio data is streamed into the home hub. We thoroughly evaluate the system components, algorithms, and the full system using empirical data as well as real deployment scenarios in multiple home environments.

The contributions of this chapter are the following:

- We describe SoundSifter, the first system that addresses the overhearing problem of voice-enabled personal assistant devices like Amazon Echo, and provides an efficient, pragmatic solution to problems such as information leakage, and unauthorized or false commands due to the presence of unwanted sound sources in the environment.
- We devise an algorithm that predicts a spectral property of incoming audio to control the sampling rates of the microphone array to achieve an efficient acoustic source separation.
- We devise an algorithm that estimates noise directly from the secondary sources and nullifies residuals of secondary signals from the primary source.
- We conduct empirical and real-world experiments to demonstrate that SoundSifter runs in real-time, is noise resilient, and supports selective and personalized voice commands that commercial voice-enabled home hubs do not.

7.2 Usage Scenarios

We describe two motivating scenarios for SoundSifter that are specific to voice-controlled home hubs.

7.2.1 Voice-Only Mode

Continuous listening devices of today hear everything in their surrounding acoustic environment. The goal of *voice-only* mode is to ensure that only speech signals enter into these devices while all other sound sources in a home environment such as sounds from TV, appliances, non-voice human sounds such as laughter, crying, coughing, sounds of an activity such as cooking, cleaning, etc. are completely filtered out of the system. This is achieved in SoundSifter by using a combination of source separation, recognition, and suppression.

7.2.2 Personalized Commands

ordering something not wanted, giving input to to-do list, child mode

Command personalization may be of multiple types. we dont do the 1st two Firstly, voice commands containing an exact sequence of words or an utterance. Secondly, voice commands that contain a certain keyword or a set of keywords in it. Third, voice commands of a certain person. These are achieved in SoundSifter by first applying the voice-only mode, and then performing additional acoustic processing such as speech-to-text and speaker identification.

Note that, although we only mention home hub related usage scenarios of SoundSifter in this section, the generic notion of primary and secondary sound allows us to configure the system for other applications where it can filter in/out different primary/secondary sounds as well.

7.3 Overview of SOUNDSIFTER

SoundSifter is motivated by the need of a smart acoustic filter that inspects audio signals and takes proper actions such as filtering sounds from unwanted secondary sources and checking the content of primary signals before letting them into a voice-enabled home hub or any such continuous listening devices.

SoundSifter connects to a home hub or a mobile device’s audio jack, and can be thought of as an extension to their on-board audio I/O subsystem. It captures and processes all incoming audio streams, isolates audio signals from distinct sources, identifies and blocks out any sound that has not been labeled ‘primary’ by a user during its installation, and only lets processed audio enter into a hub for further application-specific processing.

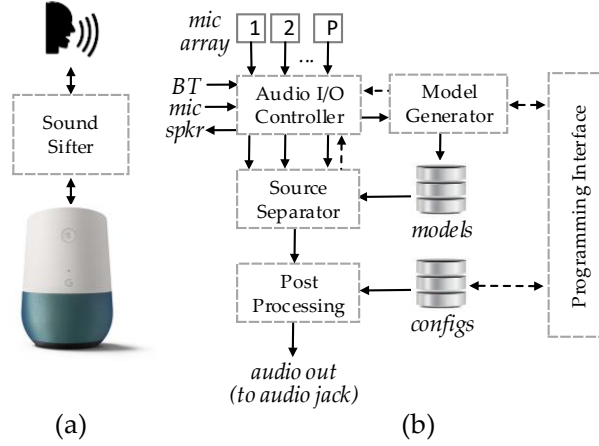


Figure 7.1: (a) System interface, and (b) block diagram of SoundSifter.

7.3.1 Basic Workflow

Figure 7.1(a) shows how SoundSifter sits between audio sources and a home hub. Further details of its internal processing blocks are shown in Figure 7.1(b). The system has an *Audio I/O Controller* that controls an array of microphones (required for source separation), a speaker, Bluetooth and an audio jack to support external audio I/O. The *Source Separator* executes the acoustic source separation algorithm by controlling the microphones via the audio I/O controller and using precomputed acoustic models. The *Post Processing* module further filters and obfuscates the already separated primary stream to eliminate traces of residuals from other sources and to enforce policies (read from a configuration file) such as personalized commands. The policies are further described in 7.3.3 as usage scenarios. Finally, the processed audio output is let go into the home hub via the audio jack.

7.3.2 Initial Setup and Programming

SoundSifter needs to be programmed once for each use case (described in the next section). Programming the device essentially means creating acoustic models for each type of *primary* sound involved in a scenario. Because these sounds are often user- or home-specific, this process requires the active engagement of a user. To make the process simple, a user is provided with a mobile app that is as easy as using a media player. The mobile app connects to SoundSifter over Bluetooth, and interacts with it by sending programming commands and receiving responses. *No audio data are exchanged between the devices.*

In the app, the user is guided to send commands to SoundSifter to record 10s – 30s audio for

each type of primary sound, label them, and specify in which scenario they will be used. The system also needs some examples of a few common types of secondary sounds for the application scenario. However, it does not require a user to record and label all possible secondary sounds. We empirically determined that as long as SoundSifter has 2-3 types of secondary sounds per use case, it is capable of detecting primary vs secondary sounds with a high accuracy and a negligible false positive rate.

Once the recording and labeling phase is over, SoundSifter uses an algorithm to create acoustic models, deletes all raw audio data, and only the labeled models are stored inside the device in a configuration file.

7.3.3 Usage Scenarios

We describe two motivating scenarios for SoundSifter that are specific to voice-controlled home hubs.

Voice-only mode: Continuous listening devices of today hear everything in their surrounding acoustic environment. The goal of *voice-only* mode is to ensure that only speech signals enter into these devices while all other sound sources in a home environment such as sounds from TV, appliances, non-voice human sounds such as laughter, crying, coughing, sounds of an activity such as cooking, cleaning, etc. are completely filtered out of the system. This is achieved in SoundSifter by using a combination of source separation, recognition, and suppression.

Personalization: Command personalization may be of multiple types: voice commands containing an exact sequence of words or an utterance, voice commands that contain a certain keyword or a set of keywords in it and voice commands of a certain person. These are achievable by first applying the voice-only mode, and then performing additional acoustic processing such as speech-to-text and speaker identification. Although we only mention home hub related usage scenarios of SoundSifter in this section, the generic notion of primary and secondary sound allows us to configure the system for other applications where it can filter in/out different primary/secondary sounds as well. Furthermore, we did not implement a speech-to-text converter in our system due to time constraints, which we leave as future work.

7.4 Studying the Problem

Prior to the development of SoundSifter, we performed studies and experiments to understand the nature of the challenge.

7.4.1 Need for Source Separation

As an alternative to source separation, we looked into simpler solutions such as filtering and noise cancellation. However, those attempts failed since the sounds that may be present in the environment overlap with one another in the frequency domain.

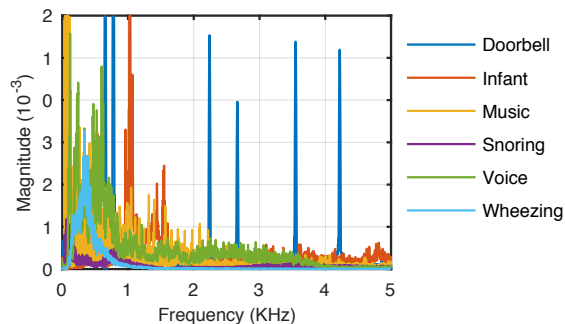


Figure 7.2: Frequency plots of variety of sounds.

In Figure 7.2 we plot frequency characteristics of a selected set of sounds. For example, speech (< 4 KHz) covers the full range of snoring (< 1.5 KHz) and asthmatic wheeze (< 1.3 KHz). Crying infants and doorbell sounds range from 500 Hz to 1.6 KHz and 4.2 KHz, respectively. Music overlaps with all sounds. Besides these, we also analyzed home appliances such as a blender, a washing machine, door slams, toilet flushes, speech signals of different sexes and age groups, and asthmatic crackling, and came to the conclusion that spatial information is the most effective method for identifying and isolating primary information containing signals from other types of unwanted sounds in a general purpose setting.

7.4.2 Number of Microphones

For effective source separation, we are required to use an array of microphones. In theory, the number of microphones should equal the number of simultaneously active sources. However, for sparse sources like audio (sources that do not produce a continuous stream), source separation can be performed with fewer microphones.

To determine an adequate number of microphones for source separation, we perform an experiment where we use an array of five microphones that captures audio signals from 2–5 simultaneously active sources: voice (primary sound), ringing phone, piano, songs, and television. Figure 7.3 shows the quality of source separation in terms of mean residuals (the lower the better) as we vary the number of microphones for different number of active sources. We observe that the residuals drop with more

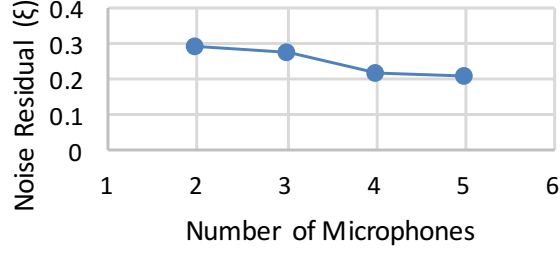


Figure 7.3: Source separation error for different number of microphones.

microphones. However, the reduction is not great for more than four microphones. Hence, we use four microphones in our system and use an additional noise removal step to nullify the remaining residuals.

7.4.3 Benefit of Rate Adaptation

According to the Nyquist theorem [200], the sampling rate of each microphone must be at least twice of the maximum frequency of any source. However, sampling an array of microphones at a very high rate costs significant CPU, memory, and power consumption. Note that our system is suitable for portable home hub devices, where a plug-in power is not always an option. In such cases, limited processing power is a challenge.

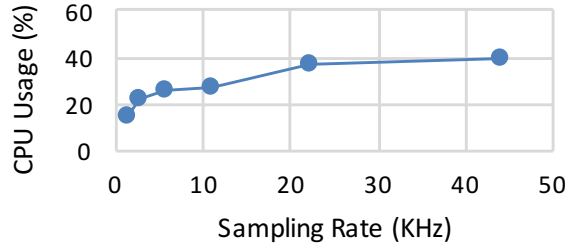


Figure 7.4: CPU usage increases with sampling rate.

To validate this, we conduct an experiment using an ARM Cortex A8-based microcontroller. In Figure 7.4, we observe that as the sampling rate is increased, CPU usage increases sharply. Memory consumption also increases from 22 KB to 704 KB as we vary the sampling rate. Based on this observation, we decide not to sample the microphones at the highest rate at all times; instead, we adopt a predictive scheme where we probabilistically choose a sufficient sampling rate for the microphone array based on previous knowledge on sound sources and signals that the system has just seen.

7.4.4 Modeling Sounds

After isolating the sources, for SoundSifter to determine which sounds to let in and which ones to block, it has to identify the source that represents the primary sound. Because primary sounds in different usage scenarios are highly subjective, SoundSifter must obtain a sufficiently large number of audio samples directly from the end user to create a robust and accurate sound classifier. On one hand, we need a large amount of training audio from the user for a robust classification. On the other hand, requiring a user to collect these data is likely to be error prone and also an inconvenience to them. Hence, it is customary to investigate techniques to create robust acoustic classifiers that generate accurate and robust models based on a limited amount of training data.

7.4.5 Need for Residue Removal

A crucial observation during our study has been that even after source separation, when we look into the stream of primary signals, we find traces of secondary sources. We realize that even though the residues are too weak to be heard, using machine learning, they can be identified and recognized.

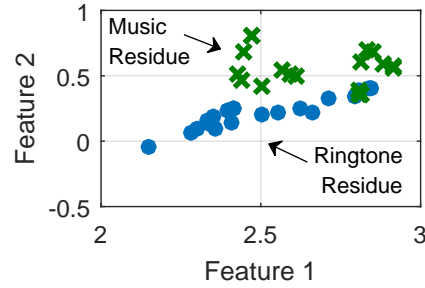


Figure 7.5: Effect of residue on primary source.

To illustrate the vulnerability of signal residue, we conducted a small scale study. We performed source separation for two cases – speech with: 1) music, and 2) a ringing phone in the background. In both cases, the same set of 20 utterances is synthetically mixed (to keep the primary sound identical) with the two background sounds. After source separation, we take the separated speech streams, compute Mel-frequency cepstral coefficient (MFCC) [21] features, and plot the utterances (total 40) in a 2D feature space as shown in Figure 7.5. In this figure, feature 1 and feature 2 denote the two highest ranked components of MFCC. It is interesting to observe that even though the residues of music and ring tone are not audible, their presence in the speech stream is statistically significant – which is enough to distinguish the two cases.

7.4.6 Analog Data Acquisition

A practical engineering challenge that we face with multi-channel audio data acquisition has been the inability of commodity embedded hardware platforms to sample analog signals at a very high rate. For example, considering the worst case where we are required to drive four analog microphones at 44.1 KHz, we need an aggregate sampling rate of 176.4 KHz. Achieving such a high rate of analog reading using off-the-shelf Linux-based embedded platforms such as Arduinos, Raspberry Pis, or Beaglebones is non-trivial. We address this implementation challenge and believe our solution will be helpful to anyone who wants to read analog audio at a high (e.g. MHz) rate.

7.5 Algorithm Design

The audio processing pipeline inside SoundSifter has five major stages. Figure 7.6 shows the stages and their interconnections. The first two stages prepare the audio streams from the microphone array for the source separation stage. These two stages together implement the proposed sampling rate adaptation scheme in order to lower the CPU and memory consumption of SoundSifter. We use FastICA [48] to perform the actual source separation. The last two stages perform further processing to identify the primary source and to nullify the residuals of other sources in it.

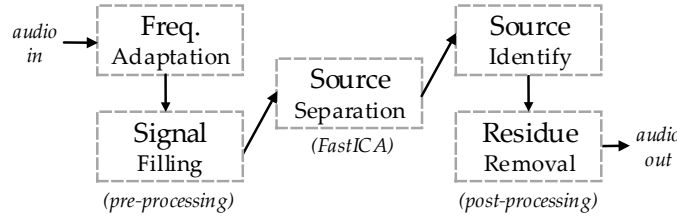


Figure 7.6: Components and interconnections inside SoundSifter’s audio processing pipeline.

In this section, we describe the first and the last two stages of SoundSifter’s audio processing pipeline, i.e. the pre-processing and post-processing stages to demonstrate how these stages work in concert to improve the efficiency and effectiveness in mitigating information leakage from unwanted acoustic sources with SoundSifter.

7.5.1 Frequency Adaptation

SoundSifter adopts a predictive scheme to determine an adequate sampling rate for the microphone array. Sampling rate adaptation in SoundSifter happens periodically, and the predicted rate is estimated based on the knowledge of the portion of the audio that the system has already seen. SoundSifter keeps an evolving Markov [201] model, whose transition probabilities help determine the

expected maximum frequency of incoming audio, given the measured maximum frequencies of the audio samples received during the last few ms.

Predicting the absolute value of the maximum frequency of incoming audio signals is practically impossible in a generic setting. However, if we divide the full range of audible frequencies into discrete levels, the transitions from one level to another can be predicted with a higher confidence. For this, SoundSifter uses six ranges of frequencies by dividing the maximum sampling rate into six disjoint sets $44.1/2^h$ KHz, where $0 \leq h \leq 5$. We denote these by the set $\{f_i\}$, where $0 \leq i \leq 5$.

Furthermore, since we are aiming at applying the *past predicts the future* principle, a question that immediately comes up is how much past data to use for an accurate prediction of the desired sampling rate? To determine this, we conduct an experiment to quantify the look-back duration. We experiment with two different Markov models:

- **1-Step Look-back:** Uses a 6-state Markov model, where each state corresponds to a frequency in $\{f_i\}$, resulting in a 6×6 transition matrix having transitions of the form $f_i \rightarrow f_j$.
- **2-Step Look-back:** Uses a 36-state Markov model, where each state corresponds to a pair of frequencies (f_i, f_j) , resulting in a 36×36 transition matrix having transitions of the form $(f_i, f_j) \rightarrow (f_j, f_k)$.

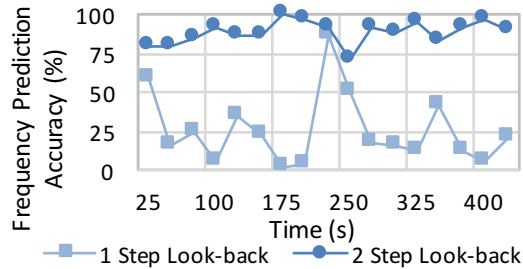


Figure 7.7: 2-Step Look-back performs better than 1-Step Look-back for loud music

Figure 7.7 shows a comparison of these two frequency prediction approaches for an 8-minute recording of loud music. Sampling frequency is adapted every 500 ms, compared with the ground truth, and the prediction accuracy is reported every 25 seconds. We observe that the 2-step look-back, i.e. the 36 state Markov model is a significant improvement over the 1-step look-back and has an average prediction accuracy of 88.23% for this very challenging case. Higher than 2-step look-back

might show a slightly better accuracy, but such a model would result in an explosion of states. Hence, we use the 2-step look-back model in SoundSifter.

The proposed Markov model evolves over time. Initially, all transition probabilities in the model are set so that the microphones are sampled at the highest rate. As the model starts to make predictions, the probabilities are updated following a simple reward and punishment process where every correct/incorrect predict is rewarded/punished by increasing/decreasing the probability. Both the processes of making a prediction and updating the probability are $\mathcal{O}(1)$ operations.

Ideally one would expect SoundSifter to adapt all its microphones to the minimum required frequency of the moment, but there is a small caveat. Because frequency prediction is not 100% accurate, there is a chance that occasionally we will have mispredictions. Cases when the predicted frequency is lower than the desired one, we will not be able to correct it in the next step unless we have an oracle. To address this, we keep the sampling frequency of one microphone fixed at 44.1 KHz, while the sampling rates of all other microphones are adapted as described above. In case of mispredictions, this microphone serves as the oracle and helps determine the correct frequency.

7.5.2 Signal Prediction and Filling

Adapting sampling rates at runtime has its benefits, but it also introduces an *alignment* problem during source separation. By default, standard source separation algorithms such as the FastICA assume that all microphones are sampled at a known fixed rate. In SoundSifter, this assumption does not hold anymore, i.e. different microphones may be sampled at different rates. Therefore, we are required to re-align samples from all the microphones and fill the gaps (missing samples) in each of the low rate microphones. Because we work with a small amount of signals at a time, although we expand the low rate samples, the process does not increase the total memory consumption significantly.

We illustrate this using two microphones: $x_i(t)$ and $x_j(t)$, supposing they are sampled at 44.1 KHz and 11.025 KHz, respectively. Within a certain period of time (e.g., between two consecutive frequency adaptation events), the first microphone will have four times more samples than the second microphone. If we align them in time, there will be three missing values in the second stream for each sample. If we assume a matrix of samples where each row corresponds to one microphone, the

resultant matrix after alignment would look like the following:

$$\begin{bmatrix} x_{i1} & x_{i2} & x_{i3} & x_{i4} & x_{i5} & x_{i6} & x_{i7} & x_{i8} & x_{i9} & \dots \\ x_{j1} & ? & ? & ? & x_{j5} & ? & ? & ? & x_{j9} & \dots \end{bmatrix}$$

To fill the missing values in the matrix, we formulate it as an interpolation problem and try to predict the missing values in two ways:

- **Linear Interpolation:** Uses line segments to join consecutive points and any intermediate missing value is predicted as a point on the line segment.
- **Cubic Spline Interpolation:** Uses piece-wise third order polynomials [202] to construct a smooth curve that is used to interpolate missing samples.

The benefit of linear interpolation is that it is faster (e.g., 58 times when compared to cubic spline for one second audio) than its higher order counterpart, but it performs very poorly in predicting audio samples if the gaps between given points are large. Cubic spline produces smoother curves and performs comparatively better for large gaps in missing values, but its running time is slower. We pick linear interpolation and a suitable length for interpolation, so that it runs fast and is also fairly accurate.

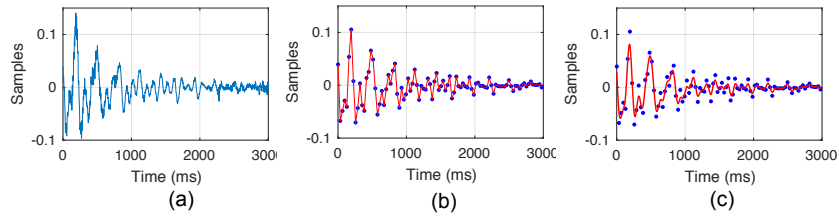


Figure 7.8: Predicting missing values of a signal (a) using linear (b) and spline (c) interpolation.

Figure 7.8 compares both linear and spline methods for signal interpolation to *raise up-sample* a signal from 1.378 KHz to 44.1 KHz. We observe that for 10 ms sample windows, linear interpolation produces faster and better results than cubic splines.

7.5.3 Modeling and Recognizing Sounds

After source separation, SoundSifter classifies each isolated source as primary or secondary. If we had sufficient training examples for each type of sound from an end user, the problem would

be as simple as creating a standard machine learning classifier. However, to reduce the burden of data collection on a user and to improve the robustness of the created classifier in a principled way, we perform an additional *data augmentation* step prior to creating the classifier model. Recall that this is a one time step that happens during the installation and programming of SoundSifter for a particular usage scenario.

Data Augmentation The basic principle behind data augmentation [203, 204] is to generate new training examples from existing ones by perturbing one or more acoustic characteristics. In SoundSifter, we use two specific types of augmentation techniques to increase the user contributed data size by about 13 times. These techniques are listed in Table 7.1.

Action	Description
<i>f</i> -warping	Remapping the frequency axis: $f_i \rightarrow \alpha f_i$, where $\alpha \in [0.8, 1.2]$.
Inject Noise	Superimposing simulated noise. Noise models are created offline.

Table 7.1: Data augmentation techniques applied to increase user-contributed samples.

To apply frequency warping [144], each sample goes through a mapping phase 10 times, each time using a different α that is chosen uniformly at random. Simulated random noise [165] is applied to about one-third of these examples to further augment the data set. Overall, we obtain a 13.33-fold boost in the number of training examples giving us original samples as well as their perturbed versions that are resilient to noise and changes in frequency due to environmental effect.

Feature Extraction and Classification For each audio frame, a 13-element MFCC feature vector is computed. Following common practice, a number of consecutive feature vectors are used to calculate 13 deltas and 13 double deltas to obtain 39-element feature vectors for each frame. Finally, the mean, standard deviation, and range of these vectors are taken to obtain a single 39-element feature vector. A random forest [205] classifier is used to create the final classifier.

For speaker identification, we extend the feature vector of the previous step to include *pitch* as an extra feature. Following [4], we estimate pitch using the *zero crossing rate* (ZCR) of the audio in the time domain:

$$ZCR = 0.5 \times \sum_{i=2}^n |sign(s_i) - sign(s_{i-1})| \quad (7.1)$$

Here, s_i represents audio samples, n is the length of a frame and $sign(x)$ is either $+1$ or -1 depending on whether $(x > 0)$ or $(x < 0)$.

7.5.4 Eliminating Secondary Residues

After source separation and source identification, we obtain a primary and a number of secondary sources. As seen previously in Figure 7.3, the residuals were not zero even in the best case, and in Figure 7.5 we observed that such residuals are significant enough to recognize secondary sources embedded within the isolated primary stream.

In order to nullify these residuals we employ a customized *adaptive noise cancellation* technique [206]. Standard noise cancellation algorithms either assume simple Gaussian noises or use sophisticated hardware to capture noise sources to ‘subtract’ noise spectra from the main audio stream. In SoundSifter, we are lucky to have the separated secondary source streams readily available as a by product of source separation. These secondary streams are used as negative feedback to remove their respective residues from the primary stream.

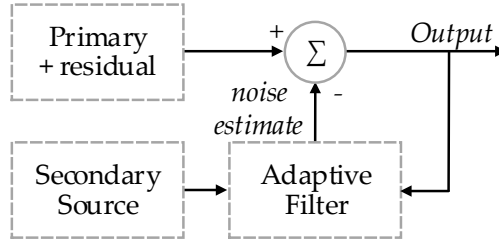


Figure 7.9: Leveraging isolated secondary sources to nullify their residue through a customized adaptive noise cancellation process.

Figure 7.9 shows the noise cancellation process where secondary sources are iteratively used to estimate residual noise signals contributed by each of them, which are then subtracted from the primary source to obtain residue-free signals. As an illustration, we consider p and n_i as the primary and secondary sources after the source separation step, respectively. An input to the noise canceller is $(p + n_i^r)$, where n_i^r denotes the residuals. The secondary source n_i is another input to the canceller that is passed through an adaptive filter to produce an output \hat{n}_i^r that is a close replica of n_i^r . The output of the adaptive filter is subtracted from the primary input to produce the system output $z = p + n_i^r - \hat{n}_i^r$, which indicates the error signal for the adaptive process.

We adjust the filter using least mean squares (LMS) [207] algorithm that minimizes the output power. We use a stochastic gradient descent method where the filter weights are adapted based on

the error at the current time step. The weight update function for the least mean squares algorithm is:

$$W_{n+1} = W_n - \mu \nabla \epsilon[n] \quad (7.2)$$

Here, ϵ represents the mean-square error, and μ is a constant that controls the speed of convergence.

7.6 Implementation Notes

Due to space limitations we only describe some key implementation issues.

7.6.1 Amazon Alexa Voice Service

To demonstrate SoundSifter, we must connect it to a home hub such as Amazon Echo or Google Home. However, at present, none of these devices come with an audio input where we can pipe in the processed audio from SoundSifter. Hence, we use Amazon Voice Service (AVS) to turn a Raspberry Pi into an Alexa-enabled device which provides us with multiple options for audio inputs, i.e. audio jacks as well as Bluetooth. We followed the step-by-step procedure [208] that Amazon recommends to enable their voice service API in Raspberry Pi platform. Figure 7.10 shows our custom home hub in a 3D printed case, which is functionally identical to an Amazon Echo device.

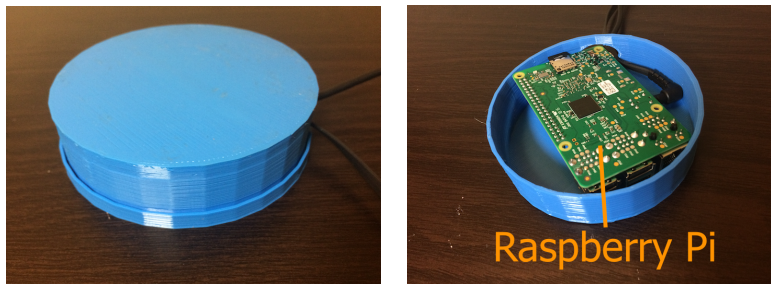


Figure 7.10: Alexa-enabled Raspberry Pi in a case.

7.6.2 SoundSifter in a BeagleBone Black

We have developed SoundSifter based on an open-source hardware platform called the Beaglebone Black [209]. An advantage of BeagleBone Black over other open platforms is that, besides the main ARM Cortex-A8 CPU, it has two additional cores known as the *programmable real-time units* (PRUs) [210]. Each PRU provides fast (200 MHz, 32-bit) real-time access to a number of I/O pins. This lets us sample multiple analog audio inputs at a very high rate.

We use a PRU enabled BeagleBone Black as SoundSifter’s processor. For the microphone array, we use four Electret microphones [211]. For aesthetics, we 3D-print a case that contains the complete

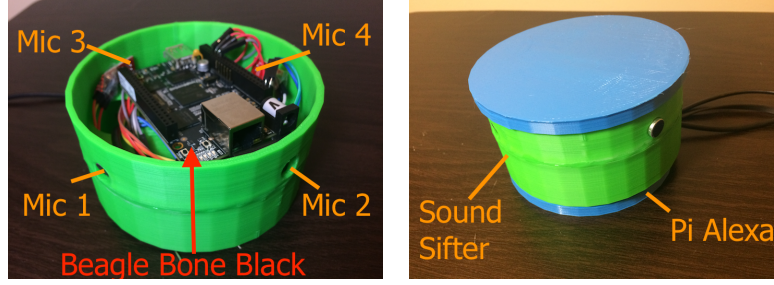


Figure 7.11: (Left) SoundSifter in its open case. (Right) the case sits on top of Alexa as a cover.

SoundSifter system. In Figure 7.11 we show the SoundSifter inside a green open case (left). The lower half of this case is hollow, which allows us to put this on top of the Alexa device of Figure 7.10 as a cover (right).

7.6.3 Libraries

For Fast ICA we used the Modular toolkit for Data Processing (MDP) [212], which is a Python-based data processing framework. We use the Pandas [213] library for interpolation. We use libpruio [214] for PRUs, which is designed for easy configuration and data handling at high speed. To use this library, we load kernel driver `uio_pruss` and enable PRU subsystems by loading the universal device tree overlay [215].

7.7 Evaluation

We perform three types of experiments. First, we evaluate the execution time, CPU, and memory usage of SoundSifter. Second, we evaluate the four key algorithms of SoundSifter using an empirical dataset. Third, we perform an end-to-end evaluation of different use cases of SoundSifter and compare the performance with an Amazon Echo device.

7.7.1 Microbenchmarks

Figure 7.12 shows execution times of five major components inside SoundSifter’s audio processing pipeline for processing one second of audio. Frequency adaptation is a constant time operation that takes only 1 ms. It is essentially a matrix look-up operation in the transition matrix. Signal alignment and filling using linear interpolation takes 35 ms on average. As expected, the most time consuming operation in SoundSifter is the source separation step that takes about 179 ms. For source identification, SoundSifter takes 121 ms to calculate the features and 3 ms for classification. The last component of SoundSifter residue removal takes 54.68 ms. Overall, SoundSifter’s execution

time is 394 ms for processing 1 second audio, which means, the system runs in real-time.

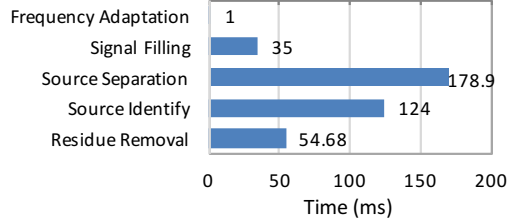


Figure 7.12: Run-time Analysis.

Table 7.2 lists the CPU and memory usage for the components of the audio processing pipeline. The most expensive operation is source separation. It requires 57.4% of the CPU and 5.2% memory. Other tasks such as source identification, signal filling, and residue removal use 50.8%, 33.6% and 11.5% of the CPU, respectively.

	CPU (%)	Memory (%)
Signal Filling	33.6	2.8
Source Separation	57.4	5.2
Source Identify	50.8	6.3
Residue Removal	11.5	2.5

Table 7.2: CPU and memory usage.

7.7.2 Algorithm Evaluation

To evaluate various components of SoundSifter, we use an empirical dataset which is segmented into three categories of sounds as described in Table 7.3. We collect this data in a 15 square feet room. For some of the analyses, such as (Figure 7.13, 7.15, 7.16, 7.18), where we want to evaluate the performance of the system at the maximum sampling rate (i.e. 44.1 KHz), we have used a multi-channel microphone setup [216]. This has allowed us to evaluate the proposed algorithms over a wider range of sampling rates. In the final prototype of SoundSifter, we, however, use Electret microphones [217] which have a frequency range of 20 Hz to 20 kHz. This is not an issue as long as the application, where SoundSifter is in use, deals with signals < 20 kHz. For example, in our chosen applications, the maximum frequency of any sound type in the environment is 10 KHz.

Frequency Adaptation We compare SoundSifter’s 2-step look-based frequency adaptation algorithm’s accuracy with that of a 1-step look back Markov model in four test scenarios. These scenarios represent: 1) a person talking, 2) two-person conversation and a TV in the background, 3)

Dataset	Count	Examples	Length
Speech	150	conversations (1-10 persons)	200 <i>min</i>
Home	54	TV, phone, mouse, keyboard	75 <i>min</i>
Song	10	rock, country	55 <i>min</i>

Table 7.3: Description of the empirical dataset.

two-person conversation with a loud song in the background, and 4) two-person conversation while both TV and loud music are playing.

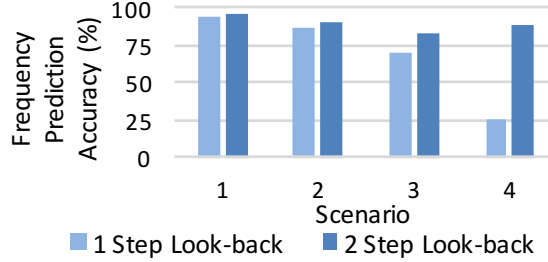


Figure 7.13: SoundSifter has higher accuracy than 1 Step Look-back for all scenarios.

In Figure 7.13, we observe that although the performance of both approaches drop as the scenarios become harder, the 2-step look-back model always performs better than the 1-step look-back method. The 2-step look-back model maintains an accuracy in the range of 88.24%-95.26%, whereas the 1-step look-back model’s accuracy drops from 94.63% to 25.18%.

A reason behind SoundSifter’s frequency adaptation was to reduce resource consumption of the embedded system. To evaluate this, we compare SoundSifter’s CPU usage with that of a BeagleBone Black that is sampling four sensors at 44.1 KHz. We consider three scenarios demonstrating three different types of acoustic environments: 1) a noisy environment where a high-volume song is playing, 2) a living room where two persons are talking and a television is running, and 3) a relatively quiet environment where two persons are talking. The ranges of frequencies in these three scenarios are 11.03–5.51 KHz, 5.51–2.76 KHz, and ≤ 2.76 KHz, respectively.

Figure 7.14 shows that as the range of frequencies of the sound sources vary, because of the adaptive nature of SoundSifter, its CPU usage decreases from 48.3%-38.1%. The CPU usage of the BeagleBone remains fixed at 55.8% at all times.

Signal Prediction and Filling We measure the performance of signal prediction and filling of SoundSifter by comparing its performance with a cubic spline-based interpolation scheme. We take

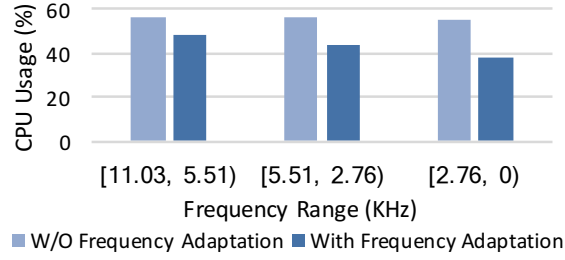


Figure 7.14: Frequency adaptation reduces CPU usage especially for lower frequency ranges.

a 44.1 KHz audio and down-sample it to produce five signal streams having the rates of 1.378 KHz, 2.756 KHz, 5.512 KHz, 11.025 KHz, 22.05 KHz, respectively. These are then up-sampled again to get back to 44.1 KHz, by using two interpolation methods: linear (as done in SoundSifter) and cubic spline. The quality of interpolation is measured in terms of their correlation with the original 44.1 KHz signals. We run this experiment on 150 speech clips of 200 minutes, 10 song clips of 50 minutes and four sound clips of 50 minutes.

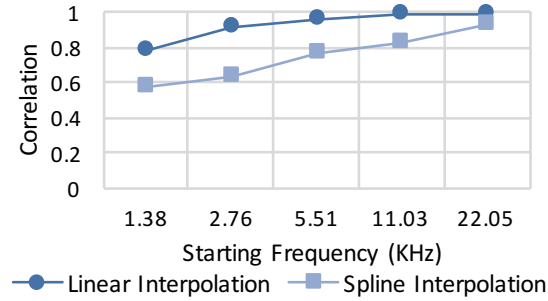


Figure 7.15: Linear Interpolation is more effective than Spline Interpolation for Signal Prediction and Filling.

Figure 7.15 shows that for 22.05 KHz, the predicted signals in SoundSifter show 98% correlation, whereas spline interpolation shows 93%. For lower frequencies, spline’s performance drops significantly. SoundSifter’s predicted signal shows more than 90% correlation even for 2.756 KHz. At 1.378 KHz, SoundSifter’ correlation drops to 79%, which is still significantly higher than that of spline interpolation.

Sound Modeling and Recognition We illustrate the performance of SoundSifter’s sound modeling and recognition with the help of two scenarios: 1) voice-only mode, and 2) personalized mode.

For the voice only mode, we test the accuracy of primary and secondary source recognition after source separation, where speech is kept as the primary source.

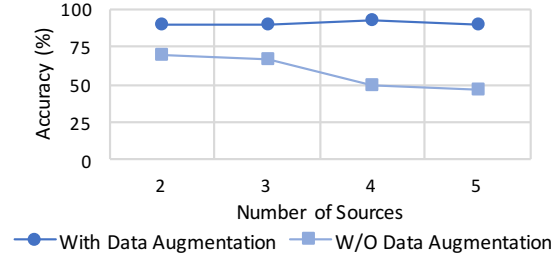


Figure 7.16: SoundSifter’s data augmentation technique improves modeling accuracy

In Figure 7.16, we observe that SoundSifter’s data augmentation technique outperforms a classifier that is trained without data augmentation. For 2-3 sources, SoundSifter shows 89%-90.14% accuracy, whereas without data augmentation, a classifier achieves only 66%-70% accuracy. As the number of secondary sources increases, we see that without data augmentation the accuracy drops below 50% for 4-5 sources. SoundSifter’s classification accuracy remains stable even when the number of sources is high. For 4-5 sources, SoundSifter achieves 89.13%-92.78% accuracy, and its false positive rate has always been 0%.

For the personalized mode, we want to recognize the particular user’s commands and ignore anything else. For this, we collect 64 voice commands from three persons as primary sources and 53 voice commands from seven persons as secondary sources. For primary speaker recognition, SoundSifter achieves 94.87% accuracy. From Figure 7.17, we see that SoundSifter is able to detect all the secondary users’ commands, resulting in a 0% false positive rates. For primary speaker detection, SoundSifter was able to detect 58 out of 64 primary commands. Hence, this shows that SoundSifter is able to accept or reject commands based on personalization.

		Predicted	
		Primary	Secondary
Actual	Primary	58	6
	Secondary	0	53

Figure 7.17: Confusion matrix for primary speaker recognition among up to 10 persons.

Residue Removal We quantify the effectiveness of residue removal by showing that the residue in the separated signals is reduced after applying the process. For this experiment, we consider audio clips from a varying number of sound sources and apply Fast ICA with and without residual removal and compare the noise residual (ξ) we get from these two approaches.

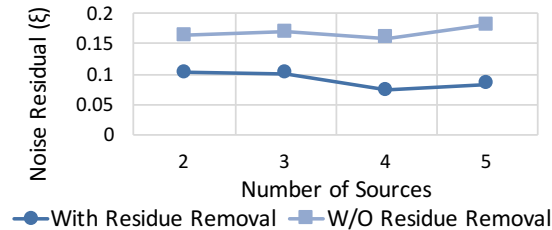


Figure 7.18: Residue Removal reduces noise residual from the primary signal.

From Figure 7.18, we see that for 2, 3, 4 and 5 sources Fast ICA without residual removal has 0.164, 0.169, 0.159 and 0.181 noise residual, whereas Fast ICA with residual removal has 0.102, 0.101, 0.073 and 0.084 noise residual, respectively. From this result, we deduce that using Fast ICA with residual removal has better performance for removing residue from the signal than using only Fast ICA.

7.7.3 Full System Evaluation

Scenarios To demonstrate the effectiveness of SoundSifter, we compare its performance with an Amazon Echo device in three different scenarios. Table 7.4 lists the scenarios.

Scenario	Participants	Noise Level
Voice Mode (normal)	15	13 dB
Voice Mode (noisy)	10	51 dB
Personalized Mode	10	13 dB

Table 7.4: Description of the scenarios.

These scenarios are designed to illustrate SoundSifter’s performance under different conditions and demands. In the first scenario, we consider a normal environment where only one person commands SoundSifter and Echo at the same time. In this experiment, we want to demonstrate that the source separation and residue removal steps do not damage the quality of the audio signals in SoundSifter. 50 commands from 15 different participants are used in this experiment.

For the second scenario, we consider a noisy environment. We ask our participants to issue commands while loud music is playing in the background. We want to show that due to source

separation and residue removal, SoundSifter is more resilient to noise than Amazon Echo. We have collected 50 commands from 10 different participants for this scenario.

In the third scenario, we want to demonstrate that SoundSifter is personalized for a particular user and an intruder is not able to issue arbitrary commands to SoundSifter. Amazon Echo does not have this feature. For this experiment, we consider one of the ten participants as the primary user and the other nine users as intruders. 50 voice commands for both the primary user and the intruders are used in this experiment.

Comparison with Amazon Echo From Figure 7.19, we see that in the first scenario, both SoundSifter and Echo are able to respond to all 50 commands. Hence, the source separation and residue removal steps of SoundSifter did not affect the quality of audio. In the second scenario, SoundSifter responds to 46 out of 50 commands, whereas Echo was only able to respond to 26 of them. This shows that SoundSifter is more resilient than Echo in a noisy environment.

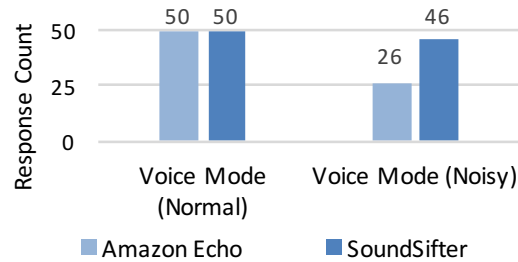


Figure 7.19: For noisy voice mode, SoundSifter was able to respond to 46 commands whereas, Echo responded to only 26 commands.

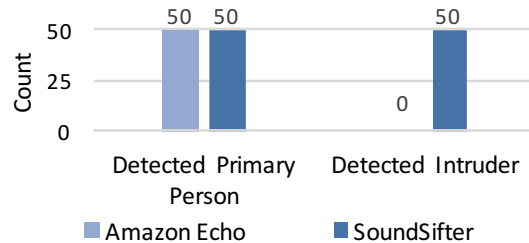


Figure 7.20: For personalized mode, SoundSifter was able to detect all 50 intruder commands, Amazon Echo fails to detect any of them.

In the third scenario, because Echo does not have any personalization support, it was unable to

detect any intruders. On the other hand, from Figure 7.20, we find that SoundSifter is able to detect all 50 of the intruder’s commands. This shows the strength of the special feature of SoundSifter that is not currently supported by any commercial home hub devices.

7.8 Discussion

At present, home hub devices like Google Home and Amazon Echo do not have any audio input port where we can pipe in processed audio from SoundSifter. This is why we had to develop a custom Amazon Echo clone using a Raspberry Pi. For our system to work seamlessly with a commercial home hub, either the device has to have an audio jack or it has to support open source software, so that our algorithms can be run on them.

In SoundSifter, the calculation of audio features takes a significant amount of computation time. Hence, the system is not fully ready for applications that require a very low latency. To address this issue, as future work, we aim to develop new acoustic features that are lightweight but similar to MFCC features in terms of performance.

The accuracy of the sound classifier depends on training, which is done by the end user who has to record audio samples of his voice commands and secondary sound types. This may be cumbersome to many users, and if the classifier is not trained sufficiently, the accuracy of primary source classification may drop. To address this issue, as a future extension, we plan to develop new classifiers that require a small number of training examples.

SoundSifter is not connected to the Internet, and audio processing happens inside the device. After processing, SoundSifter deletes the audio stream periodically. However, our system is secure as long as a hacker does not tamper with the device physically (e.g. by connecting extra wires to get raw data) or inject software Trojans (e.g., by altering the source code to dump the memory to files or by disabling/bypassing the algorithms during processing).

7.9 Related Work

Recent works have explored the security issues related to continuous listening devices. In [218] the authors present an attack mechanism related to Google voice search by injecting audio files with malicious audio commands. [219] shows the vulnerability of smart assistants with inaudible commands from long distance. Similarly [220, 221] proposes inaudible and malicious voice commands to attack voice assistants. [222] proposes an external hardware to match the vibration between the

body and speech signal to detect authenticated user. [223] proposes a generic scheme that intervenes the user and requires manual review for each potential voice command. [224] explores the sensitivity of voice command on the security privacy of the personal assistants. In this chapter, we are the first to explore the privacy issues related to the contextual information leakage through the voice commands.

Solutions to the source separation problem under different assumptions on sources and mixing systems fall into three main categories. The first category is based on *Independent Component Analysis* (ICA) [46], where statistical independence of sources is exploited to estimate the sources. Although efficient implementations exist [48, 225], these algorithms have certain fundamental limitations, e.g., they support at most one Gaussian source, and do not exploit signal properties such as non-negativity or sparsity. The second category applies *Non-negative Matrix Factorization* (NMF) [39] to exploit non-negativity of real-world signals. However, their lack of statistical assumptions on data does not guarantee a correct decomposition of sources. The third type is based on *Sparse Component Analysis* (SCA) [47], which exploits sparsity in signals, e.g., acoustic signals. In this chapter, we could have used any of the three methods, but we implement FastICA [225, 48] which is a proven and well-used algorithm.

In many digital systems, dynamic voltage and frequency scaling [226, 227, 228, 229, 230] are applied to scale up/down the clock frequency of the entire system to lower the power consumption. Our problem is significantly different and harder than that since we are required to dynamically adapt sampling frequencies of an array of microphones (each one is assigned a different rate) and make sure that the independent component analysis framework for source separation still works.

Standard practices toward dealing with limited training data fall broadly into two categories – *data augmentation* and *classifier fusion*, which are often used together. Data augmentation techniques [203, 204] generate new training examples by perturbing acoustic characteristics of existing examples, e.g., frequency warping [144], modifying tempo [164], and adding simulated reverberation [145], and noise [165]. In classifier fusion [146, 166], outputs of multiple classifiers are combined to improve the overall accuracy. These techniques vary from simple voting and averaging [167], to ranking [231, 168], to learning decision templates [148, 168]. In this chapter, we only use data augmentation and avoided classifier fusion since that would require sharing and exchanging classifier models of different users – which may violate their privacy.

7.10 Summary

This chapter describes a system that mitigates information leakage due to the presence of unwanted sound sources in an acoustic environment when using voice-enabled personal assistant devices like Amazon Echo. We developed new algorithms to make acoustic source separation CPU and memory efficient, and to remove residuals of unwanted signals from the main audio stream. The performance of the system has been compared with that of commercial continuous listening devices to show that it accurately filters out unwanted sounds and thus protects against personal and contextual information leakage where existing devices fail.

CHAPTER 8

Conclusion and Future Works

8.1 Summary of the Dissertation

In this thesis, we address the effect of sensor and environment heterogeneity on machine learning based classifier for acoustic and WiFi signal. We also explore the problem of lack of labeled examples in the context of acoustic event recognition and WiFi-based activity recognition. To this end, we propose an unsupervised domain discovery algorithm to find the latent domains (sensors, environment) for any general purpose timeseries signal. Then, we also propose a multi-source to multi-target domain adaptation algorithm to remove the domain-induced bias from the neural network architecture. We implement the proposed algorithms in a system named Sound-Adapter and evaluate the proposed algorithms for acoustic event recognition. For WiFi-based activity recognition, we propose a one-shot domain adaptation method WiDeo to adapt a neural network based classifier trained on one environment to a new environment with only one labeled sample per activity class. To solve the problem of lack of labeled example, we propose two systems Sound-Semantics and WiFringe for audio and WiFi-based sensing, respectively, where we leverage knowledge from external modality to classify classes with no labeled examples. We show for the first time that by utilizing the state-of-the-art semantic representation of labels and verb attributes learned from word’s definition, we can augment the sensing capability of acoustic and WiFi-based sensing systems that lack adequate training examples. To evaluate our proposed algorithms, we collect data from real scenario with multiple volunteers and present an elaborate sets of experiments. Finally, we study the privacy issues of continuous listening device and propose a privacy preserving acoustic sensing system SoundSifter to mitigate the contextual information leakage. We developed algorithms to make acoustic source separation CPU and memory efficient, and to remove residuals of unwanted signals from the main audio stream to preserve the privacy of the users.

8.2 Future Directions

- **Active Learning:** In this thesis, we present methodologies to integrate knowledge from external domain into acoustic and WiFi based classifiers. A potential approach for these sensing systems to continuously learn is active learning, where the learning algorithm iteratively asks the user or developer for label for a subset of the unlabeled data. Active learning in combination with our proposed zero shot learning framework will pave the way for lifelong learning sensing systems.
- **Multi-modal Sensing:** While we propose acoustic and WiFi sensing in separate systems, the extension to my thesis is to integrate the two sensing modality in an end-to-end system. Multi-modal input from both acoustic and WiFi domain will make a robust and more effective sensing system.
- **Multi-person WiFi Sensing:** Our proposed WiFi based activity recognition algorithms are trained with training data collected from only a single person in an environment. To the best of my knowledge, currently there is no work in the literature that deals with multi-person activity recognition. To make WiFi based activity recognition system more feasible, it is imperative to make these models compatible for multi-person activity scenario. In this case, a large scale data collection with multi-person activities and signal separation models need to be explored.

APPENDIX A

WIFI CSI COLLECTION

A.1 CSI Data Capture

In this section, we briefly give a overview of the data collection method for WiFi CSI signal. We use the tools provided by Halperin et.al. [125]. The tool is based on Intel Wi-Fi Wireless Link 5300 802.11n MIMO radios. The CSI tool provides channel matrices for 30 subcarrier groups, one group for every 2 subcarriers at 20 MHz or one group in 4 at 40 MHz. The CSI tool is installed following the instructions described in [232].

As the network manager is required to be disabled we connect to access point using the following commands:

```
sudo modprobe -r iwldvm iwlwifi mac80211
sudo modprobe iwlwifi connector_log=0x1

sudo ifconfig wlan0 down
sudo ifconfig wlan0 up

sudo iwconfig wlan0 essid wideo #replace the AP name
sudo dhclient wlan0
```

Then we use the following commans for logging CSI value:

```
sudo linux-80211n-csitool-supplementary/netlink/log_to_file csi.dat
```

A.2 CSI Data Processing

The code for extracting CSI data from binary file into a timeseries data and converting to frequency domain is provided in [233].

A.3 Word Embedding

We use the word embedding from [234] and verb attributes from [235].

APPENDIX B

AUDIO PROCESSING

B.1 Synchronous Audio Recording

For synchronous audio collection, we use a network-based data collection setup using MQTT [236] as the message broker. The idea is to use a laptop as the MQTT server and make it (or an external speaker) play the audio files in sequence. Simultaneously, we send a message to all the clients (e.g., raspberry pi, smartphone) to record the audio file being played. We open-source the necessary codes in [237].

B.2 Audio Data Processing

We use librosa [238]- an open source library developed for Python for audio signal processing tasks i.e., fourier transform, spectrogram extraction and so on.

B.3 Source Separation

An implementation of fast-ICA for source separation is available in [239].

B.4 Gradient Reversal

Tensorflow code for gradient reversal is in [240].

B.5 High Sampling Rate

We use libpruio [241] for PRUs, which is designed for easy configuration and data handling at high speed. To use this library, we load kernel driver uio_pruss and enable PRU subsystems by loading the universal device tree overlay [215].

REFERENCES

- [1] “iOS Siri from Apple.” <http://www.apple.com/ios/siri/>.
- [2] M. T. Islam, B. Islam, and S. Nirjon, “Soundsifter: Mitigating overhearing of continuous listening devices,” in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 29–41, ACM, 2017.
- [3] P. Lamere, P. Kwok, E. Gouvea, B. Raj, R. Singh, W. Walker, M. Warmuth, and P. Wolf, “The cmu sphinx-4 speech recognition system,” in *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2003), Hong Kong*, vol. 1, pp. 2–5, 2003.
- [4] H. Lu, A. B. Brush, B. Priyantha, A. K. Karlson, and J. Liu, “Speakersense: Energy efficient unobtrusive speaker identification on mobile phones,” in *International conference on pervasive computing*, pp. 188–205, Springer, 2011.
- [5] A. Wang, “The shazam music recognition service,” *Communications of the ACM*, vol. 49, no. 8, pp. 44–48, 2006.
- [6] Z. Chen, M. Lin, F. Chen, N. D. Lane, G. Cardone, R. Wang, T. Li, Y. Chen, T. Choudhury, and A. T. Campbell, “Unobtrusive sleep monitoring using smartphones,” in *Proceedings of the 7th International Conference on Pervasive Computing Technologies for Healthcare*, pp. 145–152, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013.
- [7] C. Clavel, T. Ehrette, and G. Richard, “Events detection for an audio-based surveillance system,” in *2005 IEEE International Conference on Multimedia and Expo*, pp. 1306–1309, IEEE, 2005.
- [8] B. U. Töreyn, Y. Dedeoğlu, and A. E. Çetin, “Hmm based falling person detection using both audio and video,” in *International Workshop on Human-Computer Interaction*, pp. 211–220, Springer, 2005.
- [9] J. Chen, A. H. Kam, J. Zhang, N. Liu, and L. Shue, “Bathroom activity monitoring based on sound,” in *International Conference on Pervasive Computing*, pp. 47–61, Springer, 2005.
- [10] S. Nirjon, R. F. Dickerson, Q. Li, P. Asare, J. A. Stankovic, D. Hong, B. Zhang, X. Jiang, G. Shen, and F. Zhao, “Musicalheart: A hearty way of listening to music,” in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pp. 43–56, ACM, 2012.
- [11] H.-K. Ra, A. Salekin, H.-J. Yoon, J. Kim, S. S. Nirjon, D. J. Stone, S. Kim, J.-M. Lee, S. H. Son, J. A. Stankovic, *et al.*, “Asthmaguide: an asthma monitoring and advice ecosystem,” in *Wireless Health*, pp. 128–135, 2016.
- [12] A. Mathur, T. Zhang, S. Bhattacharya, P. Veličković, L. Joffe, N. D. Lane, F. Kawsar, and P. Lió, “Using deep data augmentation training to address software and hardware heterogeneities in wearable and smartphone sensing devices,” in *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 200–211, IEEE Press, 2018.
- [13] A. Mathur, A. Isopoüssu, F. Kawsar, N. Berthouze, and N. D. Lane, “Mic2mic: using cycle-consistent generative adversarial networks to overcome microphone variability in speech

- systems,” in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, pp. 169–180, ACM, 2019.
- [14] J. Hoffman, B. Kulis, T. Darrell, and K. Saenko, “Discovering latent domains for multisource domain adaptation,” in *European Conference on Computer Vision*, pp. 702–715, Springer, 2012.
 - [15] M. Mancini, L. Porzi, S. Rota Bulò, B. Caputo, and E. Ricci, “Boosting domain adaptation by discovering latent domains,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3771–3780, 2018.
 - [16] Y. Wang, J. Liu, Y. Chen, M. Gruteser, J. Yang, and H. Liu, “E-eyes: device-free location-oriented activity identification using fine-grained wifi signatures,” in *Proceedings of the 20th annual international conference on Mobile computing and networking*, pp. 617–628, ACM, 2014.
 - [17] X. Liu, J. Cao, S. Tang, and J. Wen, “Wi-sleep: Contactless sleep monitoring via wifi signals,” in *Real-Time Systems Symposium (RTSS), 2014 IEEE*, pp. 346–355, IEEE, 2014.
 - [18] H. Abdelnasser, M. Youssef, and K. A. Harras, “Wigest: A ubiquitous wifi-based gesture recognition system,” in *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pp. 1472–1480, IEEE, 2015.
 - [19] F. Adib and D. Katabi, *See through walls with WiFi!*, vol. 43. ACM, 2013.
 - [20] X. Liu, J. Cao, S. Tang, J. Wen, and P. Guo, “Contactless respiration monitoring via off-the-shelf wifi devices,” *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2466–2479, 2016.
 - [21] S. Nirjon, R. F. Dickerson, P. Asare, Q. Li, D. Hong, J. A. Stankovic, P. Hu, G. Shen, and X. Jiang, “Auditeur: A mobile-cloud service platform for acoustic event detection on smartphones,” in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pp. 403–416, ACM, 2013.
 - [22] Y. Wang, X. Jiang, R. Cao, and X. Wang, “Robust indoor human activity recognition using wireless signals,” *Sensors*, vol. 15, no. 7, pp. 17195–17208, 2015.
 - [23] S. Yue, H. He, H. Wang, H. Rahul, and D. Katabi, “Extracting multi-person respiration from entangled rf signals,” *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 2, p. 86, 2018.
 - [24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.
 - [25] “Intel ultimate n wifi link 5300.” <https://www.intel.com/content/www/us/en/wireless-products/ultimate-n-wifi-link-5300-brief.html>.
 - [26] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in neural information processing systems*, pp. 3111–3119, 2013.
 - [27] J. R. Firth, “A synopsis of linguistic theory, 1930-1955,” *Studies in linguistic analysis*, 1957.
 - [28] B. Wu, “An introduction to neural networks and their applications in manufacturing,” *Journal of Intelligent Manufacturing*, vol. 3, no. 6, pp. 391–403, 1992.

- [29] M. Partio, B. Cramariuc, M. Gabbouj, and A. Visa, “Rock texture retrieval using gray level co-occurrence matrix,” in *Proc. of 5th Nordic Signal Processing Symposium*, vol. 75, Citeseer, 2002.
- [30] J. Turian, L. Ratinov, and Y. Bengio, “Word representations: a simple and general method for semi-supervised learning,” in *Proceedings of the 48th annual meeting of the association for computational linguistics*, pp. 384–394, Association for Computational Linguistics, 2010.
- [31] R. Zellers and Y. Choi, “Zero-shot activity recognition with verb attribute induction,” *arXiv preprint arXiv:1707.09468*, 2017.
- [32] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.
- [33] J. Herault and C. Jutten, “Space or time adaptive signal processing by neural network models,” in *Neural networks for computing*, vol. 151, pp. 206–211, AIP Publishing, 1986.
- [34] T.-P. Jung, S. Makeig, C. Humphries, T.-W. Lee, M. J. Mckeown, V. Iragui, and T. J. Sejnowski, “Removing electroencephalographic artifacts by blind source separation,” *Psychophysiology*, vol. 37, no. 02, pp. 163–178, 2000.
- [35] G. Srivastava, S. Crottaz-Herbette, K. Lau, G. Glover, and V. Menon, “Ica-based procedures for removing ballistocardiogram artifacts from eeg data acquired in the mri scanner,” *Neuroimage*, vol. 24, no. 1, pp. 50–60, 2005.
- [36] K. I. Diamantaras and T. Papadimitriou, “Mimo blind deconvolution using subspace-based filter deflation,” in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP’04). IEEE International Conference on*, vol. 4, pp. iv–433, IEEE, 2004.
- [37] S. Cruces-Alvarez, A. Cichocki, and L. Castedo-Ribas, “An iterative inversion approach to blind source separation,” *IEEE Transactions on Neural Networks*, vol. 11, no. 6, pp. 1423–1437, 2000.
- [38] S.-M. Cha and L.-W. Chan, “Applying independent component analysis to factor model in finance,” in *International Conference on Intelligent Data Engineering and Automated Learning*, pp. 538–544, Springer, 2000.
- [39] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.
- [40] Q.-H. Lin, F.-L. Yin, T.-M. Mei, and H. Liang, “A blind source separation based method for speech encryption,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 6, pp. 1320–1328, 2006.
- [41] S. Li, C. Li, K.-T. Lo, and G. Chen, “Cryptanalyzing an encryption scheme based on blind source separation,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 4, pp. 1055–1063, 2008.
- [42] E. Vincent, R. Gribonval, and C. Févotte, “Performance measurement in blind audio source separation,” *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

- [43] S. Haykin and Z. Chen, “The cocktail party problem,” *Neural computation*, vol. 17, no. 9, pp. 1875–1902, 2005.
- [44] M. Zibulevsky and B. A. Pearlmutter, “Blind source separation by sparse decomposition in a signal dictionary,” *Neural computation*, vol. 13, no. 4, pp. 863–882, 2001.
- [45] P. Bofill and M. Zibulevsky, “Underdetermined blind source separation using sparse representations,” *Signal processing*, vol. 81, no. 11, pp. 2353–2362, 2001.
- [46] A. Hyvärinen and E. Oja, “Independent component analysis: algorithms and applications,” *Neural networks*, vol. 13, no. 4, pp. 411–430, 2000.
- [47] R. Gribonval and S. Lesage, “A survey of sparse component analysis for blind source separation: principles, perspectives, and new challenges,” in *ESANN’06 proceedings-14th European Symposium on Artificial Neural Networks*, pp. 323–330, d-side publi., 2006.
- [48] A. Hyvarinen, “Fast and robust fixed-point algorithms for independent component analysis,” *IEEE transactions on Neural Networks*, vol. 10, no. 3, pp. 626–634, 1999.
- [49] W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative study of cnn and rnn for natural language processing,” *arXiv preprint arXiv:1702.01923*, 2017.
- [50] I. V. Tetko, P. Karpov, R. Van Deursen, and G. Godin, “State-of-the-art augmented nlp transformer models for direct and single-step retrosynthesis,” *Nature communications*, vol. 11, no. 1, pp. 1–11, 2020.
- [51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *arXiv preprint arXiv:1706.03762*, 2017.
- [52] S. Pradhan, W. Sun, G. Baig, and L. Qiu, “Combating replay attacks against voice assistants,” *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 3, pp. 1–26, 2019.
- [53] “The Complete Guide to Hearable Technology in 2019.” <https://www.everydayhearing.com/hearing-technology/articles/hearables/>.
- [54] “Hearables Are Growing Fast.” <https://voicebot.ai/2019/09/13/idc-says-hearables-are-now-biggest-wearables-segment-and-growing-fast/>.
- [55] N. Bui, N. Pham, J. J. Barnitz, Z. Zou, P. Nguyen, H. Truong, T. Kim, N. Farrow, A. Nguyen, J. Xiao, *et al.*, “ebp: A wearable system for frequent and comfortable blood pressure monitoring from user’s ear,” in *The 25th Annual International Conference on Mobile Computing and Networking*, pp. 1–17, 2019.
- [56] A. Wang and S. Gollakota, “Millisonic: Pushing the limits of acoustic motion tracking,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–11, 2019.
- [57] G. Laput, K. Ahuja, M. Goel, and C. Harrison, “Ubioustics: Plug-and-play acoustic activity recognition,” in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, pp. 213–224, 2018.
- [58] I. Hwang, S. Liu, E. J. Rozner, and C. N. Sze, “Determining a behavior of a user utilizing audio data,” May 3 2018. US Patent App. 15/336,614.

- [59] R. Nandakumar, S. Gollakota, and N. Watson, “Contactless sleep apnea detection on smartphones,” in *Proceedings of the 13th annual international conference on mobile systems, applications, and services*, pp. 45–57, ACM, 2015.
- [60] J. Xie, S. Jiang, W. Xie, and X. Gao, “An efficient global k-means clustering algorithm,” 2011.
- [61] Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation,” *arXiv preprint arXiv:1409.7495*, 2014.
- [62] A. Mesaros, T. Heittola, and T. Virtanen, “A multi-device dataset for urban acoustic scene classification,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, pp. 9–13, November 2018.
- [63] A. Das, N. Borisov, and M. Caesar, “Do you hear what i hear?: Fingerprinting smart devices through embedded acoustic components,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 441–452, ACM, 2014.
- [64] N. Roy, H. Hassanieh, and R. Roy Choudhury, “Backdoor: Making microphones hear inaudible sounds,” in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 2–14, 2017.
- [65] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, “Mobile device identification via sensor fingerprinting,” *arXiv preprint arXiv:1408.1416*, 2014.
- [66] G. Gokul, Y. Yan, K. Dantu, S. Y. Ko, and L. Ziarek, “Real time sound processing on android,” in *Proceedings of the 14th International Workshop on Java Technologies for Real-Time and Embedded Systems*, pp. 1–10, 2016.
- [67] K. J. Piczak, “ESC: Dataset for Environmental Sound Classification,” in *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pp. 1015–1018, ACM Press.
- [68] H. B. Sailor, D. M. Agrawal, and H. A. Patil, “Unsupervised filterbank learning using convolutional restricted boltzmann machine for environmental sound classification,” 2017.
- [69] B. E. Kingsbury, N. Morgan, and S. Greenberg, “Robust speech recognition using the modulation spectrogram,” *Speech communication*, vol. 25, no. 1-3, pp. 117–132, 1998.
- [70] S. Nirjon, R. F. Dickerson, P. Asare, Q. Li, D. Hong, J. A. Stankovic, P. Hu, G. Shen, and X. Jiang, “Auditeur: A mobile-cloud service platform for acoustic event detection on smartphones,” in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pp. 403–416, 2013.
- [71] M. Almeida, M. Bilal, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Varvello, and J. Blackburn, “Chimp: Crowdsourcing human inputs for mobile phones,” in *Proceedings of the 2018 World Wide Web Conference*, pp. 45–54, 2018.
- [72] S. Ruder, “An overview of multi-task learning in deep neural networks,” *arXiv preprint arXiv:1706.05098*, 2017.
- [73] L. Deng, M. L. Seltzer, D. Yu, A. Acero, A.-r. Mohamed, and G. Hinton, “Binary coding of speech spectrograms using a deep auto-encoder,” in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

- [74] M. T. Islam and S. Nirjon, "Soundsemantics: exploiting semantic knowledge in text for embedded acoustic event classification," in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, pp. 217–228, ACM, 2019.
- [75] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [76] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [77] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, pp. 478–487, 2016.
- [78] S. Łukasik, P. A. Kowalski, M. Charytanowicz, and P. Kulczycki, "Clustering using flower pollination algorithm and calinski-harabasz index," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2724–2728, IEEE, 2016.
- [79] S. Aranganayagi and K. Thangavel, "Clustering categorical data using silhouette coefficient as a relocating measure," in *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, vol. 2, pp. 13–17, IEEE, 2007.
- [80] D. de Godoy, B. Islam, S. Xia, M. T. Islam, R. Chandrasekaran, Y.-C. Chen, S. Nirjon, P. R. Kinget, and X. Jiang, "Paws: A wearable acoustic system for pedestrian safety," in *Internet-of-Things Design and Implementation (IoTDI), 2018 IEEE/ACM Third International Conference on*, pp. 237–248, IEEE, 2018.
- [81] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability of co-training," in *Proceedings of the ninth international conference on Information and knowledge management*, pp. 86–93, 2000.
- [82] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.
- [83] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, *et al.*, "Cnn architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 131–135, IEEE, 2017.
- [84] A. Chowdhery, P. Warden, J. Shlens, A. Howard, and R. Rhodes, "Visual wake words dataset," *arXiv preprint arXiv:1906.05721*, 2019.
- [85] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.
- [86] "Raspberry Pi Model 3." <https://www.raspberrypi.org/products/raspberry-pi-3-model-a-plus/>.
- [87] "Lenovo IdeaPad." <https://www.lenovo.com/us/en/laptops/thinkpad>.
- [88] "USB Mic 3." <https://www.amazon.com/dp/B07N2WRHMY>.
- [89] "USB Mic 2." <https://www.amazon.com/dp/B01142EP0>.
- [90] "USB Mic 1." <https://www.amazon.com/dp/B00UZY2YQE>.

- [91] “USB Mic 4.” <https://www.amazon.com/dp/B0028Y4DCC>.
- [92] “Matrix Voice Microphone.” <https://www.matrix.one/products/voice>.
- [93] H. Zha, X. He, C. Ding, M. Gu, and H. D. Simon, “Spectral relaxation for k-means clustering,” in *Advances in neural information processing systems*, pp. 1057–1064.
- [94] H. Wu, J. Feng, X. Tian, F. Xu, Y. Liu, X. Wang, and S. Zhong, “secgan: A cycle-consistent gan for securely-recoverable video transformation,” in *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges*, pp. 33–38, 2019.
- [95] B. Gong, Y. Shi, F. Sha, and K. Grauman, “Geodesic flow kernel for unsupervised domain adaptation,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2066–2073, IEEE, 2012.
- [96] M. Mancini, L. Porzi, S. R. Bulò, B. Caputo, and E. Ricci, “Inferring latent domains for unsupervised deep domain adaptation,” *IEEE transactions on pattern analysis and machine intelligence*, 2019.
- [97] Y. Aytar, C. Vondrick, and A. Torralba, “Soundnet: Learning sound representations from unlabeled video,” in *Advances in Neural Information Processing Systems*, pp. 892–900, 2016.
- [98] B. Gong, K. Grauman, and F. Sha, “Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation,” in *International Conference on Machine Learning*, pp. 222–230, 2013.
- [99] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. J. Smola, “Correcting sample selection bias by unlabeled data,” in *Advances in neural information processing systems*, pp. 601–608, 2007.
- [100] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, “Domain separation networks,” in *Advances in neural information processing systems*, pp. 343–351, 2016.
- [101] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, “Cycada: Cycle-consistent adversarial domain adaptation,” *arXiv preprint arXiv:1711.03213*, 2017.
- [102] A. Kurtz, H. Gascon, T. Becker, K. Rieck, and F. Freiling, “Fingerprinting mobile devices using personalized configurations,” *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 1, pp. 4–19, 2016.
- [103] B. Gong, K. Grauman, and F. Sha, “Reshaping visual datasets for domain adaptation,” in *Advances in Neural Information Processing Systems*, pp. 1286–1294, 2013.
- [104] R. Van Noorden, “The ethical questions that haunt facial-recognition research,” *Nature*, vol. 587, no. 7834, pp. 354–358, 2020.
- [105] C. Tong, S. A. Taylor, and N. D. Lane, “Are accelerometers for activity recognition a dead-end?,” in *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications*, pp. 39–44, 2020.
- [106] Z. Chen, L. Zhang, C. Jiang, Z. Cao, and W. Cui, “Wifi csi based passive human activity recognition using attention based blstm,” *IEEE Transactions on Mobile Computing*, 2018.

- [107] F. Wang, W. Gong, and J. Liu, "On spatial diversity in wifi-based human activity recognition: A deep learning based approach," *IEEE Internet of Things Journal*, 2018.
- [108] J. Ma, H. Wang, D. Zhang, Y. Wang, and Y. Wang, "A survey on wi-fi based contactless activity recognition," in *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCCom/IoP/SmartWorld), 2016 Intl IEEE Conferences*, pp. 1086–1091, IEEE, 2016.
- [109] W. Jiang, C. Miao, F. Ma, S. Yao, Y. Wang, Y. Yuan, H. Xue, C. Song, X. Ma, D. Koutsonikolas, *et al.*, "Towards environment independent device free human activity recognition," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pp. 289–304, ACM, 2018.
- [110] Y. Zheng, Y. Zhang, K. Qian, G. Zhang, Y. Liu, C. Wu, and Z. Yang, "Zero-effort cross-domain gesture recognition with wi-fi," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 313–325, ACM, 2019.
- [111] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
- [112] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *CVPR*, 2017.
- [113] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *CVPR*, 2017.
- [114] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.
- [115] I. Tenney, D. Das, and E. Pavlick, "Bert rediscovers the classical nlp pipeline," *arXiv preprint arXiv:1905.05950*, 2019.
- [116] Z. Gao, A. Feng, X. Song, and X. Wu, "Target-dependent sentiment classification with bert," *IEEE Access*, vol. 7, pp. 154290–154299, 2019.
- [117] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.
- [118] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *International conference on rough sets and knowledge technology*, pp. 364–375, Springer, 2014.
- [119] A. Virmani and M. Shahzad, "Position and orientation agnostic gesture recognition using wifi," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 252–264, ACM, 2017.
- [120] W. Wang, A. X. Liu, M. Shahzad, K. Ling, and S. Lu, "Understanding and modeling of wifi signal based human activity recognition," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pp. 65–76, ACM, 2015.
- [121] W.-k. Lu and Q. Zhang, "Deconvolutive short-time fourier transform spectrogram," *IEEE Signal Processing Letters*, vol. 16, no. 7, pp. 576–579, 2009.

- [122] R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality reduction by learning an invariant mapping,” in *null*, pp. 1735–1742, IEEE, 2006.
- [123] G. Koch, “Siamese neural networks for one-shot image recognition,” 2015.
- [124] “Intel nuc mini pc.” <https://www.intel.com/content/www/us/en/products/boards-kits/nuc.html>.
- [125] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, “Tool release: Gathering 802.11 n traces with channel state information,” *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, pp. 53–53, 2011.
- [126] I. W. Selesnick and C. S. Burrus, “Generalized digital butterworth filter design,” *IEEE Transactions on signal processing*, vol. 46, no. 6, pp. 1688–1694, 1998.
- [127] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, “Activity recognition using cell phone accelerometers,” *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.
- [128] S. Sigg, M. Scholz, S. Shi, Y. Ji, and M. Beigl, “Rf-sensing of activities from non-cooperative subjects in device-free recognition systems using ambient and local signals,” *IEEE Transactions on Mobile Computing*, vol. 13, no. 4, pp. 907–920, 2014.
- [129] S. Sigg, U. Blanke, and G. Troster, “The telepathic phone: Frictionless activity recognition from wifi-rssi,” in *Pervasive Computing and Communications (PerCom), 2014 IEEE International Conference on*, pp. 148–155, IEEE, 2014.
- [130] A. E. Kosba, A. Saeed, and M. Youssef, “Rasid: A robust wlan device-free passive motion detection system,” in *Pervasive computing and communications (PerCom), 2012 IEEE international conference on*, pp. 180–189, IEEE, 2012.
- [131] S. Sigg, S. Shi, and Y. Ji, “Rf-based device-free recognition of simultaneously conducted activities,” in *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pp. 531–540, ACM, 2013.
- [132] G. Malysa, D. Wang, L. Netsch, and M. Ali, “Hidden markov model-based gesture recognition with fmcw radar,” in *Signal and Information Processing (GlobalSIP), 2016 IEEE Global Conference on*, pp. 1017–1021, IEEE, 2016.
- [133] M. Zhao, F. Adib, and D. Katabi, “Emotion recognition using wireless signals,” in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pp. 95–108, ACM, 2016.
- [134] K. Ramasubramanian, “Using a complex-baseband architecture in fmcw radar systems,”
- [135] F. Adib, H. Mao, Z. Kabelac, D. Katabi, and R. C. Miller, “Smart homes that monitor breathing and heart rate,” in *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pp. 837–846, ACM, 2015.
- [136] Q. Pu, S. Gupta, S. Gollakota, and S. Patel, “Whole-home gesture recognition using wireless signals,” in *Proceedings of the 19th annual international conference on Mobile computing & networking*, pp. 27–38, ACM, 2013.
- [137] Y. Wang, K. Wu, and L. M. Ni, “Wifall: Device-free fall detection by wireless networks,” *IEEE Transactions on Mobile Computing*, vol. 16, no. 2, pp. 581–594, 2017.

- [138] G. Wang, Y. Zou, Z. Zhou, K. Wu, and L. M. Ni, “We can hear you with wi-fi!,” *IEEE Transactions on Mobile Computing*, vol. 15, no. 11, pp. 2907–2920, 2016.
- [139] W. Xi, J. Zhao, X.-Y. Li, K. Zhao, S. Tang, X. Liu, and Z. Jiang, “Electronic frog eye: Counting crowd using wifi,” in *Infocom, 2014 proceedings ieee*, pp. 361–369, IEEE, 2014.
- [140] H. Li, W. Yang, J. Wang, Y. Xu, and L. Huang, “Wifinger: talk to your smart devices with finger-grained gesture,” in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 250–261, ACM, 2016.
- [141] W. Wang, A. X. Liu, and M. Shahzad, “Gait recognition using wifi signals,” in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 363–373, ACM, 2016.
- [142] R. H. Venkatnarayan, G. Page, and M. Shahzad, “Multi-user gesture recognition using wifi,” in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 401–413, ACM, 2018.
- [143] H. Zou, Y. Zhou, J. Yang, H. Jiang, L. Xie, and C. J. Spanos, “Deepsense: Device-free human activity recognition via autoencoder long-term recurrent convolutional network,” in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2018.
- [144] N. Jaitly and G. E. Hinton, “Vocal tract length perturbation (vtlp) improves speech recognition,” in *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language*, 2013.
- [145] R. F. Dickerson, E. Hoque, P. Asare, S. Nirjon, and J. A. Stankovic, “Resonate: reverberation environment simulation for improved classification of speech models,” in *Proceedings of the 13th international symposium on Information processing in sensor networks*, pp. 107–118, IEEE Press, 2014.
- [146] D. Ruta and B. Gabrys, “An overview of classifier fusion methods,” *Computing and Information systems*, vol. 7, no. 1, pp. 1–10, 2000.
- [147] M. A. Tanner and W. H. Wong, “The calculation of posterior distributions by data augmentation,” *Journal of the American statistical Association*, vol. 82, no. 398, pp. 528–540, 1987.
- [148] L. I. Kuncheva, J. C. Bezdek, and R. P. Duin, “Decision templates for multiple classifier fusion: an experimental comparison,” *Pattern recognition*, vol. 34, no. 2, pp. 299–314, 2001.
- [149] C. H. Lampert, H. Nickisch, and S. Harmeling, “Attribute-based classification for zero-shot visual object categorization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 3, pp. 453–465, 2014.
- [150] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, *et al.*, “Devise: A deep visual-semantic embedding model,” in *Advances in neural information processing systems*, pp. 2121–2129, 2013.
- [151] N. Japkowicz and S. Stephen, “The class imbalance problem: A systematic study,” *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.
- [152] M. A. Guvensan, Z. C. Taysi, and T. Melodia, “Energy monitoring in residential spaces with audio sensor nodes: Tinyyears,” *Ad Hoc Networks*, vol. 11, no. 5, pp. 1539–1555, 2013.

- [153] G. Tzanetakis and P. Cook, “Musical genre classification of audio signals,” *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.
- [154] D. T. Blumstein, D. J. Mennill, P. Clemins, L. Girod, K. Yao, G. Patricelli, J. L. Deppe, A. H. Krakauer, C. Clark, K. A. Cortopassi, *et al.*, “Acoustic monitoring in terrestrial environments using microphone arrays: applications, technological considerations and prospectus,” *Journal of Applied Ecology*, vol. 48, no. 3, pp. 758–767, 2011.
- [155] Y. Dedeoglu, B. U. Toreyin, U. Gudukbay, and A. E. Cetin, “Surveillance using both video and audio,” in *Multimodal Processing and Interaction*, pp. 1–13, Springer, 2008.
- [156] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, “Unsupervised feature learning for audio classification using convolutional deep belief networks,” in *Advances in neural information processing systems*, pp. 1096–1104, 2009.
- [157] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, “Improving Word Representations via Global Context and Multiple Word Prototypes,” in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2012.
- [158] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.
- [159] E. L. Lehmann and G. Casella, *Theory of point estimation*. Springer Science & Business Media, 2006.
- [160] “Tensorflow lite.” https://www.tensorflow.org/lite/tfmobile/android_build.
- [161] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [162] A. Kumar, M. Khadkevich, and C. Fügen, “Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 326–330, IEEE, 2018.
- [163] Y. Tokozume, Y. Ushiku, and T. Harada, “Learning from between-class examples for deep sound recognition,” *arXiv preprint arXiv:1711.10282*, 2017.
- [164] N. Kanda, R. Takeda, and Y. Obuchi, “Elastic spectral distortion for low resource speech recognition with deep neural networks,” in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pp. 309–314, IEEE, 2013.
- [165] N. Morales, L. Gu, and Y. Gao, “Adding noise to improve noise robustness in speech recognition,” in *INTERSPEECH*, pp. 930–933, 2007.
- [166] E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell, “Darwin phones: the evolution of sensing and inference on mobile phones,” in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 5–20, ACM, 2010.
- [167] L. I. Kuncheva, “A theoretical study on six classifier fusion strategies,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 2, pp. 281–286, 2002.

- [168] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE transactions on pattern analysis and machine intelligence*, vol. 16, no. 1, pp. 66–75, 1994.
- [169] N. C. Mithun, S. Munir, K. Guo, and C. Shelton, "ODDS: real-time object detection using depth sensors on embedded GPUs," in *IPSN*, 2018.
- [170] A. SALEKIN, J. W. EBERLE, J. J. GLENN, B. A. TEACHMAN, and J. A. STANKOVIC, "A weakly supervised learning framework for detecting social anxiety and depression," 2018.
- [171] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems*, pp. 3630–3638, 2016.
- [172] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean, "Zero-shot learning by convex combination of semantic embeddings," *arXiv preprint arXiv:1312.5650*, 2013.
- [173] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele, "Latent embeddings for zero-shot classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 69–77, 2016.
- [174] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng, "Zero-shot learning through cross-modal transfer," in *Advances in neural information processing systems*, pp. 935–943, 2013.
- [175] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell, "Zero-shot learning with semantic output codes," in *Advances in neural information processing systems*, pp. 1410–1418, 2009.
- [176] Z. Zhang and V. Saligrama, "Zero-shot learning via semantic similarity embedding," in *Proceedings of the IEEE international conference on computer vision*, pp. 4166–4174, 2015.
- [177] E. Kodirov, T. Xiang, and S. Gong, "Semantic autoencoder for zero-shot learning," *arXiv preprint arXiv:1704.08345*, 2017.
- [178] S. Palipana, D. Rojas, P. Agrawal, and D. Pesch, "Falldefi: Ubiquitous fall detection using commodity wi-fi devices," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, p. 155, 2018.
- [179] W. He, K. Wu, Y. Zou, and Z. Ming, "Wig: Wifi-based gesture recognition system," in *Computer Communication and Networks (ICCCN), 2015 24th International Conference on*, pp. 1–7, IEEE, 2015.
- [180] M. Schulz, D. Wegemer, and M. Hollick, "Nexmon: Build your own wi-fi testbeds with low-level mac and phy-access using firmware patches on off-the-shelf mobile devices," in *Proceedings of the 11th Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization, WiNTECH '17*, pp. 59–66, Oct. 2017.
- [181] R. Srinivasan, M. T. Islam, B. Islam, Z. Wang, T. Sookoor, O. Gnawali, and S. Nirjon, "Preventive maintenance of centralized hvac systems: Use of acoustic sensors, feature extraction, and unsupervised learning,"
- [182] T. DelSole, "A fundamental limitation of markov models," *Journal of the atmospheric sciences*, vol. 57, no. 13, pp. 2158–2168, 2000.

- [183] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.
- [184] K. Krishna and N. M. Murty, “Genetic k-means algorithm,” *IEEE Transactions on Systems Man And Cybernetics-Part B: Cybernetics*, vol. 29, no. 3, pp. 433–439, 1999.
- [185] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine,” in *International workshop on ambient assisted living*, pp. 216–223, Springer, 2012.
- [186] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “A public domain dataset for human activity recognition using smartphones,” in *Esann*, 2013.
- [187] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, “One-shot imitation learning,” in *Advances in neural information processing systems*, pp. 1087–1098, 2017.
- [188] E. Kodirov, T. Xiang, Z. Fu, and S. Gong, “Unsupervised domain adaptation for zero-shot learning,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2452–2460, 2015.
- [189] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, “Label-embedding for image classification,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 7, pp. 1425–1438, 2016.
- [190] X. Xu, T. Hospedales, and S. Gong, “Transductive zero-shot action recognition by word-vector embedding,” *International Journal of Computer Vision*, vol. 123, no. 3, pp. 309–333, 2017.
- [191] K. Liu, W. Liu, H. Ma, W. Huang, and X. Dong, “Generalized zero-shot learning for action recognition with web-scale video data,” *arXiv preprint arXiv:1710.07455*, 2017.
- [192] N. Madapana and J. P. Wachs, “A semantical & analytical approach for zero shot gesture learning,” in *Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on*, pp. 796–801, IEEE, 2017.
- [193] H.-T. Cheng, M. Griss, P. Davis, J. Li, and D. You, “Towards zero-shot learning for human activity recognition using semantic attribute sequence model,” in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pp. 355–358, ACM, 2013.
- [194] “Say hello to Cortana.” <http://www.microsoft.com/en-us/mobile/experiences/cortana/>.
- [195] “OK Google Voice Search and Actions.” <https://support.google.com/websearch/answer/2940021?hl=en>.
- [196] “Amazon Echo: Always Ready, Connected, and Fast.” www.amazon.com/echo.
- [197] S. Boll, “Suppression of acoustic noise in speech using spectral subtraction,” *IEEE Transactions on acoustics, speech, and signal processing*, vol. 27, no. 2, pp. 113–120, 1979.
- [198] P. C. Loizou, *Speech enhancement: theory and practice*. CRC press, 2013.
- [199] R. Martin, “Spectral subtraction based on minimum statistics,” *power*, vol. 6, p. 8, 1994.

- [200] S. K. Mitra and J. F. Kaiser, *Handbook for digital signal processing*. John Wiley & Sons, Inc., 1993.
- [201] S. P. Meyn and R. L. Tweedie, *Markov chains and stochastic stability*. Springer Science & Business Media, 2012.
- [202] B. A. Barsky, R. H. Bartels, and J. C. Beatty, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Los Altos, Calif.: M. Kaufmann Publishers, 1987.
- [203] X. Cui, V. Goel, and B. Kingsbury, “Data augmentation for deep neural network acoustic modeling,” *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 9, pp. 1469–1477, 2015.
- [204] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, “Audio augmentation for speech recognition,” in *Proceedings of INTERSPEECH*, 2015.
- [205] T. K. Ho, “Random decision forests,” in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1, pp. 278–282, IEEE, 1995.
- [206] B. Widrow, J. R. Glover, J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, J. E. Dong, and R. C. Goodlin, “Adaptive noise cancelling: Principles and applications,” *Proceedings of the IEEE*, vol. 63, no. 12, pp. 1692–1716, 1975.
- [207] B. Widrow and S. D. Stearns, “Adaptive signal processing,” *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1985, 491 p.*, vol. 1, 1985.
- [208] “Amazon Voice Service.” <https://developer.amazon.com/alexa-voice-service>.
- [209] “BeagleBone Black.” <http://beagleboard.org/black>.
- [210] “Beaglebone black pru.” <http://beagleboard.org/pru>.
- [211] “SparkFun Electret Microphone Breakout.” <http://www.ti.com/product/OPA344>.
- [212] “Modular toolkit for data processing (mdp).” <http://mdp-toolkit.sourceforge.net/>.
- [213] “Pandas library.” <http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.interpolate.html>.
- [214] “libpruio.” <http://users.freebasic-portal.de/tjf/Projekte/libpruio/doc/html/index.html>.
- [215] “Device tree overlay.” <https://github.com/beagleboard/bb.org-overlays>.
- [216] “Tascam.” <http://www.kraftmusic.com/tascam-trackpack-4x4-complete-recording-studio-bonus-pack.html>.
- [217] “Electret mic.” <https://www.adafruit.com/product/1713>.
- [218] W. Diao, X. Liu, Z. Zhou, and K. Zhang, “Your voice assistant is mine: How to abuse speakers to steal information and control your phone,” in *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, pp. 63–74, 2014.

- [219] N. Roy, S. Shen, H. Hassanieh, and R. R. Choudhury, “Inaudible voice commands: The long-range attack and defense,” in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pp. 547–560, 2018.
- [220] T. Vaidya, Y. Zhang, M. Sherr, and C. Shields, “Cocaine noodles: exploiting the gap between human and machine speech recognition,” in *9th {USENIX} Workshop on Offensive Technologies ({WOOT} 15)*, 2015.
- [221] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, “Dolphinattack: Inaudible voice commands,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 103–117, 2017.
- [222] H. Feng, K. Fawaz, and K. G. Shin, “Continuous authentication for voice assistants,” in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pp. 343–355, 2017.
- [223] G. Petracca, Y. Sun, T. Jaeger, and A. Atamli, “Audroid: Preventing attacks on audio channels in mobile devices,” in *Proceedings of the 31st Annual Computer Security Applications Conference*, pp. 181–190, 2015.
- [224] F. H. Shezan, H. Hu, J. Wang, G. Wang, and Y. Tian, “Read between the lines: An empirical measurement of sensitive applications of voice personal assistant systems,” in *Proceedings of The Web Conference 2020*, pp. 1006–1017, 2020.
- [225] A. Hyvärinen and E. Oja, “A fast fixed-point algorithm for independent component analysis,” *Neural computation*, vol. 9, no. 7, pp. 1483–1492, 1997.
- [226] E. Le Sueur and G. Heiser, “Dynamic voltage and frequency scaling: The laws of diminishing returns,” in *Proceedings of the 2010 international conference on Power aware computing and systems*, pp. 1–8, 2010.
- [227] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott, “Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling,” in *High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on*, pp. 29–40, IEEE, 2002.
- [228] K. Choi, R. Soma, and M. Pedram, “Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times,” *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 24, no. 1, pp. 18–28, 2005.
- [229] S. Herbert and D. Marculescu, “Analysis of dynamic voltage/frequency scaling in chip-multiprocessors,” in *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*, pp. 38–43, IEEE, 2007.
- [230] W. R. Dieter, S. Datta, and W. K. Kai, “Power reduction by varying sampling rate,” in *Proceedings of the 2005 international symposium on Low power electronics and design*, pp. 227–232, ACM, 2005.
- [231] K. Woods, W. P. Kegelmeyer, and K. W. Bowyer, “Combination of multiple classifiers using local accuracy estimates,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 405–410, 1997.

- [232] “Linux 802.11n csi tool installation instructions.” <https://dhalperi.github.io/linux-80211n-csitool/installation.html>.
- [233] “Csi data parsing.” <https://github.com/tamzeed-unc/csi-processing/tree/master>.
- [234] “Glove: Global vectors for word representation.” <https://nlp.stanford.edu/projects/glove/>.
- [235] “Verb attributes.” <https://github.com/uwnlp/verb-attributes>.
- [236] “Mqtt: The standard for iot messaging.” <https://mqtt.org>.
- [237] “Mqtt server side code.” <https://github.com/tamzeed-unc/sync-audio-rec-mqtt>.
- [238] “Audio and music processing in python.” <https://librosa.org>.
- [239] “Fastica: a fast algorithm for independent component analysis.” <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.FastICA.html>.
- [240] “Gradient flip.” https://github.com/pumpikano/tf-dann/blob/master/flip_gradient.py.
- [241] “Beaglebone black pru-icss resources.” <http://beagleboard.org/pru>.