Acoustic Intelligence in Machines

Anurag Kumar

CMU-LTI-18-015
Sep 27th, 2018

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: Audio Event Detection, Weak Label Learning, Deep Learning for Sounds, Transfer Learning for Sounds, Natural Language Understanding of Sounds

I have often lamented that we cannot close our ears with as much ease as we can our eyes.
–Richard Steele, 1720

To My Parents

Abstract

One of the desiderata in machine intelligence is that machines must be able to comprehend sounds as humans do. They must know about various sounds, associate them with physical objects, entities or events, be able to recognize and categorize them and know or be able to discover relationships between them. We call this intelligence, Acoustic Intelligence.

Automated machine understanding of sounds in specific forms such as speech and music, has become relatively advanced, and have been successfully deployed in systems which are now part of our daily life. However, the same cannot be said about other naturally occurring sounds in our environment. Speech production is constrained by our vocal cords, restricting in certain senses the input space over which machines need to work for their automated understanding. However, naturally occurring sounds are entirely unrestricted. The problem is exacerbated by the sheer vastness of the number of sound types, the diversity, and variability of sound types, the variations in their structure, and even their interpretation.

We formalize acoustic intelligence in machines as consisting of two main problems; first, in which we aim to acquire commonsense knowledge about sounds and the second, where we consider the problem of recognizing their presence in audio recordings. The first one requires natural language understanding of sounds and the second one is about large-scale recognition and detection of sound events and scenes. On the natural language understanding front, we develop methods to identify and catalog "audible phrases" and then to extract higher-level semantic relationships for sounds using these audible phrases. To the best of our knowledge, this is the first work to extract sound related knowledge from textual data.

On the sound event recognition front, we address the primary barrier of lack of labeled data for sounds. We propose to learn from weakly labeled data and show for the first time that audio event detectors can be trained using weakly labeled data. We formulate the problem through multiple instance learning and then describe several methods under this framework for weakly supervised audio event detection. We then give deep learning methods for weakly labeled audio event detection as well, leading to the state of the art performances on several datasets. We show that these deep learning methods can be further employed in transfer learning for sounds. Finally, on the weak label front, we also propose a unified learning framework which leverages both strongly and weakly labeled data at the same time.

The above methods try to address labels challenges in the learning phase. We attempt to address the challenges during the evaluation phase as well. Evaluation of trained models on a large scale test set once again requires data labeling. We describe methods to precisely estimate the performance of a trained model under restricted labeling budget.

Acknowledgments

First and foremost, I would like to thank my advisor Bhiksha Raj. It is because of his firm belief in me and patience that I have been able to complete this journey of PhD. He was always available to not only share his immense knowledge and expertise but also consistently motivated and guided me in the right direction. He taught me the importance of big picture and long-term vision which are essential not only in research but in life as well. I could not have asked for a better advisor, and I will miss just walking in to his office to talk about research, life, and many other things.

I would like to thank Prof. Rita Singh. I worked closely with her and always looked forward to her feedback on my research. She helped me in understanding the subtleties and nuances of different problems I undertook, which played an important role in shaping my dissertation.

I would also like to express my sincere gratitude to other members of my thesis committee, Dan Ellis, Alex Hauptmann, and Louis-Philippe Morency. I am indebted for their valuable feedback, comments, and technical discussions on my dissertation. The lessons I learned from their insights and difficult questions not only impacted this dissertation but will continue to shape my future research endeavors.

My sincere thanks also to Dinei Florencio for hosting and mentoring me at Microsoft Research. He gave me the opportunity to work on speech enhancement, an area of research which I found to be incredibly fascinating and interesting. I would then like to thank Christian Fugen and Maksim Khadkevich at Facebook Research for hosting me for another exciting summer in industry. I had the opportunity to see some real-life applications of my research with them.

I would like to wholeheartedly thank the Robust-MLSP research group. My most sincere thanks and deepest gratitude to Richard Stern for his valuable comments and constructive feedback on my works and presentations. Many many thanks to Raymond Xia, Mahmoud Alismail, Benjamin Elizalde, Abelino Jimenez, Anders Oland, Tyler Vuong, Yang Gao, Wayne Zhao, Yandong Wen, Nikolas Wolfe, Shahan Memon, Hira Dhamyal, Wenbo Liu, Anjali Menon, Nstor Becerra Yoma, and Nia Bradley. I am grateful for their suggestions and helpful discussions. I learned something from each one of them, and am fortunate to have them as my collaborators and friends. I would especially like to thank Raymond Xia for his numerous suggestions on my thesis presentation and draft; and Ankit Shah and Benjamin Elizalde for being amazing collaborators on some of my works. I will always cherish and remember the countless dinners and endless fun with Mahmoud, Benjamin, Raymond, and Tyler.

I would also like to acknowledge the amazing LTI staff for their help and support. I want to especially thank Stacey Young for her exceptional readiness in helping and supporting students.

Lastly, I would like to thank my family. I will begin by thanking my parents, Saroj and Surendra, for their unconditional and never-ending love, and for supporting every decision I made. I want to thank Anamika, Rajesh, Vivek and Neha for their continuous love, support, motivation, and encouragement. My nieces Khushi and Shruti for being the highlight of my yearly vacation to India. They brought smiles on my face more than anyone else in this world.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Unlike seeing, where one can look away, one cannot 'hear away' but must listen ... hearing implies already belonging together in such a manner that one is claimed by what is being said.

Hans-Georg Gadamer

We begin by quickly defining perception and intelligence, two terms which will frequently occur in this dissertation. The definition of *Perception* according to the Oxford English Dictionary is *"The ability to see, hear or become aware of something through the senses"*. This perception of the environment around us leads to human cognition and intelligence. Defining "Intelligence" is hard, as R. J. Sternberg said (quoted in [Gregory and Zangwill, 1987]), " Viewed narrowly, there seem to be almost as many definitions of intelligence as there were experts asked to define it" [Legg et al., 2007]. However, to take one of the simplest definition from American Heritage Dictionary [Legg et al., 2007], *Intelligence* can be defined by "The capacity to acquire and apply knowledge". Hence, the idea of perception and intelligence, in simpler words involves observing and understanding the environment around us and then using that knowledge in our behavior and interaction with the surroundings.

In this dissertation, we are concerned with machine perception and intelligence. *Machine Perception* refers to the idea of machines having capabilities to perceive the environment, similar to humans and animals capabilities. Borrowing Malcolm Tatum's words [1], Machine Perception refers to "the capability of a computer system to interpret data in a manner that is similar to the way humans use their senses to relate to the world around them". Artificial Intelligence (AI) or Machine Intelligence is the process of a machine *perceiving* the data in different forms and then taking appropriate decisions to achieve an end goal.

Going back to human perception, vision and sound are unarguably two essential senses for humans to perceive the world around us. In particular, our focus throughout this dissertation is on sounds. Auditory stimuli play a critical role in helping us understand and respond to the environment around us. Due to its significance, sound perception by humans has been widely studied in the field of Psychoacoustics [Alberti, 2001, Moore, 2012, Plack, 2018]. It studies both anatomy and the physiology of our auditory system and also the perception of the sound by our brain. The ear system (outer, middle, inner) first converts the sound pressure waves into nerve impulses, and then these nerve impulses are carried to the brain stem, where the

[1]http://www.wisegeek.com/what-is-machine-perception.htm

perception happens. Our brain is not only able to recognize the sound but is also capable of sound localization and association with sources. Altogether, human auditory perception is highly developed, and in fact, it has been shown that sounds can sometimes suppress visual perception [Hidaka and Ide, 2015].

Altogether, given the significance of vision and sounds for humans, it is not hard to realize their importance for artificial intelligence. Hence, AI researches focus a lot on the automated understanding of visual and audio data. In this work, the focus is on audio data, which can feed acoustic information to machines. Audio data in the simplest terms are digital representations of sound pressure waves, waves which are responsible for the auditory stimuli in humans and animals.

Besides their inherent significances, as noted above, there is one important characteristic of sounds which gives them a crucial advantage over vision. Sounds are *omni-directional*, due to which they often provide information about objects and regions which are outside of our field of view. We use these additional pieces of information in our interaction with the surroundings. This advantage over vision makes automated understanding of sounds even more crucial for the further advancement of artificial intelligence. We expect and need machines to be able to make sense of sounds. We would expect that machines should be able to listen and recognize auditory objects, actions, events; and use sound as a source of information in the decision-making process. The term *Machine Hearing* has also been used to refer to such capabilities in machines [Lyon, 2010]. Now that we have established the importance of sounds for artificial intelligence, let us define *Acoustic Intelligence in Machines*.

We call systems which know about various sounds, associate them with physical objects, entities or events, can recognize and categorize them, know or able to discover relationships between them as ***Acoustically Intelligent***. This dissertation aims to develop methods and algorithms to create technologies which can enable machines to be *Acoustically Intelligent*.

As far as acoustic intelligence in machines is concerned, sounds in specific forms such as speech and music have received the attention of machine learning and signal processing community for quite a long time [Juang and Rabiner, 2005, Schloss, 1986]. Speech is perhaps the most important mode of communication and research works on their automated understanding and interpretation have been studied for the past several decades. Isolated digit recognition in speech started as early as 1952 at Bell Laboratories. In recent years, for Automatic Speech Recognition (ASR), human parity have been claimed under certain situations [Xiong et al., 2016]. ASR and music retrieval systems are now part of several devices and applications which are in constant use by millions of consumers all around the world. However, the same cannot be said about other naturally occurring sounds in our environment.

As one would expect, the early works on sound events tried to understand the human perception of them [McAdams, 1993, Repp, 1987, Warren and Verbrugge, 1984]. Soundscape related works which tried to understand the environment through sounds also had similar motivations [Carles et al., 1992, Schafer, 1993], that is understanding human perception of sounds. These works tried to understand and explain human recognition and perception of sounds. For example, [McAdams, 1993] described stagewise processes for human auditory recognition and identification.

Some of the early works on automatic methods for classification and retrieval of sound events came in early and mid-1990's [Feiten and Günzel, 1994, Wold et al., 1996]. The first decade of 21^{st} century saw some more interests in audio content analysis and audio event classification, primarily in the context of multimedia retrieval and surveillance. The interest in recognizing and detecting sounds continued to grow, and in last few years it has started to receive considerable attention. We give a detailed account of the literature in this area in a later chapter. The main driving factor behind the increasing significance of the field has been the numerous applications automated sound understanding might have. As more and more intelligent systems become part of our daily life, the role sound of sound in them will become more apparent and obvious. Several applications are outlined later in this chapter. In the next sections, starting with sound events and acoustic scenes, we lay down some of the problems in machine understanding of sounds. An outline of the solutions proposed for the problems is also presented.

1.1 Sound/Audio Events and Acoustic Scenes

In this dissertation document, we will interchangeably use *Audio Event* and *Sound Event* to refer to the same phenomenon. How would one define *Sound Event* or *Audio Event* ? Before going into that it might be a good idea to understand the notion of *objectness* in sounds.

1.1.1 Sound Objects

Humans are able to identify not only acoustic phenomenon which they are familiar with but are also able to detect and make sense of those we have never encountered earlier, based only on how the phenomenon stands out against the background. This ability greatly enables us to form our own vocabulary of sounds from repetitions of the detected novel phenomena. Cognitively, it is argued by several researchers that there may be an underlying model of "objectness" that human (or animal) listeners subscribe to, and that we are able to detect the occurrence of acoustic phenomena that conform to this notion of objectness [Dyson and Alain, 2004, Griffith and Warren, 2003, Kubovy and Van Valkenburg, 2001].

The concept of an object is itself hard to articulate, and philosophers through time, from Leucippus and Plato to Kant [Kant, 2003] and Russel [Russell, 1945] have struggled to define it. Cognitive scientists too have found it difficult to arrive at a precise definition. From a purely cognitive point-of-view, objects are defined as the bases of experience [Griffith and Warren, 2003]; a definition that conforms with Merriam Webster dictionary's definition of an object as "something that may be perceived by the senses". In effect, by these definitions, objects are perceptual entities that are fundamentally a function of the sensory processes that perceive them. Even so, a concrete definition remains elusive, with definitions largely being along the lines of a composition of parts or percepts [Feldman, 2004, Treisman, 1986].

However, in the physical world, the concept of an object is something all of us are familiar with, in spite of all the inherent ambiguities (e. g. A door handle is an object, but then so is a door that it is a part of). To quote American jurist Potter Stewart out of context, "we know one when we see one". The word "see" is particularly relevant here – although it is agreed that the concept of "objecthood" extends to perceptions derived from all senses, the majority of the discussions and descriptions in the literature have centered on visual objects.

Not surprisingly then, researchers in computer vision have attempted to emulate this human facility of detecting objects in their visual field. In the context of computer vision, this translates to detecting objects in digital images, based only on their "objectness". For computational purposes, however, they have avoided hierarchical-grouping-based definition of objects such as in [Feldman, 2004], and work from the simpler saliency-based description given in Alexe et al. [Alexe et al., 2010]: an image object is defined as something which holds at least one of the following three characteristics (a) it has a well defined closed boundary in space (b) it has an appearance which is different from the surrounding (c) stands out as salient or is unique within the image.

A significant literature has since sprung up that builds on the concept of objectness to detect objects in images.

The concept of a “sound object” follows the same rationale as a “visual” object. Sound objects are distinct acoustic percepts that we distinguish from the background as possibly representing a cogent phenomenon or source that stands apart from the background. Unfortunately, they too suffer from the Potter-Stewart syndrome: we know them when we encounter them, but they are hard to define.

Since objects are essentially cognitive constructs, let us then refer to researchers in auditory cognition to obtain a definition. From a cognition perspective, sound objects – if they exist – are sensory entities built upon auditory stimuli; hence, while we refer to “sound” objects in deference to the fact that these objects are definitive units that we expect to detect in digital recordings of sound, the literature in cognition refers to “auditory” objects. For want of a convincing argument to the contrary, we will assume that sound objects are the same as auditory objects.

Even among auditory neuroscientists, we encounter debate and diversity of definition. Bregman [Bregman, 1990] rejects the very notion of an auditory object, claiming instead that auditory “streams” are the fundamental units of the auditory world. More commonly, auditory objects are associated with sources [Griffith and Warren, 2003]. The flaw in this association is that a source may produce more than one type of sound; at the same time, a phenomenon that a human may identify as a sound object (*e.g*, a snippet of music) may have many sources.

The peripheral auditory system constructs two-dimensional characterizations of sound akin to “images” [Patterson et al., 1995, Shamma, 2001]. Based on this, researchers such as Kubovy and Valkemberg [Kubovy and Van Valkenburg, 2001] propose a visual analogy that resembles the propositions in Alexe *et al.* [Alexe et al., 2010]: auditory objects are defined as salient phenomena that lie within clear boundaries in two-dimensional characterizations such as spectrograms. By this rule, individual formants in a recording would stand out as objects. However, it is questionable how much the visual analogy can be applied. The fundamental difference is that visual objects are formed from *presence* of physical objects, while sound objects result from their *actions*. Lemaitre and Heller [Lemaitre and Heller, 2013] studied a taxonomy of everyday sounds and concluded that sounds produced through “specific actions” are easier to identify compared to general ones, a characteristic that may be at variance with visual objects.

To reference some more researchers, *e.g.* [Dyson and Alain, 2004] propose that auditory objects are formed through the grouping of time-frequency components based on perceptual expectations. Unfortunately, this definition is simultaneously over generative and over inclusive – grouping can group parts of what a human would identify as an object as separate; at the same time a background, which is not an object, too would be grouped. Moreover, segregation and grouping of time-frequency components may occur at many scales, not all of which conform to our notion of objects. Other scientists have proposed *detection-based* definitions, via models of the cognitive processes that find acoustic objects. Shamma [Shamma, 2001] suggests that objects are identified from coincident spike discharges in separated auditory nerves, Husain *et al.* [Husain et al., 2004] propose that objects may be detected through a hierarchical 3-stage process in the brain, Griffiths and Warren [Griffith and Warren, 2003] suggest a cascade of processes, and Adams and Janata [Adams and P., 2002] propose a template-based model for auditory object detection.

Even though the literature provides many hints, it does not provide a definitive answer to what a sound object may be, and how it can be characterized in terms that will allow us to build a computational model. Some of the principles proposed in the literature are, however, useful. Griffiths *et al.* [Griffith and Warren, 2003] aver that auditory objects must be temporally restricted. Clearly, sound objects possess saliency, as suggested by Kubovy and Vanvalkemberg [Kubovy and Van Valkenburg, 2001]. They must stand out against the background.

Beyond these guidelines, we must eventually state a definition of sound objects ourselves. Besides the perplexities pointed out earlier, we also note that other factors such as overlap, amplitude, rhythm etc. can create further difficulties in defining sound objects. However, similar issues such as occlusion, and perspective are also faced in computer vision; these are lateral to the definition of objects themselves. Possibly the most transparent way to define sound objects is one alluded to earlier: based on human judgment. Simply stated, our definition will be – if a human can detect it, it is an object. Pierre Schaefer's work [Chion, 1983] on sound objects is one of the early works on the theory of sound objects and our definition of sound objects closely follows his idea of it.

Sound objects are generally noted to have the first two of the following properties and may or may not satisfy the third property. (a) They have clear onset time (b) they are acoustically salient they possess characteristics that distinctly separate them from what listeners may perceive as background, noise or silence, and (c) the offset time is evident. The relaxation on offset time is because we find that the onset of a detected object is always noticeable to listeners, whereas the offset time may not be distinctly noted and has poor agreement rate. The idea of sound events can be based on the idea of sound objects.

1.1.2 Sound Events

Sound objects, by our convention, are cognitively distinct units. While they may form events or be constituents of events, they will not themselves be comprised of events. Hence, we place sound objects as a more fundamental unit compared to sound events. Sound events are more subjective to human interpretation and even naming conventions. For example, by our convention, a pure tone is a sound object. However, whether one would call it a sound event or not is debatable. From the computational modeling perspective, work such as [Gemmeke et al., 2017] even includes *silence* as a sound event in the vocabulary. Including the absence of any acoustic phenomena in the vocabulary, however, makes sense when the end goal is to build a computational model for detection of sound events.

[Virtanen et al., 2018] describes a sound event as a "specific sound produced by a distinct physical sound source". However, as the authors point out, associating a single source to an event might be difficult in several cases and in fact, what constitutes a source might itself be subjective. For example, "*car passing by*" event, may be hard to associate with one source, wheel, engine etc. are all playing a role in this case. Moreover, the way we "name" a sound can itself be confusing as far as associations with sources are concerned. For *car passing by*, we are associating the whole "car" with the event, rather than the wheels or engines, which one can argue are the primary sources for this event. Since our goal is computational modeling of acoustic phenomena, we follow a simple definition.

A sound event is simply a collection of sound objects standing out collectively as a salient acoustic phenomenon, identifiable and distinguishable by humans. It has salient and well-defined characteristics which are identifiable by humans, and has an onset and offset time, and of brief time duration. The identifiability and distinguishability often lead to association with physical sources and a semantically meaningful name. Examples of sound events are *laughter, glass breaking, birds chirping* etc.

1.1.3 Acoustic Scenes

Acoustic scenes have much broader characteristics compared to sound events, primarily referring to the acoustic traits of a whole environment. It is " entirety of sound that is formed when sounds from various sources, typically from a real scenario, combine to form a mixture" [Virtanen et al., 2018]. Acoustic scenes do not only consist of different sound events, but also noises. They often have complex and long-term acoustic signatures. Examples of acoustic scenes would be, *Home, Beach, Street, Market*, etc.

1.2 Acoustic Intelligence in Machines (AIM)

Humans (and possibly other animals) have remarkable acoustic intelligence. We are not only able to identify auditory percepts with which we are familiar with, but are also able to recognize and detect sounds for which we do not know the name or source. We are also able to identify a sound event in the presence of other sound events and noise. Our brain is able to pay attention to an acoustic phenomenon while managing to filter out others, well known as the *cocktail part effect* [Alain et al., 2000, Arons, 1992]. In fact, our ability to find distinct acoustic phenomenon allows us to build our sound event vocabulary. How humans understand sounds is an active field of research on its own [Moore, 2012, Zwicker and Fastl, 2013]. In this dissertation, we are concerned with developing methods which can be used to create such capabilities in machines.

Acoustic intelligence in machines is clearly necessary for the success of artificial intelligence and building such capabilities in machines is not easy to the say the least. Humans, as stated before, have capabilities to detect even unfamiliar auditory percepts. Based on that perception, humans can even make a good estimate of the source, the action on the source producing the sound, its shape, size, material, and even the environment. We are born with faculties which helps us achieve such perception capabilities, and we learn the association abilities by observing and interacting with the environment around us. Hence, having such capabilities in machines is easier said than done and will require decades of research.

To build acoustic intelligence capabilities in machines, one can argue to take a bottom to top approach; where we first develop algorithms and techniques for machines to be able to detect sound objects, without explicitly referring to any semantic name. Associating a given sound object or collection of them to a semantic name can be done in the next step. The goal, here, would be to arrive at a computational model which combines cognitive aspects of sound with their detection through different machine learning and signal processing algorithms. On the surface, this approach seems extremely meaningful and consequential, as it mimics the human capability of identifying the even novel acoustic phenomenon in a given audio recording. Given the output of such a model, the task of label assignment to such segments is the next sub-problem we need to solve. We might not be able to assign a label to some of the detected objects, but this is an inherent characteristic of humans as well. Hence, this approach seems fundamental by design. However, the aforementioned sub-problems are relatively harder to solve.

Despite our definition of the sound objects being computationally motivated, building computational models to detect a generic sound object is an arcane task. The approach outlined above is hard to implement in practice. If we consider a music note as a sound object, then we might have to label every note as a sound object while collecting data for sound objects. In one of our work, we simply considered a set of sound events as sound objects [Kumar et al., 2014], making the task of data collection easier. However, this simply falls under the approach we discuss next. Moreover, the task of label assignment to generic sound objects is again hard to precisely define, especially the granularity at which the labels should be or can be assigned.

Given the above problems, a more direct approach might be better suited and useful for immediate future. The more direct approach considers a vocabulary of sound events or acoustic scenes and then develop methods for machines to recognize and detect them in an audio recording or an audio stream. This approach might seem restrictive at the beginning, where we are focusing on a small predefined set of sounds, but, is in fact more useful in the direct applications. For example, being able to recognize and detect *screaming* and *gunshots* can be directly useful for surveillance applications. Moreover, from the machine learning and signal processing perspectives, directly attempting to recognize sound events seems a better ground to start with. Overall, detecting specific sound events is perhaps more worthwhile, especially given the fact that this field is still young and a lot of research is yet to come. The argument of limited size vocabulary can be further countered by doing recognition and detection at a large scale, where the system is designed to detect a large number of sounds.

Sound Event Detection (SED) or ***Audio Event Detection*** (AED) refers to the goal of machines being able to detect sound events in an audio recording. To be more precise, the *detection* task refers to temporally locating the occurrences of an event in an audio recording. Hence, detection can give a complete description of a long recording. In *Classification* or *Recognition* tasks, on the other hand, we try to classify or categorize the audio recording to one of the sounds in the vocabulary set. We would like our system to be able to detect and recognize a large number of sound events, hundreds or possibly thousands.

The first thing we need to state for sound event detection is the set of sound events to be detected. Which sound events should machines detect? How do we *catalog sounds* or build a vocabulary of sound events? Will a manually produced vocabulary suffice? How would a machine know which sounds it can expect in an environment, say *Park*. These questions point us towards commonsense knowledge humans have about sounds. We know how we colloquially refer to a sound event, in other words, the linguistic terms we use to identify a sound. How would a machine know about and understand such terms? Over time and with experience, we know what type of sounds we can expect in an environment. Can we machines learn such knowledge?

The above discussion lays down the motivation and goals of this dissertation. Figure 1.1 shows acoustic intelligence for machines as envisioned here. It primarily consists of two components, one with the goal of creating a knowledge-base for sounds and the second which focuses on recognizing and detecting sound events and scenes. Ideally, the two components should interact with each other in an intelligent manner which can lead to a better perception of sounds by machines.

Figure 1.2 lays out some problems and methods in each component of AIM. Even though we believe the interlinking between the knowledge base of sounds and sound event detection is extremely important for an acoustically intelligent system; this interlinking has not been studied in this dissertation. Instead, we focus on the individual components and develop methods and algorithms for different sub-problems within each component.

As expected, recognizing and detecting sound events is one of the primary goals. The particular focus as far as AED is concerned is on developing methods and frameworks which can scale audio event detection. However, along with that, machines need to have some commonsense knowledge about sounds, similar to humans. We obtain this knowledge by continuously observing, interacting and learning from the environment. We learn names for sounds, often in more than one language. We associated different sounds together; for example, we know that *cheering* and *laughing* often occur together. Machines also need to have this knowledge, and then only the detection and recognition of sound events can be more nuanced and useful. In fact, cataloging

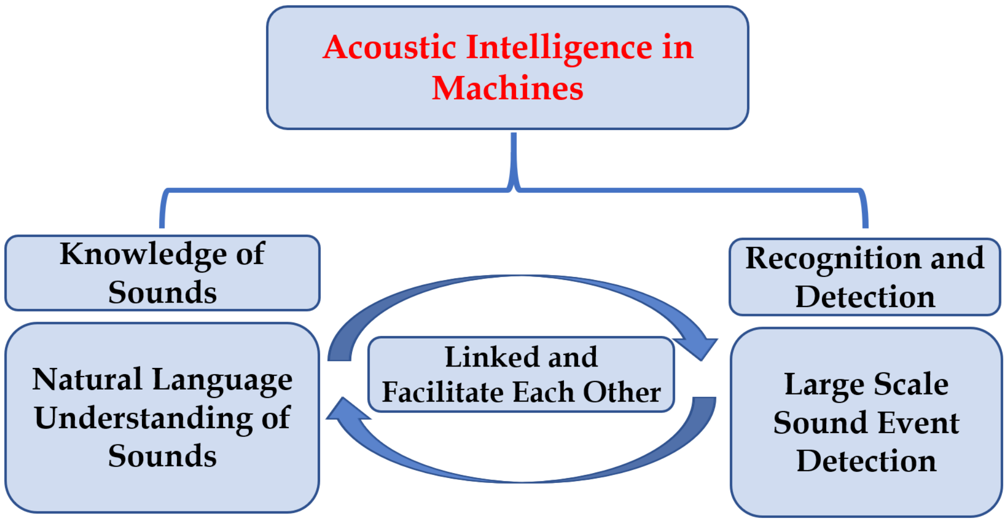

Acoustic Intelligence in Machines
Knowledge of Sounds
Natural Language Understanding of Sounds
Linked and Facilitate Each Other
Recognition and Detection
Large Scale Sound Event Detection

Figure 1.1: Acoustic Intelligence in Machines (AIM)

sound names should be the first step towards large-scale audio event detection. We need a large vocabulary to start with for large-scale sound event detection (LASSED).

Hence, natural language understanding of sounds is required, which would involve machine learning and natural language processing methods for cataloging, interpreting and understanding sounds. All in all, creating a knowledge base of sounds for machines. The dissertation proposes solutions for some fundamental aspects of creating a knowledge base for sounds, a research problem which to the best of our knowledge has not been explored much. From there one, a major part of the thesis is devoted to scaling audio event detection, where our proposed ideas have not only made large scale SED viable but have also opened up newer problems which would require the attention of the research community. In the next sections, an overview of problems and their proposed solutions is laid down.

1.3 Natural Language Understanding of Sounds

The field of automated understanding of sounds so far has mainly focused on recognizing sound events and acoustic scenes in audio recordings [2] [Stowell et al., 2015]. However, for machines to be able to actually understand and interpret various sounds, associate them with physical objects, entities or events, know or discover relationships between them; we need something beyond mere recognition of events. A comprehensive knowledge base about sounds is desirable. This knowledge base, should not only contain a large number of sound categories organized in a meaningful hierarchy but should also contain other information, such as acoustic relations. Machines should be able to learn sound event co-occurrence relations, an acoustic scene - event relations, source - event relations, to name a few. They should know different names for the same sound, for example, *glass shattering* and *glass breaking* refers to the same acoustic phenomenon. These are important as machines can have sound understanding capabilities similar to humans

[2]http://dcase.community/

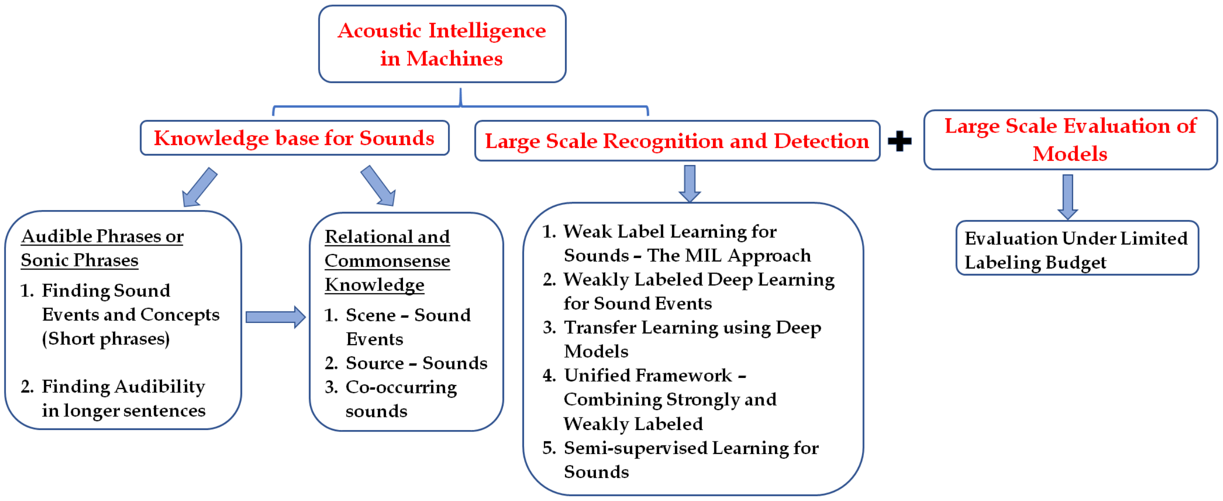

Acoustic Intelligence in Machines
Knowledge base for Sounds
Large Scale Recognition and Detection
Large Scale Evaluation of Models
Audible Phrases or Sonic Phrases
1. Finding Sound Events and Concepts (Short phrases)
2. Finding Audibility in longer sentences
Relational and Commonsense Knowledge
1. Scene - Sound Events
2. Source - Sounds
3. Co-occurring sounds
1. Weak Label Learning for Sounds - The MIL Approach
2. Weakly Labeled Deep Learning for Sound Events
3. Transfer Learning using Deep Models
4. Unified Framework - Combining Strongly and Weakly Labeled
5. Semi-supervised Learning for Sounds
Evaluation Under Limited Labeling Budget

Figure 1.2: Acoustic Intelligence in Machines

only if they have some linguistic grasp of how we understand and interpret sounds. It must learn the word phrases that identify sounds, associate sounds with these identifiers and then be able to learn to recognize and categorize them.

It is fairly easy to register that this will require us to develop methods using machine learning and natural language processing. Surprisingly, there has been no work, to the best of our knowledge, which focuses on automated methods for mining sound related knowledge on a large scale.

The first requirement for this commonsense knowledge about sounds is to understand how we "name" sounds. To put it differently, we need an extensive list of sound events or methods to find such sound names and then use those for extracting higher-level semantic knowledge. This step is also necessary to create a large vocabulary of sounds for LASSED. A sound comprehension engine must possess a comprehensive catalog of sounds that one may recognize in recordings. Except for a few hand-curated lists, no such large list of sound events, organized in any manner exists. Given the large number of sounds which we hear and recognize and the variety of ways we might refer to the same sound, creating an exhaustive list of sounds manually will be a tedious process, to say the least.

To develop methods for creating a large sound event catalog, we need to consider language and semantics as they play roles in how we describe or name a sound event. The general problem is to build algorithms which can help machine discover the sense of audibility in a given phrase. For example, phrases such as *glass breaking*, *birds chirping*, immediately signal a notion of audibility in our mind. This applies for short phrases such as *glass breaking* as well as longer phrases which can be a whole sentence. For example, a long sentence such as, *The fast running car turned right and hit the wall*, also brings out a scene in our mind which has acoustic components in it.

We define phrases which carry a notion of audibility in it as *Audible Phrases*. However, for the purpose of cataloging sound events, unigram and bigram phrases are of more immediate concern, and they will be the focus of our attention. Examples of such phrases include *laughter, music, birds chirping, honking*.

We propose to automatically create such a catalog of sound events by mining a large text corpus. We parse textual web data to identify audible phrases based on their patterns of occurrences and semantics. Put succinctly, we introduce the idea of text-based understanding of sounds, with the goal of developing methods to extract sound related knowledge from text. Other modalities such as images can also play a role here, even though we have not explored this aspect in this work.

We propose methods which search for potential sound concepts or sound event names in an unsupervised manner, or with minimal supervision. Our method is simple, yet highly effective. It relies on simple pattern search and then parts of speech tagging. More sophisticated methods can be used, provided more supervision in terms of human labeling is possible. We end up with a catalog of over 100,000 sound events, currently the most extensive known vocabulary of sound events. Once we have this extensive list, we then show how they are useful in automatically extracting higher-level semantic relations. More specifically, we develop methods for finding the acoustic scene and events relations. For a given acoustic scene, we try to find sound events which one can expect to find in that scene or environment. For example, it is common sense knowledge for humans, that we can expect to hear sounds like *Children Laughing, Birds Chirping, Footsteps* etc. in a *Park*. Our work is the first one to show that such relations for sounds can be learned through text, paving the way for a knowledge base for sounds [Kumar et al., 2017b].

1.4 Large Scale Sound Event Detection

Even though the field of sound event classification and detection has only recently started to attract significant attention, the importance of audio content analysis was realized early on [Feiten and Günzel, 1994, Wold et al., 1996]. From around mid-2000's, there has been a steady flow of research works on sound event detection, even though the volume has been small. Most of them focused on developing a variety of machine learning and signal processing techniques for recognizing sound events in audio or video recordings. Primary applications motivating the works were the content-based retrieval of multimedia [Kiranyaz et al., 2006] and surveillance [Clavel et al., 2005b]. A comprehensive study of the current literature is provided in later chapters. However, a quick look at it will reveal that these works are severely limited in scale and scope. Several factors could be attributed to these limitations. The first one is the absence of a large meaningful vocabulary of sound events. We outlined this issue and solutions in the previous section.

However, a bigger challenge exists, the challenge of *labeling audio recordings*. Obtaining labeled data is a requirement for any supervised machine learning solution including sound event detection. However, this reliance on well-labeled data is the biggest impediment to scaling SED. We address two problems concerning labeling issues in this dissertation, first one for model training and the second one during evaluation. Scaling AED is a major theme of this dissertation, and our proposed ideas and methods have turned out to be vital in scaling sound event detection.

1.4.1 Labeled Data Challenge

Large scale labeled datasets have played a critical role in the recent success in several fields such as image object recognition and speech recognition [Amodei et al., 2015, Deng et al., 2009, Hannun et al., 2014, Krizhevsky et al., 2012]. Moreover, it is well known and understood that learning algorithms generalize much better with large datasets and in a large number of cases the performance can be improved merely by training on large datasets [Banko and Brill, 2001].

Deep learning methods have been successful primarily due to the availability of large labeled datasets. However, the lack of large scale datasets for sound events, have for long been the limiting

Table 1.1: Some Public Datasets of Sound Events

Dataset	# of Events	Information
Urbansounds	10	$\sim$ 45-50 min per event, but clips have 90% overlap
ESC-50	50	3.33 min per event
DCASE 2013	16	24 short clip per class
DCASE 2016	18	Total audio data $\sim$ 60 min
FBK-Irst	16	Total audio data $\sim$ 1.7 hours
DARES	Descriptive Events	Total 2 hours audio data

factor in SED. The principal reason behind this problem is the extreme difficulty in annotating audio recordings.

Producing labeled data for supervised learning usually requires a human annotator to mark the beginnings and ends of an audio event in an audio recording. It is not only an immensely time-consuming process but also a costly affair. As a result of which, the amount of labeled data available for any sound event in most datasets is usually very small. Moreover, the arduous labeling process is too resource intensive to do for a large number of sound events. Hence, it leads to small vocabulary sound event datasets, often the number of sound events in the datasets are limited to something between 5 and 20. This can be observed in most of the publicly available datasets for sound events.[3]

Table 1.1 shows basic information about some public datasets to illustrate this. The amount of audio data available per event are very small. In fact, in most cases, only a few minutes of audio data per event is available. Another sound event dataset [Heller et al., 2009] has examples of sounds falling under different broad categories such as *impact sounds* or *rolling events.* Here also the number of examples are very few in each case.

This lack of labeled or fully supervised data creates major challenges in the learning process. Especially, as is evident from other fields, the larger the data, the better it is. Labeling audio recordings is a naturally burdensome task, and not much can be done about it. What we can instead do, is to develop algorithms which can get around it; methods which rely on manual annotation efforts to the least possible extent.

We introduce audio event detection using weakly labeled data or in other words, weakly supervised learning of audio events. Fully supervised learning requires labeled audio recordings in which time stamps of occurrences of the events are given. The time stamps allow one to extract out parts of the audio which contain the event of interest, and then use that as positive examples for that event, while using the rest as negative examples. This process is necessary for fully supervised learning where labeled examples of an event would be needed. We refer to the labeled data with time stamps as ***Strongly Labeled*** data.

We proposed and showed that the audio event detects need not rely on these strongly labeled data; instead, we trained models for AED using audio recordings for which only *weak labels* are available [Kumar and Raj, 2016b]. In ***Weakly Labeled*** data, labels are available only at recording level. Recording level labels imply only the presence or absence of an audio event is known, the time stamps of occurrences or even the number of occurrences of the event are not available.

Learning from weakly labeled data offers several advantages. *Firstly*, creating weakly labeled data is much easier compared to strongly labeled data. It requires significantly lesser effort,

[3]Audioset: A large-scale sound event dataset, released in April 2017, will be discussed eventually.

and one need not go back and forth in the recording to mark the beginnings and ends of the event. *Second* and the more significant advantage is that weak label learning shows us a way to exploit the audio data from the web. Most of the audio (multimedia) data on the web have some associated metadata which can be used to infer their weak labels automatically. Hence, if weakly supervised learning of sound events is possible, then the whole manual (labeling) factor can be removed from the learning process.

Our approach for building audio event detects from weakly labeled data is based on the Multiple Instance Learning (MIL) framework [Dietterich et al., 1997]. We formulate the problem of AED using weakly labeled data as a MIL problem and show that the MIL framework can be successfully used to train audio event detectors with only weak labels.

Next, we address AED using weakly labeled data on several fronts. We propose scalable multiple instance learning methods to tackle the issue of scalability, with which several MIL algorithms suffer. The scalable MIL methods not only improve training times by orders of magnitude but also leads to improvement in the performance.

AED with weak labels shows a way to obtain large amounts of training data, and hence, we next move into deep learning methods for AED. We propose several methods for training deep neural networks (DNNs) using weakly labeled data. We also show that these deep learning models can be used for transfer learning, which can be very helpful for situations where large-scale datasets might not be available. Subsequently, we try to analyze challenges associated with weakly supervised learning, especially when training from weakly labeled audios obtained from the web. More specifically, we look into "label noise" problems in web crawled audio data.

Weakly supervised learning (WSL) presents certain challenges of their own, especially when working with data obtained from the web without any manual supervision. *Label noise* and *Signal Noise* needs to be addressed in the learning process. To address these challenges to a certain extent, we propose a novel unified learning framework called Weakly and Strongly Labeled (WEASL) learning [Kumar and Raj, 2017a]. In this unified framework, we show that one can learn simultaneously from strongly (supervised) as well as weakly supervised exemplars. Our main idea behind WEASL is that it can be cast as a constraint form of semi-supervised learning. It allows us to exploit labeled data in both forms and we empirically show that a small amount of strongly labeled data can give a substantial improvement over only weakly supervised learning.

1.4.2 Evaluation of Models on Large Scale

Machine learning methods to reduce the need for labeling resources in the *training phase* have been in the focus of attention of researchers for a long time. These methods include unsupervised, semi-supervised and active learning [Friedman et al., 2001]. However, in machine learning, the evaluation of trained models for any classification task is almost as critical as the training step. For example, the trained event detectors trained must be appropriately evaluated on an unseen test set for a well-rounded understanding of the training algorithm.

Evaluation on a large scale is necessary to understand the generalization capabilities of the algorithm better. The problem, however, is that evaluation of trained models also needs labeling resources. We need ground truth for the test set, and this can turn out to be a severe limitation when we would like to estimate the performance of the system on a huge unseen test set. For example, one may wish to deploy a text categorization system to categorize billions of web pages. In other situation, one may want to evaluate an audio/multimedia event detection system on millions of videos from YouTube. Clearly, labeling a large chunk of these massive test sets is almost impossible. At best only a small percentage of test points can be labeled to estimate the performance. In other cases, we might have to actively evaluate classifier as test data keeps coming in.

The problem outlined above makes us wonder about the best ways to evaluate our trained models when the labeling budget for is small. The goal here is to select instances from the test set for labeling, in an intelligent manner such that the performance estimated on this smaller set is as close as possible to the one on the whole set. Stated differently, we would like to estimate the "true accuracy" using as little labeling resources as possible. This problem might appear similar to active learning, where instances are selected for labeling so that the newly labeled instances can be included in the training set, which consequently is expected to lead to better generalization. However, the goal in the evaluation phase is very different from that during training. In instance selection for evaluation, the objective is to precisely estimate the performance of an already trained classifier on a given test set using minimum labeling resources. How the classifier was trained, or the training algorithm itself is immaterial. Due to this difference in objective, it is not possible to directly port methods from active learning.

We propose methods to estimate classifier accuracy on large-scale test sets under restricted labeling budget [Kumar and Raj, 2018a]. The methods are based on stratified sampling [Cochran, 2007]. We show that the variances of the estimated accuracy, using methods based on stratified sampling are much lower compared to that using the naive method of randomly sampling instances and labeling it. This means that the proposed methods lead to a more precise estimation of accuracy for a fixed labeling budget. In other words, the amount of labeling resources required to estimate accuracy accurately is reduced by a significant amount (as much as 60% in certain cases).

1.5 Applications

Given the significance of sounds in our interaction with the environment and it is no wonder that recognizing and detecting sound events is critical to several applications. One of the most important and direct applications of sound event detection is in the content-based retrieval of multimedia data on the web. The amount of consumer-generated multimedia data on the internet has grown almost exponentially in recent times. Popular multimedia websites such as YouTube, gets hundreds of hours of multimedia recordings uploaded on it every *minute*. There are several such sites on the internet today, each of which attracts similarly large amounts of data. The recordings are mostly unannotated; descriptions if any are limited to simple high-level metadata such as the author, or a brief legend indicating the overall content. Often the legends themselves are cryptic and uninformative to the uninformed, *e.g.* "My favorite clip".

In order to be able to organize, categorize, summarize and index these recordings such that they can be retrieved through meaningful queries, one requires analysis of their *content*. Given the somewhat spotty nature of the metadata, the description of the content must usually be automatically obtained. This naturally requires automatic identification of the *objects* and *events* that occur in the recording. Multimedia recordings have both video and audio components. Often, the sounds in the recordings carry information that the video itself may not. Thus, not only the *visual* objects in the recordings be automatically detected, it is also important to detect the sounds that occur in them. This makes audio event detection extremely important for the content analysis of the multimedia data on the web. Several works have tried to incorporate information from audio for multimedia content understanding [Ayari et al., 2011, Jiang et al., 2010, Kiranyaz et al., 2006, Wold et al., 1996, Ye et al., 2012]. Detection and captioning of sounds in YouTube videos have also been done [4] [Wang et al., 2017]. Audio event detection has also been used in sports video analysis [Xiong et al., 2003b].

Another early application of audio event detection has been in surveillance [Atrey et al., 2006, Clavel et al., 2005b, Valenzise et al., 2007]. Capturing audio for surveillance purposes is much easier, it travels through obstacles and unlike vision-based surveillance does not require a "line of sight". Hence, audio-based surveillance can be done with much ease, provided automated methods for audio content analysis is available. Sound events such as *Gunshots*, *Screaming* etc. were of particular interest in the early works on AED for surveillance.

In recent years, however, the scope of applications of sound understanding has widened and gone beyond multimedia retrieval and surveillance. Voice-based personal assistants have gained considerable importance, and millions of such devices are being used every day. These devices are continuously analyzing audio, mostly speech, to provide the user with the desired information. As these assistants become ubiquitous and their use cases in our daily life grow, they will have to rely on sounds beyond speech to be useful for us. [5] This is natural, as humans rely on sounds and hence it is expected that intelligent devices will also have to rely on sounds to better understand the context and generate appropriate answers. In fact, such devices are already being developed and used by consumers [6].

In general, sound understanding is expected to play a more prominent role in human-computer/robot-interaction [Janvier et al., 2012, Maxime et al., 2014, Ren et al., 2017]. Robots can interact naturally with people and the environment only if it has listening capabilities as well, which will feed acoustic knowledge to the decision-making process in it.

Smart homes, smart cities, smart hospitals can all benefit from understanding and recognizing sounds. Sound event detection has already found its way in smart homes. Recognition of sound events such as window breaking, smoke alarm, dog barking etc. have found use cases in smart homes [Virtanen et al., 2018]. More uses of sound based context recognition for smart homes are on the works.

Smart cities are another major area of applications for automated sound analysis. Smart city applications have been a major motivation behind studies on urban soundscapes and scene classification tasks. Bello *et. al.* (in Chapter 13 [Virtanen et al., 2018]) points out several applications of automated analysis of urban soundscapes. One such application is in autonomous vehicles which clearly will form an important component of the urban environment when they become prevalent around us. Other major applications include audio based surveillance and noise monitoring. Human monitored surveillance is hard to do, especially since the demand for surveillance is growing. Hence, automated surveillance is becoming increasingly important, and audio is going to play a critical role in automated surveillance systems [Rabaoui et al., 2008, Valenzise et al., 2007]. Monitoring noise pollution in an urban environment is extremely important as it can lead to a variety of health issues. Under these circumstances, concrete efforts are being made by the research community to use automated sound understanding methods for noise monitoring. Automated understanding of sounds has also been shown to be useful in geotagging or more specifically city identification [Kumar et al., 2017a].

Self-driving cars or autonomous vehicles, in general, are yet another domain where sounds can be useful. Sensors to recognize sounds such as siren are an integral part of several self-driving

[4] https://ai.googleblog.com/2017/03/adding-sound-effect-information-to.html
[5] http://blog-idcuk.com/sound-recognition-as-a-key-strategic-technology-for-artificial-intelligence/
[6] https://www.audioanalytic.com/audio-analytic-partnership-hive/

True Class	Predicted Class		
	Positive	Negative	Total
Positive	True Positive (tp)	False Negative (fn)	p
Negative	False Positive (fp)	True Negative (tn)	n
Total	p' = tp + fp	n' = fn + tn	N

Table 1.2: True Condition and Prediction by Classifier

cars[7][8]. Sounds in autonomous vehicles can be used where omnidirectional nature of the signal needs to be exploited, such as cases where vision will fail to provide any information. This includes incoming traffic, emergency vehicles, incoming calls [Virtanen et al., 2018].

Health care is another area where the automated understanding of sounds has high application potential. Specific applications of audio analysis in smart hospitals and assisted livings including fall detection using audio, monitoring elderly and disabled persons using sound analysis [Abdoune and Fezari, 2014, Cheffena, 2016, Fleury et al., 2008, Khan et al., 2015, Litvak et al., 2008, Popescu et al., 2008], respiratory event detection [Coyle et al., 2007].

Just as sounds are crucial for humans, they play an important role in the life of other animals as well. Sounds are one of the most direct ways by which animals communicate with each other. This leads to applications in wildlife and animal habitat monitoring. Perhaps the most important one is of bird classification and recognition using their sounds [Stowell, 2018, Stowell and Plumbley, 2014, Stowell et al., 2016]. It can further be used to study their migratory habits as well. Terrestrial and marine animals monitoring using their sounds have also been shown to be successful [Fristrup and Watkins, 1993, Gunasekaran and Revathy, 2010, Lynch et al., 2013, Mitrovic et al., 2006]. A recent [9] work from Microsoft have shown promise of *Elephant* monitoring using sounds.

In this section, we pointed out some of the applications where the development of acoustic intelligence in machines are expected to play important roles. The demand for machines to be acoustically intelligent will grow as AI gets more integrated into our daily life. The research field of automated sound understanding is still young, and as it grows up, more applications will come into the picture.

1.6 Performance Metrics

In this section, we give an overview of different performance metrics which will appear in this dissertation. We provide only a brief description of all metrics and request the readers to refer to the following works for more detailed descriptions [Buckley and Voorhees, 2004, Fawcett, 2004, Irsoy et al., 2012].

Let C be the classifier for which we are computing the performance, and N be the total number of instances in the test set. Table 1.2 shows the true condition and prediction possibilities. True positive (tp) is the total number of instances which are predicted as positive class and are also from the positive class. False positive (fp) on the other hand are those instances which are predicted as positive but actually belong to the negative class. True negative (tn) represents the

[7]https://medium.com/waymo/sounds-of-the-self-driving-car-c26f30fcf76c

[8]https://www.technologyreview.com/s/604272/a-sense-of-hearing-could-make-cars-safer-and-more-reliable/

[9]`https://news.microsoft.com/on-the-issues/2018/08/09/can-sound-help-save-a-dwindling-elephant-population-scien`
`?utm_source=li&utm_source=Direct&utm_medium=organic&utm_campaign=cmg_as`

total number of instances for which true and predicted class are both negative. False negative (fn) on the other hand are those negatively labeled instances which for which the prediction by the classifier is positive class.

False positive and false negative are errors whereas true positive and true negative represent correct predictions. The accuracy and error rates are defined by

$$\begin{aligned} Accuracy &= \frac{tp + tn}{N} && (1.1) \\ Error &= \frac{fp + fn}{N} = 1 - Accuracy && (1.2) \end{aligned}$$

The true positive rate and false positive rates are defined by $TPR = \frac{tp}{p}$ and $FPR = \frac{fp}{n}$. The performance of a binary classifier is often evaluated by Receiver Operating Characteristic (ROC) curves [Fawcett, 2004]. ROC curves are obtained by plotting TPR vs FPR at different detection thresholds. The area under ROC curves (AUC) is used as a single metric to characterize ROC curves. The simplest interpretation of AUC is that it represents the probability that the classifier will rank a randomly chosen positive example higher than a random negative example. The higher the AUC the better it is. The AUC for a perfect classifier will be 1.0 and 0.5 for a random classifier.

Similar to ROC curves, sometimes precision-recall curves are also used. Precision is defined as tp/p' and recall by tp/p. Precision essentially measures how *precise* the classifier is in its prediction of the positive class whereas recall measures what fraction of the positives are correctly predicted by the classifier. The area under precision-recall curves are also higher for better classifiers.

Another type of curve often used are error curves [Martin et al., 1997]. Detection Error Tradeoff (DET) curves plot error measures on both axes, missed detection rate on the y-axis and false positive rate on the x-axis. Miss detection rate is simply $1 - TPR$ or the fraction of positively labeled instances which are not correctly predicted by the classifier. Often, the axes are scaled in DET curves for better visibility. The area under the DET curves are desired to be as low as possible, the lower the better. The area under the DET curve for a random classifier will be 0.5. Another metric widely used to characterize DET curves is the Equal Error Rate (EER). EER is defined as the value when the missed detection rate and the false positive rate are equal.

The last measure which will figure in this thesis is average precision (AP) [Buckley and Voorhees, 2004]. The average precision is a ranked measure metric. AP of a ranked list is given by

$$AP = \frac{\sum_{i=i}^{N} P(i) I_{+}(i)}{N_{+}} \qquad (1.3)$$

where N_{+} is the number of positive instances in the test set, $I_{+}(i)$ is an indicator of whether the i^{th} test instance in the ranked list is a positive instance for the class. $P(i)$ is the fraction of the top-ranked i instances which are positive or in other words precision measure for the top i instances. Note that average precision depends on the class prior and hence the numbers can often be low if the class priors are low.

1.7 Organization

The organization of this dissertation document is as follows. In the next chapter, we introduce natural language understanding for sounds. We first describe methods for finding audible phrases or cataloging sound event names. Then we discuss how higher-level semantic relations for sounds can also be extracted from a large text corpus. In Chapter 3, we introduce weak label learning for sound event detection. We describe our method and then present empirical results to show that the AED using weakly labeled data is possible. Chapter 4 discusses deep learning methods for audio event detection using weakly labeled data. More specifically, we describe convolutional neural networks based approaches for AED using weakly labeled data. Towards the end, we also discuss factors which affect learning from weakly labeled data. Chapter 5 introduces another novel learning framework for weakly labeled data. In this chapter, we describe a unified learning framework where we try to learn simultaneously from weakly and strongly labeled datasets. The problem is formulated as a constraint form of semi-supervised learning, and then we propose a graph-based solution to the problem.

Chapter 6 takes deep learning for sound events further by presenting transfer learning methods based on our deep learning model. We show that on tasks where large datasets might not be available, features extracted from deep models can lead to the state of art results on the given task.

Chapter 7 then changes gear to the evaluation phase, where we discuss the problems of large-scale evaluation under a limited budget. We show how our proposed methods can reduce labeling budget by as much as 60% for precise estimation of the accuracy of a trained classifier. Chapter 8 and 9 then look into the future of sound event detection. We talk about some examples of systems and applications of automated sound analysis in Chapter 8, and then we conclude in chapter 9, discussing some future works along the way.

Chapter 2

Natural Language Understanding of Sounds

A versifier arranges sounds; a poet arranges meaning in the sounds.

Dejan Stojanovic

Language is one of the most distinguishing factors between animals and us humans, the most intelligent species on earth. The role of language in intelligence has perplexed philosophers, rattled behavior and neuroscientists, consumed psychologists, and dazed computer scientists. Over the years, several works in these areas have tried to understand the relation and importance of language to human cognition and intelligence [Dennett, 1994, Donald, 1993, Lupyan, 2012, Russell and Norvig, 2016]. Our thoughts, understanding, and response to the environment around us, are all build around the language(s) we speak. It is then understandable that language plays an essential role in the human understanding of sounds as well; and hence, in this chapter of Acoustic Intelligence in Machines we develop methods through which language can help in the automated understanding of sounds.

2.1 Introduction

An acoustically intelligent system is expected to know about, listen, understand and meaningfully interpret sounds. Mere classification and recognition of sound events are not sufficient, as a matter of fact, classification and recognition itself are incomplete without a linguistic understanding of the sounds. This holds true for visual perception as well.

In the field of computer vision, it is noticeable that it has advanced beyond the mere recognition of a few visual objects. Object detection in vision has been successfully scaled to thousands of visual object categories [Deng et al., 2009]. Moreover, these thousands of categories are often organized into a hierarchical structure which allows higher level semantic analysis. Visual concept ontologies [1] have been proposed to enhance the semantic understanding vision can provide. Systems such as Never Ending Image Learner (NEIL) [Chen et al., 2013b] not only detects thousands of visual objects and scenes in images but is also designed to find a variety of commonsense

[1]http://disa.fi.muni.cz/results/software/visual-concept-ontology/

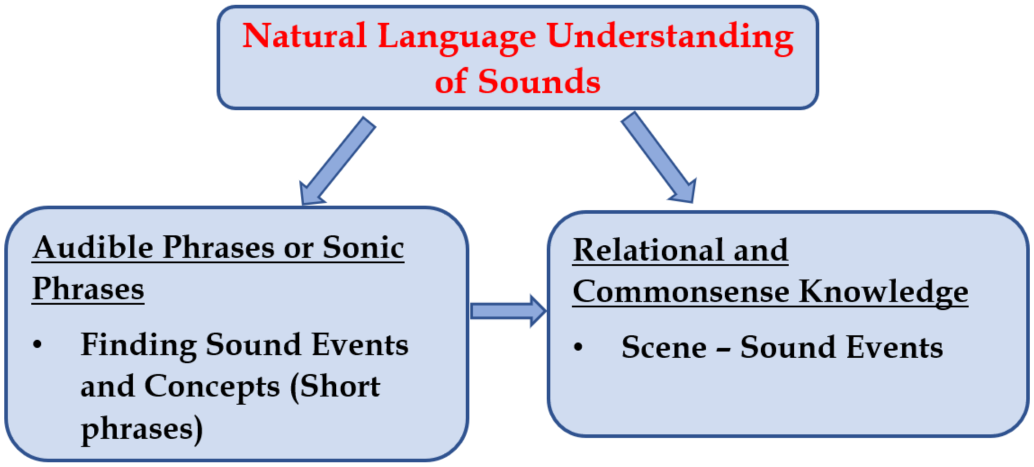

Natural Language Understanding of Sounds
Audible Phrases or Sonic Phrases
Finding Sound Events and Concepts (Short phrases)
Relational and Commonsense Knowledge
Scene – Sound Events

Figure 2.1: Natural Language Understanding of Sounds: Sub-problems we study.

visual relationships. One such visual relation is the similarity between two objects; for example, *Umbrella looks similar to Ferris Wheel*. NEIL aims to learn such relation in a continuous, never-ending fashion. The motivation is to create a visual knowledge base, similar to what Never Ending Language Learner (NELL) [Carlson et al., 2010] does by mining textual web data.

Another architecture, EventNet [Ye et al., 2015] is tailored towards multimedia content. It organizes 500 multimedia events using over 4000 visual concepts. Unarguably, these visual relations and ontologies are crucial for a semantics-driven search of multimedia data on the web. Besides these, the integration of vision and language is actively investigated in the other learning tasks as well. Generating natural language captions and descriptions of images is one such problem, and it has received considerable attention in the last few years [Karpathy and Fei-Fei, 2015, Xu et al., 2015a]. The problem generalizes to narration generation for videos [Donahue et al., 2015]. Visual Question Answering (VQA) [Antol et al., 2015] is yet another task which integrates language and vision. This coupling between language and vision is crucial if we want to have visual intelligence beyond recognition of a few visual objects in images.

Similar advancement is expected for sounds. We outlined and emphasized on this need previously in the introductory chapter (Section 1.3). Besides being able to recognize a large number of sounds and machines should also have semantic knowledge about sounds. In fact, the two tasks are interlinked with each other (Figure 1.1), and are expected to aid each other.

A critical aspect of this interlinking is the vocabulary, or the set of sound events machines should be aware of. How do we build a vocabulary of sounds? Is a handcrafted list sufficient? Perhaps not, considering the wide variety of sounds we hear and the different ways in which we express sounds. Second, and perhaps more important, how would machines gather common sense knowledge about sounds. For example, the fact that *honking, beeping* and *engine running* can all be related through one common source *car*. Scene - Sounds is another significant acoustic relation, which can be very useful for audio based context recognition systems. An acoustic scene *Park*, consists of sounds events such as *children laughing*, *birds chirping*, *footsteps* etc. is a commonsense knowledge, but how would machine learn these relations automatically.

Figure 2.1 shows an outline of what we aim to achieve by proposing to introduce natural language understanding of sounds. We introduce a text-based understanding of the sounds by proposing methods to mine sound related knowledge from textual data automatically. To the best of our knowledge, this is the first instance of work on a large scale text-based understanding of sounds. The first step focuses on the vocabulary of sound events, which we alternately, also refer to as sound concepts in this chapter. The vocabulary of sounds events in the current sound event detection literature is usually very small. More importantly, if we aspire to create a knowledge base for sounds, we need to start with being able to find phrases which carry a notion of audibility in it. Once we can successfully discover such phrases in the text, we look into methods for mining other commonsense acoustic knowledge. Specifically, we propose methods to find sound events which might occur in a given acoustic scene.

2.1.1 Related Works

Over the years, several handcrafted taxonomies of sounds have been proposed. Several of these taxonomies study environmental sounds in the context of soundscape research and contains a small number of sounds categorized in different groups. One of the most important studies in this area has been by Schafer, who also coined the term "Soundscape" [Schafer, 1993]. Schafer divided sounds into 6 broad categories, *Natural, Human, Society, Mechanical, Silence, and Indicators.* Gaver [Gaver, 1993] did another important work on the human perception of natural sounds. Here, sounds are studied from the perspective of sources and actions, or interactions between materials. The interaction between materials is taken as the basis of categorizing sounds. Some other works related to the taxonomical categorization of environmental sounds are [Brown et al., 2011, Raimbault and Dubois, 2005, Salamon et al., 2014]. One useful taxonomy of urban sounds comes from [Salamon et al., 2014], which also provides a dataset for urban sound events. However, there is no clear consensus on building such taxonomies, and in most cases, they are based on subjective opinions. Moreover, in several of these urban taxonomies, a large part of the taxonomy is made up of broad categories, and the number of low-level sound concepts is once again small. Other sound events dataset such as [Burger et al., 2012, Piczak, 2015b] are also sources of small lists of sound events.

Audioset [Gemmeke et al., 2017], released in 2017, is a large scale sound events dataset. Along with an ontology of sound events, it also provides labeled examples for over 500 sound events. Audioset, once again a handcrafted list, also organizes the set of sound events in a meaningful hierarchy. A well-structured hierarchy is not only helpful in the systematic recognition of sound events but also helps in the labeling process. Audioset consists of 7 categories at the top, *Human Sounds, Animal Sounds, Natural Sounds, Music, Sound of things, Channel-Environment-Background, Source-Ambiguous Sounds.* A total of 632 audio events are listed in Audioset, and labeled examples are provided for 527 sound events. The ontology provides a simple name for the sound event and also a short description of the sound event.

The motivation behind Audioset, is primarily to create a large dataset for audio event detection, along the lines of Imagenet [Deng et al., 2009] for image object recognition. Several factors were kept in mind while developing the ontology, such as sound concepts which were not distinguishable by listeners were merged into one category. The sound event names were also modified for convenience and brevity. For example, *walking on leaves* simply became *walking.* However, this can broaden the scope of the event to a considerable extent. *Walking on leaves* is very different from the sound of *walking on wooden floor.*

Even though Audioset is relatively much larger compared to other ontologies and datasets, it is tiny compared to the vast number of sounds we encounter every day. Moreover, often we refer to the same sound phenomena in different ways, which again limits the utility of a predefined small set of sounds like Audioset. Especially, considering that our goal is much broader and larger, which is to have a knowledge base of sounds. Hence, handcrafted lists and ontologies will not suffice, and we need methods which can automatically find phrases which have "audibility" in it. These automatic methods can also help create ontologies which can be further employed in other problems related to sounds. It can also help us find multiple ways of describing the same acoustic phenomenon, e.g *glass breaking* and *glass shattering*.

Hence, we introduce a text-based understanding of the sounds, by describing methods to mine sound related knowledge from textual data automatically [Kumar et al., 2017b]. We first address the problem of finding sound concepts in a text and then show a way to use those for establishing higher level semantic knowledge.

We describe methods which employ natural language processing and machine learning techniques on a large corpus of text for automated discovery of sound concepts. Our general approach is first to generate plausible "candidate" descriptions based on structural and common sense rules and to subsequently "filter" and "refine" the candidate lists based on a variety of classifications and filtering strategies. Under this general framework, we propose a simple yet effective *unsupervised approach*, based on Parts of Speech (POS) tags for discovering sound concepts. We then follow it by proposing a word embedding based supervised method for classifying a given text phrase into a sound phrase (concept) or a non-sound-phrase (non-sound-concept). Through this supervised method, we obtain a classifier which can identify the notion of audibility in any given text phrase.

Once we have an extensive list of sound events, a variety of other problems related to sounds can be solved. We propose methods for one specific type of relation, namely, *events which are found in a scene*. To make it more precise, we give a method for automatically describing an acoustic scene or an environment through sound events we might hear in that environment. Learning such scene - events relations can be beneficial in creating sound ontologies.

2.2 Audible Phrases or Sonic Phrases

We define *Audible Phrases* or *Sonic Phrases* as phrases which express a perception of audibility in it. The way we describe or name an acoustic phenomenon can vary depending on speaker and context. There are words which inherently refer to a sound, onomatopoeic words, for example, *murmur, laugh, clap, chirp* etc. However, beyond these words, sounds are often expressed in an intricate and complex manner. Often, we try to refer to the production process while naming or describing any sound. However, this complicates things as the production of sounds can be a complex process. This is very different from visual objects which are defined by their presence. Sounds, on the other hand, are results of actions and interactions between objects. The type of sound produced depends both on the physical object(s) as well as the action on the object(s) (or interaction among the objects). *Metals clinking* and *Metals scraping* are two very distinct sounds, even though *metal* is involved in both cases. Similarly, *wood falling* and *glass falling* are two very distinct sounds, even though the same action *falling* is involved in both cases.

While we have these short phrases representing sounds, sometimes longer sentences carry a notion of audibility in them. As an example, the sentence *The car took a sharp turn and then crashed into the wall*, clearly triggers audibility in our mind.

Our focus in this work is on developing methods which can automatically mine sound concepts or sound event names, rather than finding the possibility of audibility in longer sentences. This is of more immediate concern, more so because there is no other work on automated methods for finding audibility in texts.

2.2.1 Finding Audible Phrases: Cataloging Sounds

As mentioned before, the context plays an important role while describing sounds, which can further complicate their discovery through automated methods. To illustrate some of the ways by which sounds are described, we start with onomatopoeic words. An onomatopoeic word is fundamentally a sound word and represents an acoustic phenomenon on their own. Clearly, we would like to have these words in our catalog. Examples of such words are, *Splash, Giggle, Bark* etc.

In other cases, a direct belief of sound might not be apparent from the textual phrase on its own, but the phrase is either a source of or associated with a well-understood sound. In several of these cases, only the source or the object involved in the production of the sound is used to represent the sound. For example, it is very common to say, *I heard the sound of a jackhammer*, or *door*, or *dog*, *car*.

Moreover, sometimes action words such as *breaking* (e.g *glasses breaking*), *clinking* (e.g *metals clinking*) are added to denote sounds. From these examples, it is easy to understand that the automated generation of sound concepts is a non-trivial problem. Moreover, the automated generation of sound - descriptor phrases are confounded by the fact that they are often composed of words that by themselves may have no direct relation to sound. For instance, “Children in a playground” and “Garage door” have sound signatures, but this is not obvious from inspection of their constituent words. The associated activities that result in the sound are implicit and unstated. These subjective ways of expressing the sounds makes the problem harder.

However, phrases in such forms are often used as labels for sounds and appear in several audio event databases, and we must find ways to find such phrases automatically. So we propose an unsupervised method of obtaining sound concepts from a large text corpus.

2.2.2 Unsupervised Cataloging of Sounds

Our main idea of finding sounds in texts or generating a vocabulary of sounds from the text is a *two stage* process. In the first stage, we generate potential candidates for the vocabulary, or phrases which are potentially audible. This step is expected to be very simple and can lead to several spurious results in the list. The second stage is the *filtering* stage, where we filter the phrases obtained in the first step. This filtering can be a supervised or an unsupervised process. Figure 2.2 shows the method with unsupervised filtering. The method is based on the idea that, more often than not sound concepts are mentioned in certain specific ways. These patterns can help in identifying sound concepts.

We begin with a single pattern: “sound(s) of <Y>” where Y is any phrase, we allow Y to be up to 4 words long. We then look for occurrences of this pattern in a large corpus. In our experiments, we used the English part of ClueWeb09[2], which contains about 500 million webpages. From this we obtain a large collection of phrases such as: “sound of *honking cars*”, “sound of *gunshots*”.

However, this step produces a significant amount of noise. We, therefore, treat its output as *candidate sound concepts* and introduce a method for removing spurious entries from this collection. First, we generalize candidate concepts by replacing mentions with their part of

[2]http://lemurproject.org/clueweb09/

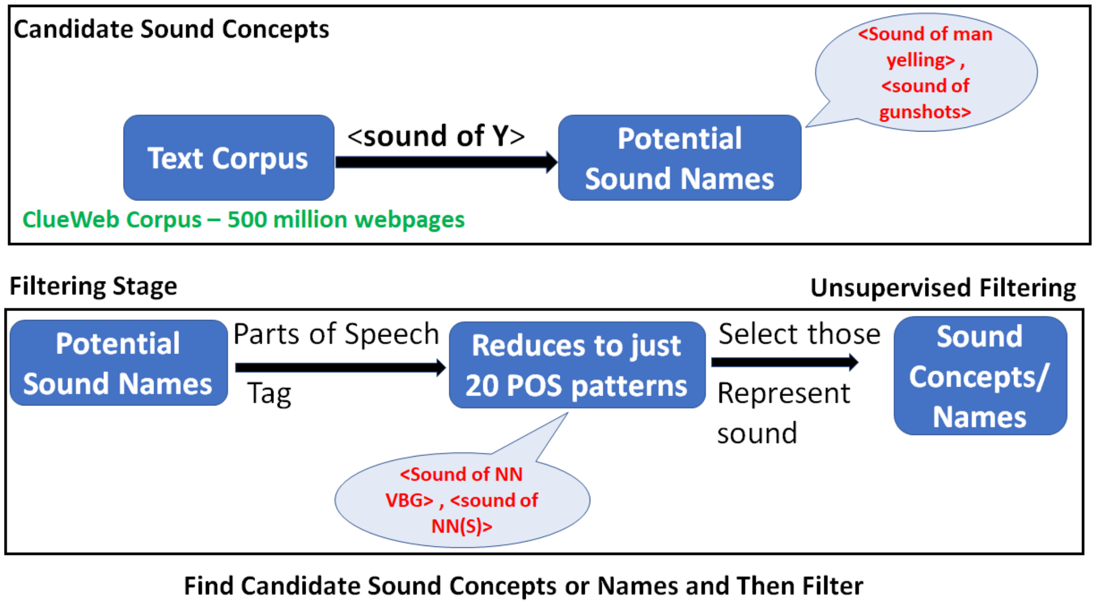

Candidate Sound Concepts
<Sound of man yelling> , <sound of gunshots>
Text Corpus
<sound of Y>
Potential Sound Names
ClueWeb Corpus – 500 million webpages
Filtering Stage
Unsupervised Filtering
Potential Sound Names
Parts of Speech Tag
Reduces to just 20 POS patterns
Select those Represent sound
Sound Concepts/ Names
<Sound of NN VBG> , <sound of NN(S)>
Find Candidate Sound Concepts or Names and Then Filter

Figure 2.2: Unsupervised cataloging of sounds

speech tags, as follows:

$$\text{sound of honking cars} = \text{sound of VBG NN}$$
$$\text{sound of gunshots} = \text{sound of NNS}$$

where, the part of speech (POS) tag VBG denotes verbs in the gerund form, NN, and NNS denote singular and plural nouns, respectively [3]. The POS generalized concepts reduce the data size to about only 20 unique patterns. Since the POS patterns are so few, we can use them to filter out noisy concepts with little effort. The key to filtering is that not all POS patterns express valid concepts. We can eliminate all but 6 of the POS patterns. For example, the pattern "sound of JJ (adjective)" does not express sound concepts. All candidate concepts that match the 6 valid POS patterns are retained, and the rest are discarded. The full list of valid POS patterns with examples are shown in Table 2.1. The patterns in Table 2.1 produced a total of $116,729$ unique sound concepts from the whole corpus. It should be noted that we have given a tiny amount of supervision in the form of pattern definitions which are most likely to be a sound concept.

2.2.3 Analysis of Unsupervised Method

The 6 POS patterns produce a total of $116,729$ sound concepts from ClueWeb corpus. The number of sound concepts for each pattern is shown in Figure 2.3. Since the list of sound concepts discovered is large, we perform a limited evaluation of the preciseness of the method. For each of the discovered sound concept phrases, we keep track of the frequency of their occurrences in the corpus. We manually label the 100 most frequent phrases for each POS pattern. A phrase is labeled positive if it is an actual sound concept or else it is labeled negative. It allows us to compute the precision for frequent (top 100) phrases for each pattern. The precision for each case is shown in Table 2.2.

[3]POS Abbreviations: https://www.ling.upenn.edu/ courses/Fall_2003/ling001/penn_treebank_pos.html

	Pattern	Example Concept
P1	$<X>$ of (DT) VBG NN(S)	honking cars, chirping birds
P2	$<X>$ of VBG	yelling, laughing
P3	$<X>$ of (DT) NN(S) VBG	dogs barking, glass breaking
P4	$<X>$ of (DT) NN(S)	gunshots, footsteps
P5	$<X>$ of (DT) NN NN(S)	church bells, train whistle
P6	$<X>$ of (DT) JJ NN(S)	classical music, heavy rain

Table 2.1: Patterns for discovering sound concepts in text. VBG is the part of speech tag for verbs in the gerund form, NN for nouns, DT for determiners, and JJ for adjectives.

	Pattern	+ in 100 Most Freq.
P1	$<X>$ of (DT) VBG NN(S)	98
P2	$<X>$ of VBG	71
P3	$<X>$ of (DT) NN(S) VBG	91
P4	$<X>$ of (DT) NN(S)	59
P5	$<X>$ of (DT) NN NN(S)	93
P6	$<X>$ of (DT) JJ NN(S)	49

Table 2.2: Precision for the 100 most frequent phrases (for each POS pattern)

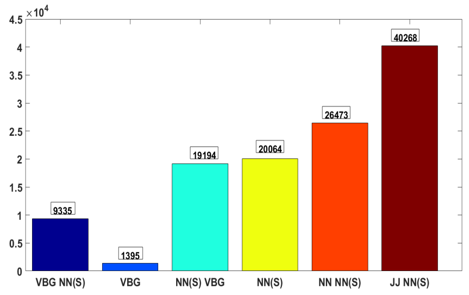


Figure 2.3: Number of sound concepts for each pattern

We observe that 3 POS patterns have more than 90% precision. The lowest is for <JJ NN(S) >at 49%. Note that, the frequency of occurrences of a discovered concept cannot be directly related to it being truly a sound concept. Frequency is a characteristic of the text corpus used. For example, *sobbing voices, siren breaking, cheering crewmen* are sound concepts but occur very few times in the text corpus. However, a good performance on the frequent phrases is an indication of an overall good performance.

POS patterns <VBG NN(S) >or <NN(S) VBG > contains a noun combined with a verb, which implies a combination of an object and an action. This combination is crucial for sound production, and often the main aspects of a sound one would expect to capture. However, based on other patterns it is worth noticing that, sounds are expressed in fairly complex ways.

To further assess the quality of discovered concepts, we consider sound events listed in some of the datasets. We considered ESC-50 [Piczak, 2015b], Urbansounds [Salamon et al., 2014], DCASE [Mesaros et al., 2016], and Audioset, currently the largest ontology and sound event list. Together all four datasets contain a list of 605 sounds (with some repetitions). A subjective look at the list of sounds in these datasets reveals some points worth mentioning.

To begin with, we note that almost all of the sounds events from these datasets are part of our discovered list. The sound classes in datasets are usually undertaken with the labeling process in mind, that is often the convenience of labeling is taken into account. In several cases, they tend to group different sounds into a single category. For example, the label *object banging* in DCASE dataset leaves us wondering about the type of “object”; this is important as different objects might produce different sounds. Consequently, the sound name *object banging* leaves us wondering about the nature of “object” and also about the exact characteristics of the sound. Our extensive list of sounds discovered show a wide range of potential objects, each of which will produce potentially different sounds. To cite a few examples from the list, *iron banging, glass* *banging, gavel banging, hammers banging, pots banging* etc. There are several such examples from across different datasets. By applying even some basic techniques such as word matching, our list can be used to get a better understanding of the sound class mentioned in the dataset.

Consider another example from the DCASE dataset, the *Dishes* sound class. This label is very generic and refers to all types of sounds associated with "dishes". Can we easily find sounds which are related to this "Dishes" class? Besides containing the class *dishes* itself, our list consists of concepts such as *dishes breaking, dishes clinking, dishes rattling, dishes shattering* etc. Clearly, these additional sound concepts represent narrower and more distinct characteristics. Since they can be easily associated with the *Dishes* class, they together give more insights into the *Dishes* sound label. Hence, a broader understanding of sounds is possible through these discovered sound concepts.

Audioset contains some sound labels which are extremely broad, for instance, *"Outside urban or manmade", "Outside rural or natural", "Inside public space"* . It can be argued that these are mostly representatives of different environments and in fact represent acoustic scenes. In these cases again the list of sounds we mined can help us understand the constituent sounds of these classes. We look into it later in this chapter.

2.2.4 Supervised Filtering

The unsupervised discovery of sound concepts in the previous section can still give spurious items. A few examples of such phrases which are not sound concepts but do not get filtered out by the two-step process in the previous section are *someone being (NN VBG), price dropping (NN VBG), gaining experience (VBG NN), happy hunters (JJ NNS)*. Hence, to improve upon the unsupervised discovery of potential sound concepts, we describe a supervised method for classifying a text phrase as an audible phrase (sound concept) or a nonaudible phrase (non-sound concept). A trained classifier is also useful in classifying any given text phrase into sound and non-sound phrase classes.

Since bigram phrases are the most dominant and expressive set of sound concepts discovered by the unsupervised method, we focus specifically on bigram phrases. A set of labeled data is required for supervised training of classifiers. To obtain a reliable set of labeled data, we manually inspect a small subset of the sound concepts obtained in the previous section and mark if it is actually a sound concept or not. Note that, in the unsupervised case only 6 POS patterns express valid sound concepts. We also use other the POS patterns to create a list of non-sound concepts as well. We manually inspect and label a small subset of this list as well. Finally, we end up with a total of ~ 6000 sound concept and non-sound concept phrases.

The text phrases need to be appropriately represented by feature vectors on which classifiers can be trained. *Word Embeddings* have been found to be very effective in capturing syntactic and semantic similarity between words [Hashimoto et al., 2015, Mikolov and Dean, 2013, Pennington et al., 2014] and have shown remarkable success in a variety of semantic tasks [Baroni et al., 2014].

In this work, we use *Word2Vec* to obtain vector representation for words in text phrases. We use Google News pre-trained embeddings [4] to represent each word by 300 dimensional vectors. We then use two methods for representing each bigram phrase. In the first case, we take the average of the word2vec representation for each word to represent the whole phrase. We refer to this representation as AWV. In the second case, we concatenate the vector representation (CWV)

[4] https://code.google.com/archive/p/word2vec/

Table 2.3: Accuracy of Supervised Learning

	AWV	CWV
Fold 1	87.03	90.00
Fold 2	89.05	89.32
Fold 3	87.84	91.87
Fold 4	89.77	90.30
Avg	88.42	90.37

for each word to obtain a 600 dimensional vector for each phrase. These vectors are then used for training a linear SVM classifier.

Analysis

We created a list of sound concepts (positive) and non-sound concepts (negative) bigram phrases. The total number of positive examples is 3189, and the total number of negative examples is 2758. We randomly divide this data into 4 folds. 3 folds are used for training, and then the trained model is tested on left out fold. The experiment is done all 4 ways. Linear SVMs are trained on both AWV features and CWV features. The accuracies for both feature representations are shown in Table 2.3. Concatenated word2vec features give slightly better performance compared to averaged word2vec features. The average accuracy of more than 90% is achieved which shows that our supervised classifier is highly reliable in classifying a text phrase as sound or non-sound phrase.

The supervised filtering we described here is just one way to illustrate another filtering process. If supervision can be made available, then a variety of other classification algorithms can also be developed. We leave them for future works and instead focus on the next part of NLU for sounds.

2.3 Learning Acoustic Scene-Concept Relations

In Figure 2.1, we argued that a major component of natural language understanding of sounds is developing methods which can automatically mine commonsense knowledge and relations about sounds. Developing a large vocabulary of sounds was the first step in this direction which we solved in the previous sections.

In this section, we consider *Acoustic Scene - Sound Event* relations, where the goal is to describe an acoustic scene or environment by the set of sounds which occur in that scene or the environment. This is usually a common sense knowledge for us. For example, we know what type sounds we can expect in *Home* or in a *Market.* Acoustic relations in these forms can be helpful in audio-based context recognition. Moreover, these relations can also provide co-occurrence information about sound concepts. For example, *laughing* and *cheering* often occur together in several acoustic environment.

From the perspective of semantic analysis in texts, we cast this task as a relation classification problem. First, we find all sentences in the ClueWeb corpus that mention at least one of the 116,729 sound concepts discovered in Section 2.2.2, and at least one acoustic environment such as "beach", "park", etc. We then apply a dependency parser[5] to any sentence that mentions a sound concept and an acoustic environment. This step produces dependencies that form a directed

[5]https://pypi.python.org/pypi/practnlptools/1.0

Table 2.4: Example Paths for Positive Training Data

prep_along()	*prep_of() sound nsubjpass() heard prep_in()*
prep_of()	*nsubjpass() filled prep_with() sound prep_of()*
prep_of() sound prep_on()	*conj_and() sounds prep_of()*
prep_with() sounds prep_of()	*prep_of() sound prep_to()*
nsubj() alive prep_with() sound prep_of()	*prep_upon()*
prep_of() sounds prep_from()	*prep_of() sounds prep_on()*
prep_of() sound nsubj() came prep_from()	*prep_of() sounds prep_at()*

graph, with words being nodes and dependencies being the edges. For example, the sentence: *"The park was filled with the sound of children playing"* , yields the following dependencies:

det(park-2, The-1)
nsubjpass(filled-4, park-2)
auxpass(filled-4, was-3)
root(ROOT-0, filled-4)
det(sound-7, the-6)
nsubj(playing-10, sound-7)
prep_of(sound-7, children-9)
prepc_with(filled-4, sound-7)'

The details of the dependency relations can be found in [De Marneffe and Manning, 2008]. Next, we traverse the dependency graph in order to obtain the path between the mention of a sound concept, in this case "children playing", and the mention of the acoustic environment "park". Shortest paths between entities have been found to be a good indicator of relationships between entities [Nakashole et al., 2013, Xu et al., 2015b]. We, therefore, extract the shortest path. In our example, the shortest path labeled with edge and node names is as follows: "nsubjpass() filled prepc_with() sound prep_of()".

2.3.1 Training Data

Given the paths, we would like to classify scene-sound pairs into those that express the relationship of interest (Sound-Found-In-Environment) and those that do not. Classifier training would require labeled training data.

To obtain training data, we proceed as follows: We sort the paths by frequency, that is, how often we have seen the path occur with different scene-sound pairs. Among the most frequent paths, we label the paths yes or no, depending on whether they express the relationship of interest. This gives us a way to generate positive and negative examples using the labeled paths.

Examples of paths that generate positive training data are shown in Table 2.4. Examples of paths that generate negative training data are shown in Table 2.5.

2.3.2 Classification

We use an LSTM recurrent neural network to learn the scene-sound relationship. Each word w is mapped to a d-dimensional vector $\boldsymbol{v_w} \in \mathbb{R}^d$ through an embedding matrix $\boldsymbol{E} \in \mathbb{R}^{|V|\times d}$, where $|V|$ is the vocabulary size, and each row corresponds to a vector of a word. We initialize the word embeddings with the 300-dimensional Google News pre-trained embeddings[4]. For the

Table 2.5: Example Paths for Negative Training Data

conj_and()	*amod()*
poss()	*nn()*
nn() sound prep_of()	*prep_through()*
prep_of()	*appos()*
det()	*conj_and() sound prep_of()*
prep_to()	*prep_of() sound nsubj() filled dobj()*

dependency relations in the path, we randomly initialize their vector embeddings and learn them during training.

Path Encoding. To encode the shortest path between a sound concept and an acoustic scene, we use a LSTM recurrent neural networks (RNN) which is capable of learning long-range dependencies. While regular RNNs can also learn long dependencies, they tend to be biased towards recent inputs in the sequence. LSTMs tackle this limitation with a memory cell and an adaptive gating mechanism that controls how much of the input to give to the memory cell, and the how much of the previous state to forget [Hochreiter and Schmidhuber, 1997].

We have a path: $\boldsymbol{p} = \boldsymbol{p}_1, ..., \boldsymbol{p}_p \in \mathbb{R}^d$ and an associated path matrix $\boldsymbol{P} \in \mathbb{R}^{p \times d}$, where each row corresponds to the embedding vector of the word in that position.
The LSTM encoder generates the path encoding, $\boldsymbol{v_p}$, as follows:

$$\begin{aligned} \boldsymbol{h}_i &= LSTM(\boldsymbol{v}_{p_i}, \boldsymbol{h}_{i-1}, \boldsymbol{c}_{i-1}), i = 1, \ldots, p \\ \boldsymbol{v}_p &= \boldsymbol{h}_i : i = p \end{aligned} \tag{2.1}$$

The LSTM encodes the word at timestep $i = t$ in the path using the word embedding vector $\boldsymbol{v}_{p_t}$, the previous output $\boldsymbol{h}_{t-1}$, and the previous state of the LSTM cell $\boldsymbol{c}_{t-1}$. The output $\boldsymbol{h}_t$ is computed using the four main elements in the LSTM cell: an input gate $\boldsymbol{i}_t$, a forget gate $\boldsymbol{f}_t$, an output gate $\boldsymbol{o}_t$, a memory cell $\boldsymbol{c}_t$ with a self-recurrent connection. The cell takes as input a d-dimensional input vector for mention word $\boldsymbol{x}_t = \boldsymbol{p}_i$, the previous hidden state $\boldsymbol{h}_{t-1}$, and the memory cell $\boldsymbol{c}_{t-1}$. It calculates the new vectors using the following equations:

$$\begin{aligned} \boldsymbol{i}_t &= \sigma\left(\boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{U}_{hi}h_{t-1} + \boldsymbol{b}_i\right), \\ \boldsymbol{f}_t &= \sigma\left(\boldsymbol{W}_{xf}\boldsymbol{x}_t + \boldsymbol{U}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right), \\ \boldsymbol{o}_t &= \sigma\left(\boldsymbol{W}_{xo}\boldsymbol{x}_t + \boldsymbol{U}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o\right), \\ \boldsymbol{u}_t &= \tanh\left(\boldsymbol{W}_{xu}\boldsymbol{x}_t + \boldsymbol{U}_{hu}\boldsymbol{h}_{t-1} + \boldsymbol{b}_u\right), \\ \boldsymbol{c}_t &= \boldsymbol{i}_t \odot \boldsymbol{u}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1}, \\ \boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \end{aligned} \tag{2.2}$$

where σ is the sigmoid function, $\odot$ is element-wise multiplication, the W and U parameters are weight matrices, and the b parameters are bias vectors.

Prediction. From the path encoding $\boldsymbol{v}_p$, we compute the output of the neural network, a distribution over the positive and negative labels. The output for each word is decoded by a linear layer and a *softmax* layer into probabilities over the labels. Therefore, the prediction $\boldsymbol{d}_r$ is given by

$$\boldsymbol{d}_r = \text{softmax}(\boldsymbol{W}_r \cdot \boldsymbol{v}_p) \tag{2.3}$$

where $\text{softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}$.

Table 2.6: 36 Acoustic environments

	Acoustic Environments			
1	Office	Farm	House	Bus
2	Parties	Funeral	Library	Park
3	Street	Parking Lot	Church	Train
4	Airplane	Wedding	Cafe	Cities
5	Campus	Ballgame	Bathroom	Classroom
6	Train Station	School	Parks	Bar
7	Grocery Store	Trucks	Forest	Restaurant
8	Subway	Airport	Arena	Construction
9	Beach	Garden	Stadium	Ranch

Table 2.7: Examples of Environment (Scene)-sounds relations discovered

Environment	**Sounds**
Forest	Birds Singing, Breaking Twigs, Cooing, Falling Water
Restaurant	Jazz, Laughter, People Talking, Music Drifting
Airport	Planes Flying, Plane Engines, Aircraft, Intercoms
Park	Laughing, Police Siren, Birds Chirping, Footsteps
Ranch	Horses, Gunfire, Tapping Water, Bulldozers
Church	Children Laughing, Church Bells, Singing, Applause
Beach	Waves Crashing, Waves Lapping, Surf Hitting
Construction	Hammering, Jackhammers, Engines, Blasting
Street	Sirens, Men Shouting, Honking Cars, Cheering
Bar	Piano Playing, Laughter, Clinking Glasses, Cheering

Analysis

In our experiments, we worked with a total of 36 acoustic environments which we define in Table 2.6, but our method is generic and can work with any number of environments. Most of acoustic scenes from DCASE 2016 scene dataset [Mesaros et al., 2016] are part of our setup as well.

Table 2.7 shows a few examples of *scene-concept* relations. The full list of related sound concepts for each acoustic scene or environment are available on this webpage[6]. Table 2.7 shows a few sounds for 10 acoustic environments. A subjective analysis of all discovered relations shows that a large part of sounds discovered for a given scene are meaningful, in the sense that the sound concept is actually found in that acoustic environment. Overall, the method shows that it is possible to mine such acoustic relations from text by applying machine learning and natural language processing techniques.

2.4 Summary and Conclusions

In this chapter, we introduced natural language understanding of sounds where we developed methods for mining sound related knowledge from text. We set the goal of automatically creating a large vocabulary of sounds and then using that to obtain other sound related knowledge. We

[6]http://www.cs.cmu.edu/%7Ealnu/SOExpt.htm Copy and Paste in browser if clicking does not work

argue that this is a critical component of building an acoustically intelligent machine.

The general approach taken for finding sounds from the text was to first find potential candidates based on simple pattern search and then filter as per requirement. The filtering process can be unsupervised or supervised. We showed that our unsupervised filtering method is simple and yet extremely effective in discovering sound concepts from a large text corpus. For the initial candidate concept phrases we started with a template of "sound(s) of <X >". One can potentially use other templates as well, such as "listening to". The filtering step can also be modified. For the supervised filtering step, we can use features other than word embeddings.

Our list of over $100,000$ sounds does contain some spurious items, phrases which are not sounds; however, even with these false entries removed, we will still have the most extensive list of sounds. This vast catalog of sounds will be useful for the further advancement of acoustical intelligence in machines in a variety of ways. It can be used for building ontologies, extracting higher label semantic information, labeling and creating larger datasets, mining richer and higher level semantic information about sounds.

We then moved to the other part of natural language understanding for sounds, which is about extracting commonsense knowledge about sounds. The specific problem we worked on is to find sounds which are expected to occur in a given acoustic scene or environment. We proposed a method which treats this problem as a relation classification task. That is whether a sound is related to the given environment or not. With very minimal supervision we trained a classifier for this relation classification task. Using our method, we were able to find sounds occurring in 36 different scenes. This could be potentially extended to other scenes as well. Although not undertaken in this work, these scenes - events pairs can be useful in acoustic scene classification tasks.

Text based understanding of sounds is vital for the larger problem of automated understanding of sounds by machines. This chapter is just the beginning of natural language understanding of sounds. We expect that in future other works will explore several other problems in this domain. One such problem can be the notion of audibility in longer phrases. For instance, "a cat runs past a dog into a wall" has the constituent "cat runs into a wall", which can be expected to result in a mewling sound. "The car crashed into the tree" also evokes a perception of audibility. Knowing the notion of audibility in longer sentences will lead to a better understanding of sounds by machines. Moreover, it can also be useful for a knowledge base such as NELL, which aims to mine all kind of commonsense knowledge through the web. This can add sound specific knowledge to NELL.

On the relations front, methods for establishing other relations can be developed. Sound - source relation is one such important relationship which can be helpful in machine perception of sounds. Human beings in general always associate sounds with a source and knowing the source of a sound is crucial for the meaningful interpretation of sounds by machines. For learning sound-source relations templates of the form, GENERATES(SOUND,PHRASE) can be used. However, we can try to directly learn sound - source relations for some of the discovered sound concepts as well. Concepts such as *car honking, glass breaking* etc. clearly contain information related to the sources, and this basic pattern can be used to associate sources with sounds in a large number of cases.

Sound concepts also need to be appropriately organized and grouped. The same acoustic phenomenon may be referred to by different names. Hence, we should develop methods by which machines should know that these phrases describe the same sounds.

Furthermore, we can develop methods for the organization of sounds. For instance, all sounds originating from the same source can be grouped together. In other case, sounds occurring in an environment can be grouped together. Hierarchies can be built through these groupings. In conclusion, we believe through our work on sounds and language, we have added an essential component to acoustic intelligence in machines.

Chapter 3

Scaling Audio Event Detection

- The Promise of Weak Label Learning

Even the weak become strong when they are united.

- Friedrich Van Schiller

3.1 Introduction

Audio Event Detection (SED) or Sound Event Detection (SED) is the problem of developing computational methods for detecting sound events in an audio or video recording. The detection task, more specifically, refers to the task of temporal localization of an event in the recording, whereas the recognition (or classification) task involves finding the presence of one or more event in a given audio recording. Throughout this and further chapters we will be considering both detection and classification of sound events in audio recordings. We will mostly use the term audio event detection (AED or SED) in general discussion and specify the distinction between detection and classification as needed, for instance, while presenting methods and results where it needs to be clarified.

The field of audio event detection is relatively younger than other fields in the broad area of audio processing, for example, automatic speech recognition. In the last few years, the field has received considerable attention, and it is now growing at a good pace. The wide range of applications of automated understanding of sounds into intelligent devices has been the primary motivating factors. However, some of the early papers appeared around late 1990's [Wold et al., 1996] and several works also appeared in the early 2000's; one can then argue that the field is at least a few decades old.

Over the years, a variety of methods for audio event detection have been proposed, exploring different machine learning and signal processing techniques. All of these works employ some form of feature representation for the audio and then supervised learning methods to train event detectors or classifiers. The generic steps for training audio event detectors are shown in Figure 3.1. The signal processing part in step four plays a vital role in developing features for audio recordings. Similar to other fields such as automatic speech recognition or object recognition in

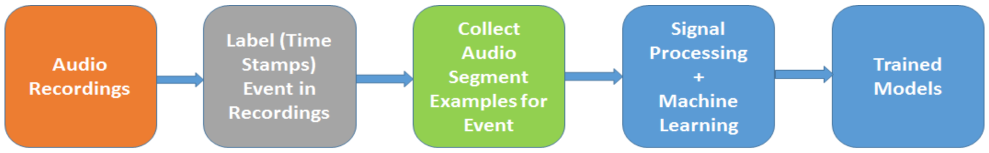

Audio Recordings
Label (Time Stamps) Event in Recordings
Collect Audio Segment Examples for Event
Signal Processing + Machine Learning
Trained Models

Figure 3.1: General Framework for Training Audio Event Detector

images, supervised learning has been in the driver's seat as far as machine learning domains are concerned. A wide range of supervised learning methods have been used, random forest, support vector machines and neural networks, to name a few. In all cases, the detectors are trained from examples of the sound to be detected. Therefore, one requires annotated data where the segments of audio containing the desired event are clearly indicated (as well as data in which the events are distinctly not present). We will refer to this type of labeling where time stamps marking the beginnings and ends of the events in the recordings are given as ***strongly labeled*** data.

This is fundamentally limiting since such well-annotated data are generally scarce and creating labeled data in those forms is very resource intensive. Since *strongly* labeled data are hard to obtain, the amount of training and test data in most cases are very small as well. This raises several issues concerning the learning process, especially that the generalization capabilities of the trained models can suffer due to small training data. Moreover, it is also a massive hindrance in scaling the AED to hundreds or potentially thousands of sound events; creating strongly labeled data for such a large number of sounds is exceptionally hard. Overall, the scale and scope of current works on AED have been limited by the lack of labeled data.

To address the challenges posed by the reliance on strongly labeled audio data, we propose weakly supervised learning for audio event detection [Kumar and Raj, 2016a,b]. This is the first work to propose weak label learning for sound events.

In ***weakly labeled*** data, the labels are available for the complete recordings. This implies that only the presence or absence of an audio event in the recordings are known. Unarguably, the annotation effort required to generate weak labels is much lesser compared to the strong labels. Hence, creating larger datasets with a considerably larger amount of labeled (weakly) audio data per event, and for a higher number of sound events is much easier in this case. The intuition behind relying on weakly labeled audio data is that recordings that have been labeled as containing the event will have regions which are consistent with one another because they contain the event but will not be consistent with any region of the recordings that are not similarly labeled. Hence, an appropriately designed machine learning and signal processing method should be able to learn from such data.

Our central idea behind learning from weakly from weakly labeled data is that we can divide a long audio recording into small segments (overlapping or not) and then devise algorithms around these segments. The constraint is that labels are available for the collection of segments and not the individual segments. We embody this principle into an algorithmic framework which falls under the general rubric of *Multiple Instance Learning*. Multiple Instance Learning (MIL) is a generalized form of supervised learning in which labels for individual instances are not known; instead, labels are available for a collection of instances, or a "bag" as it is usually called. For audio event detection using weakly labeled data, an audio recording can be considered as a bag and the segments of the recording as instances within the bag.

Moreover, using MIL framework we can also assign temporal locations to the occurrences of the events in a given audio recording. Hence, the framework can learn from training data with no temporal information about any event, but the learned models can provide temporal information about events in an audio recording. Before going into the details of the framework and the methods, we present an overview of some literature on audio event detection.

3.2 Literature on Audio Event Detection

As mentioned earlier, plenty of works on audio event detection and audio content analysis exist. One of the earliest work on content-based retrieval of audio is [Wold et al., 1996]. This system named *Muscle Fish* worked on 15 sound classes including sound events such as laughter, bells, telephone, percussion, etc. The audio recordings in this work were characterized by perceptual features, e.g. loudness, pitch, harmonicity, bandwidth. Normalized Euclidean distance or Mahalanobis measure in the nearest neighbor setting is then used for classification. Another early work [Feiten and Günzel, 1994] explored indexing of sound recordings in databases using neural nets, although not necessarily focusing on content-based analysis. [Liu et al., 1997] developed the idea of using audio information for scene classification in video recordings.

[Li, 2000] compliments the perceptual audio features in Muscle-Fish with mel cepstral coefficients (MFCCs) and uses nearest field line method [Li, 1998] for classification. They show that their method can outperform Muscle Fish system. [Guo and Li, 2003, Lu et al., 2003] used Support Vector Machines (SVMs) for classification and again combined cepstral coefficients with other acoustic features such as zero crossing rate, pitch, etc. to represent audio recordings.

The above works looked at the problem from the perspective of content-based retrieval. [Xiong et al., 2003a] can also be put in the same category, though the focus of this work is on sports videos. A later work [Huang and Cox, 2010] also explored audio events in the sports domain. The objective in these works was to detect events relevant to sports, for example, *cheering, applause* and *ball hits*.

Surveillance is another area where audio event detection found early use. Automated surveillance based on audio is advantageous as audio does not require "line of sight" and is relatively cheaper to acquire and transmit. Events such as *scream, gunshot, etc.* are central to these surveillance oriented works [Atrey et al., 2006, Clavel et al., 2005a, Pikrakis et al., 2008, Valenzise et al., 2007]. In [Valenzise et al., 2007] two parallel GMM classifiers are used for discriminating gunshots and screams from noise. The authors of [Pikrakis et al., 2008] use a Bayesian network for gunshot detection, treating it as a likelihood maximization process that is solved through dynamic programming.

In a more generic setting, taking cues from speech recognition, GMM - HMM models similar to speech recognition systems have been investigated [Mesaros et al., 2010, Zhuang et al., 2010]. Mel-Frequency cepstral coefficients (MFCCs) are most frequently used for parameterizing audio events although spectro-temporal features [Chu et al., 2009, Cotton and Ellis, 2011], Log-frequency Cepstral Coefficients, Linear Predictive Cepstral Coefficient (LPCC) [Atrey et al., 2006] under different learning frameworks have also been investigated.

Several supervised learning methods require fixed dimensional feature representation for an audio recording. A simple yet effective approach to do this is through "bags of audio words". It is essentially a population-count histogram representation – by clustering feature vectors derived from the signal to a known codebook. This approach too has been successfully applied to the problems of detecting events in audio [Kumar et al., 2013b, Pancoast and Akbacak, 2012], as well as for multimodal approaches to event detection [Ye et al., 2012] and [Wang et al., 2014b]. The bag of audio words method can be applied to a variety of low-level audio features such as MFCCs [Pancoast and Akbacak, 2012], autoencoder based features [Amid et al., 2014] and normalized spectral features [Lu et al., 2014] to name a few.

On the audio representation front, there have been attempts to obtain unsupervised representations of audio through sound units called *Acoustic Unit Descriptor* [Chaudhuri et al., 2011]. Once representations of audio recordings in terms of these acoustic unit descriptors are obtained, event detection can be done in a variety of ways. A straightforward method is to once again employ a bag of words representation and then use a classifier such as random forest [Kumar et al., 2012].

[Phan et al., 2015b] presented a random forest regression-based approach for audio event detection. The approach was also extended to early detection of audio events [Phan et al., 2015a]. Early detection of sound events is an interesting problem and can be useful in applications where we might be interested in detecting the even as soon as possible, for example in monitoring systems.

Non-negative matrix factorization (NMF) based methods have also been exhaustively explored for AED [Cotton and Ellis, 2011, Dessein et al., 2013, Gemmeke et al., 2013, Komatsu et al., 2016]. Supervectors obtained from Gaussian Mixture Models (GMMs), often used for speaker recognition and verification tasks [Bimbot and et al., 2004], have been shown to work well on the audio event and scene classification tasks also [Jin et al., 2012, Kumar et al., 2013a, Zhuang et al., 2009]. Our supervector based system was ranked among the top 7 systems in DCASE 2016 acoustic scene classification task [Elizalde et al., 2016].

The problem of polyphonic sound event detection deals with detection of multiple sound events happening simultaneously. Recent papers have attempted to address this problem through the neural network based approaches Cakir et al. [2015], Hayashi et al. [2016].

Coming to more recent developments, the success of deep neural networks (DNNs) on different tasks led to the investigation of DNNs based methods for audio event detection as well [Amid et al., 2014, Ashraf et al., 2015a, Cakir et al., 2015, Gencoglu et al., 2014, Phan et al., 2016, Piczak, 2015a, Salamon and Bello, 2017]. However, in most of these DNN based works, the amount of data used is small, and the effectiveness of deep learning methods over conventional learning approaches is arguable. More recent methods which explore large-scale audio event detection, e.g [Hershey et al., 2017], will be discussed later in this thesis.

The DCASE challenges, in 2013 [Stowell et al., 2015], 2016, 2017 and 2018 [1] have further increased the interest in the audio event and scene classification problems. A recent book [Virtanen et al., 2018], on the challenges, methods, and applications of the field of sound event and scene analysis is a good source of current literature on sound event detection. Readers can also refer to some other application oriented works mentioned in the previous chapter for more references.

Recently, some exciting works at the intersection of vision and sounds have come out. [Aytar et al., 2016] presented an approach for transferring knowledge from a network trained to recognize images to learn features for audio recordings. [Aytar et al., 2017] extended it further by learning deep features using three different modalities, image, audio, and text. Some other multimodal works have investigated the problems of locating objects in images and videos that produce sounds. In short, they try to learn correspondence between visual scenes and sounds Arandjelović and Zisserman [2017], Senocak et al. [2018]. Although, these are not specifically on the problem of sound event detection, the correspondence between vision and sound can play an important role in developing acoustic intelligence in machines.

All of the above sound event detection methods rely on fully supervised learning and hence on strongly labeled data. Each classifier/detector is trained using several clearly-demarcated

[1]http://dcase.community/

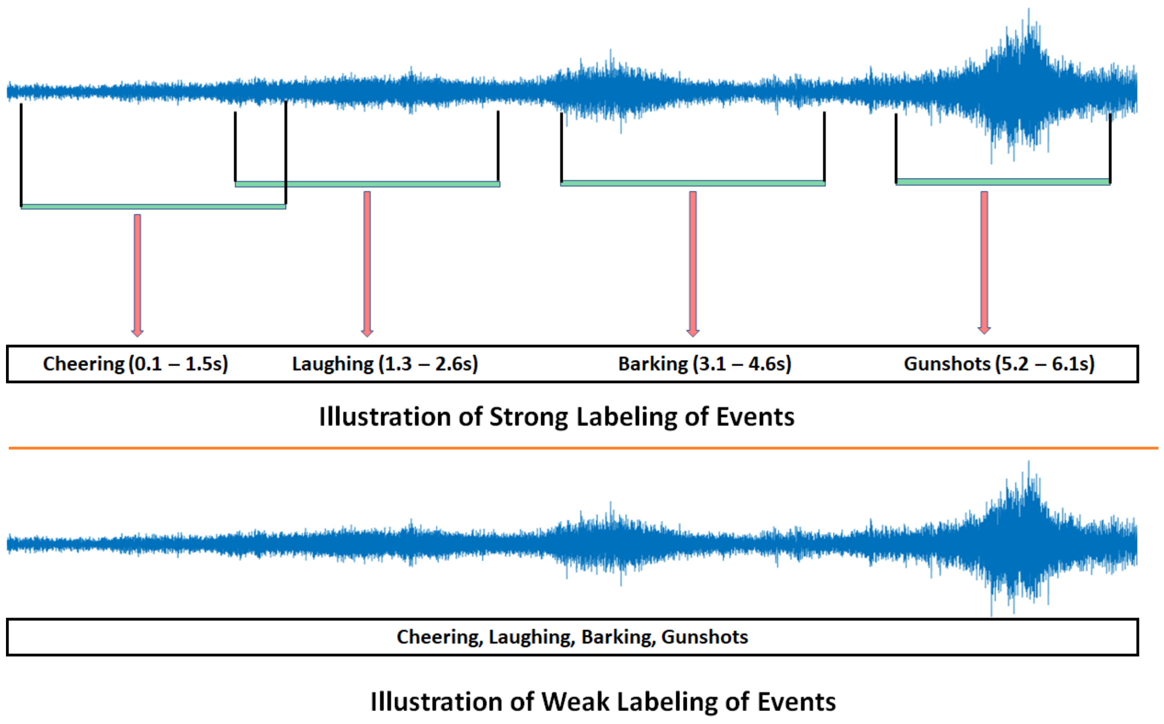

Cheering (0.1 – 1.5s)
Laughing (1.3 – 2.6s)
Barking (3.1 – 4.6s)
Gunshots (5.2 – 6.1s)
Illustration of Strong Labeling of Events
Cheering, Laughing, Barking, Gunshots
Illustration of Weak Labeling of Events

Figure 3.2: Weakly Labeled vs Strongly Labeled
Strongly labeled data contains time stamps of the occurrences of the events. Weakly labeled, on the other hand, only requires one to mark whether the event is present or not.

instances of the type of sound event it must detect, in addition to several negative instances – instances of audio segments that do not contain the event. This, as we stated before, is very hard to scale. A way to address this problem is through weakly supervised learning. Weakly supervised learning has been explored in the computer vision community to a significant extent [Cinbis et al., 2017, Duan et al., 2012b, Wang et al., 2014a], their presence in audio related tasks are more or less non-existent. A couple of prior audio-related works based around the idea of coarse labeling appeared in the domain of music processing [Mandel and Ellis, 2008, Song and Zhang, 2008]. Another audio related work exploring multi-instance multi-labeling problem is the classification of bird species based on their sounds [Briggs et al., 2012].

3.3 Weakly Labeled Learning of Audio Events

Weakly supervised audio event detection relies on weakly labeled audio recordings for training models. Weakly labeled data identifies only the presence or absence of an event in the recording; their actual location within the recordings is not marked. Figure 3.2 illustrates strong vs. weak labeling of events in audio recordings.

Strong labeling, where we need to mark temporal boundaries of the events is naturally a difficult and resource intensive task to do; it is time-consuming as well as an expensive process. Often, one must go back and forth several times in a recording to mark the beginnings and ends of the audio events. Overlapping events and noise can make the task even harder.

Another issue with strong labeling is a subjective one; which is the different interpretation of

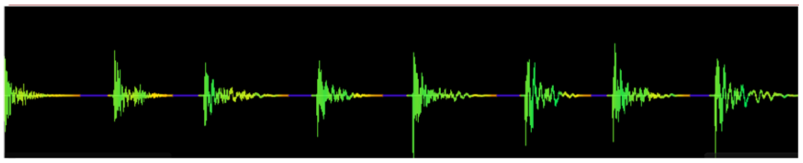


Figure 3.3: A Recording of Footsteps
Different annotators might mark different numbers of beginning and ends.

sounds by different annotators. Consider, for example, an audio recording of Footsteps sounds shown in Figure 3.3. How many beginnings and ends should be marked for this recording: one, two, four, or eight? Different annotators interpret the event differently and assign different beginnings and ends to the footsteps sound. Besides making the annotation task harder, this also creates unnecessary variability in the event exemplars, which is likely to confuse any algorithm that attempts to learn the underlying structure and composition of the audio event.

Weak label learning addresses the issues with strongly labeled data. Since timestamps need not be marked, weak labeling is relatively much easier to do and hence we can think about scaling audio event detection. Moreover, the event when present in the recording is available to the learner in its entire natural form, without any assumption or bias about the beginning and ends which might get introduced by the annotator.

However, the most significant advantage of the weakly supervised learning is that it opens up the possibility of exploiting the massive amount of audio (multimedia) data on the web. Most of the audio data on the web have some associated metadata, from which recording level labels or weak labels can be inferred. This can completely remove the manual annotation process and can potentially scale audio event detection to hundreds or thousands of hours of weakly labeled audio data. We next formulate the problem of audio event detection using weakly labeled data and then present our framework for the same.

3.3.1 Problem Formulation

Before going further, we would like to define some nomenclature which will be used throughout the thesis. By the term *Recording*, we will imply the full audio recording in its entirety, it can be just a few seconds long, or it can be several minutes long. *Segments* refers to short duration slices of the recordings, 1 to 1.5 seconds in most cases in this work.

Our objective is to train the detector models using *weakly-labeled audio data* which comprise of recordings in which only the presence or absence of these events are identified, without indicating the exact location of the event or even the number of times it has occurred. Let $R = \{R_i : i = 1 \textit{ to } N_R\}$ be the collection of audio recordings and $E = \{E_i : i = 1 \textit{ to } N_E\}$ be the set of events for which detection models must be built using R.

For each R_i, a certain subset of events from E are known to be present (weak-labels). For example, the information might say that R_i contains events E_1, E_3 and E_6. It is also possible that the subset of E present in R_i is empty, meaning that no event from E is present in R_i.

Clearly, to train a detector for an event E_i one cannot just use all recordings that are marked as containing E_i, since a significant portion of the marked recordings might also contain other events. Moreover, since the start and end times of the occurrences of event E_i in the recordings are not known, it is impossible to extract out the specific segment of the audio that contains the event for further use in supervised learning. Conventional supervised learning is thus not possible.

To solve this problem, our main idea is that instead of looking at an audio recording in its entirety, we should look at it at the segment level. Essentially, we divide the whole audio recording into multiple segments and then work with these segments. So, a weakly labeled recording gets transformed into a collection of audio segments, and the label for this collection of audio segments with respect to an event is known. We argue that learning with this labeled collection of audio segments can be formulated as Multiple Instance Learning. We present Multiple Instance Learning next and then later formalize audio event detection using weakly labeled data through multiple-instance learning.

3.4 Multiple Instance Learning

Multiple Instance Learning (MIL) is a weak form of supervised learning and was first developed in the context of drug activity detection by [Dietterich et al., 1997]. Unlike fully supervised learning which is defined in terms of instance-label pairs, MIL is described in terms of *bags*; bags are simply a collection of data points or instances. Bags are the primary units in MIL and labels are attached to the bags, rather than to the individual instances within them. A *negative* labeled bag contains negative instances only. However, a *positive* bag is one which has at least one positive instance (an instance from the target class to be classified). More informally, we can say a negative bag is "pure" whereas a positive bag is "impure". This generates an asymmetry from learning perspective as all instances in *negative bags* can be uniquely assigned negative label whereas for the *positive bags* the same does not apply; an instance in a *positive bag* may either be positive or negative. Thus, it is the *bag-label* pairs, rather than instance-label pairs, which form the training data from which a classification model must be learned.

We represent the bag-label pairs as (B_i, Y_i). Here B_i is the i^{th} bag and contains instances x_{ij} where $j = 1 \textit{ to } n_i$, and n_i is the total number of instances in the bag, *i.e.* $B_i = \{x_{ij} : j = 1 \cdots n_i\}$. $Y_i \in \{-1, 1\}$ is the label for bag B_i. $Y_i = 1$ implies a positive bag and $Y_i = -1$ implies a negative bag. Let the total number of bags be indicated by N. One can attempt to infer labels of individual instances x_{ij} in the bag from the bag label. The label y_{ij} for instances in bag B_i can be stated as:

$$Y_i = -1 \Rightarrow y_{ij} = -1 \ \forall \ x_{ij} \in B_i, \tag{3.1}$$

$$Y_i = 1 \Rightarrow y_{ij} = 1 \text{ for at least one } x_{ij} \in B_i \tag{3.2}$$

This relation between Y_i and y_{ij} is simply

$$Y_i = \max_j \{y_{ij}\}. \tag{3.3}$$

The goal in MIL is to learn a decision or classification model given a set of bag-label pairs. Several methods have been proposed to solve MIL problem Learning Axis-Parallel Concepts [Dietterich et al., 1997], Diverse Density[Maron and Lozano-Pérez, 1998], Citation-KNN[Wang and Zucker, 2000], mi-SVM and MI-SVM [Andrews et al., 2002] miGraph[Zhou et al., 2009],MILES [Chen et al., 2006]. Here, we discuss methods based on Support Vector Machine and Neural Networks which we employ for audio event detection tasks.

3.4.1 MIL using Support Vector Machines

The MIL problem can be solved using Support Vector Machines (SVM) in two forms. The first form called *mi-SVM* simply imposes *at least one positive instance in a positive bag* constraint in the conventional SVM optimization problem. The second form, called *MI-SVM*, redefines the margin formulation in SVM to account for the fact that "a positive bag contains at least one positive". In other words, it defines the margin for each bag through a single instance from the bag, and then margin maximization is done with respect to that selected instance. We present both methods here.

mi-SVM

To understand mi-SVM, let us first formalize the "*at least one positive instance in a positive bag*" constraint in mathematical form. The bag label Y_i and instance labels y_{ij} relations in Eq 3.1-3.2 can be represented in the form of linear constraints.

$$\sum_{j=1}^{n_i} \frac{y_{ij}+1}{2} \geq 1 \;\forall\; i \;s.t\; Y_i = 1, \;;\; y_{ij} = -1 \,\forall\; i \;s.t\; Y_i = -1 \tag{3.4}$$

Note that, the instance labels in the constraints defined above are unobserved hidden variables. As in conventional training of SVMs, where we must estimate the parameters of a linear (or kernelized) function such that the *margin* of training instances from the discriminant function is maximized, here too we must maximize the margin. However, since the instance labels are unknown, we modify the objective: the goal now is to maximize a *soft margin* over both the decision function and the hidden integer variables, namely the unknown labels of instances. This is done by including the constraints in Eq 3.4 in the optimization problem of SVM

$$\min_{y_{ij},\mathbf{w},b,\xi} \frac{1}{2}||w||^2 + C\sum_{ij} \xi_{ij} \tag{3.5}$$

such that

$$\forall i,j \;:\; y_{ij}(\langle \mathbf{w}, \mathbf{x}_{ij}\rangle + b) \;\geq 1 - \xi_{ij} \tag{3.6}$$

$$\xi_{ij} \;\geq 0 \;; y_{ij} \;\in\; \{-1,1\} \;; y_{ij} \;=\; -1 \,\forall\; i \;s.t\; Y_i = -1 \tag{3.7}$$

$$\sum_{j=1}^{n_i} \frac{y_{ij}+1}{2} \geq 1 \;\forall\; i \;s.t\; Y_i = 1 \tag{3.8}$$

In equation 3.5, the labels y_{ij} of instances belonging to positive bags are unknown integer variables. As a result both, the optimal labeling of these instances as well as the optimal hyperplane $(\mathbf{w}, b)$, must be computed. The separating hyperplane must be such that there is at least one pattern from every positive bag in the positive half space and all patterns belonging to negative bags are in the negative half space.

The above formulation is, however, a difficult mixed integer problem to solve. Hence, a simple heuristic is used to solve the optimization problem. The main idea is that for given integer variables , *i.e.* fixed labels, it can be solved exactly through the usual quadratic programming as in conventional SVM. Once a solution is known it can be used to impute labels to instances in positive bags to ensure that constraints are satisfied. The solution thus is a two-step iterative process:

- *Step 1:* Given the integer variables (fixed labels) solve the standard SVM.
- *Step 2:* Given the SVM solution, impute the integer label variables for the positive bags.

Algorithm 1 mi-SVM Algorithm

```
1: procedure LEARNING SVM IN MIL SETTING($B_i$, $Y_i$) // Input training bags and labels

   //Initialization Step
2:     $y_{ij} = Y_i$ for all j in Bag $B_i$
3:     repeat
4:         compute SVM solution $\mathbf{w}$, $b$ with imputed labels
5:         for all bag $B_i$ s.t $Y_i = 1$ do
6:             compute $f_{ij} = (\langle \mathbf{w}, \mathbf{x}_{ij} \rangle + b)$ $\forall$ $x_{ij}$ in $B_i$
7:             $y_{ij} = sgn(f_{ij})$ $\forall$ $j$ in bag $B_i$ s.t $Y_i = 1$
8:             if $(\sum_{j=1}^{n_i} \frac{y_{ij}+1}{2} == 0)$ then
9:                 compute $j^* = \arg\max_j (f_{ij})$
10:                set $y_{ij^*}$=1
11:            end if
12:        end for
13:    until imputed labels no longer change
14: end procedure
```

The labels of instances in the positive bags are all initialized as positive and are updated as described above. The instances in negative bags are obviously labeled negative and remain so through out the procedure. The two steps are iterated until no changes in labels occur. The overall mi-SVM is shown in Algorithm 1. If it happens during an iteration that all instances in a positive bag are labeled as negative, then the one with the maximum value for decision function is assigned a positive label. This process is repeated until no change in imputed labels is observed.

MI-SVM

The second form of SVM for MIL, MI-SVM, takes a different view of the MIL problem. Here, the margin is described directly in terms of bags. A bag is represented or "witnessed" by a single instance for which maximal output is observed and margin is determined by this instance. Thus, the optimization problem becomes

$$\begin{aligned} &\min_{\mathbf{w},b,\xi} \frac{1}{2}||w||^2 + C\sum_i \xi_{ij} \\ &s.t\ \ Y_i(\max_{\mathbf{x}_{ij}\in B_i}(\langle \mathbf{w}, \mathbf{x}_{ij}\rangle + b)) \geq 1 - \xi_i,\ \ \xi_i\ \geq 0 \end{aligned} \tag{3.9}$$

Optimization heuristic similar to miSVM is used to solve the problem. The only difference is that in this case a selector variable which determines which instance of a positive bag is "witness" is updated in each iteration. The process is repeated until no change in "witness" instance is observed for all bags.

3.4.2 MIL using Neural Networks (NN-MIL)

Neural networks can also be adapted for MIL domain [Zhou and Zhang, 2002]. In the conventional training of neural networks, instance-specific labels for a collection of training instances are available. Training is performed by updating network weights to minimize the average divergence between the actual network output in response to these training instances and the desired output, typically some representation of their assigned labels [Rumelhart et al., 1988][Werbos, 1982].

In the MIL setting, where only bag-level labels are provided for the training data, this procedure must be appropriately modified. In order to do so, the divergence to be minimized must be computed by utilizing only bag-level labels. Once again the idea of representing a bag through instance with maximal output is exploited [Dooly et al., 2003]. For convenience in discussion let us assume that bag labels in this case takes values 0 for the negative bags. Let o_{ij} represent the output of the network in response to input x_{ij}, the j^{th} instance in B_i, the i^{th} bag of training instances. We define the *bag-level* divergence for bag B_i as

$$E_i = \frac{1}{2}\left(\max_{1\leq j\leq n_i}(o_{ij}) - d_i\right)^2 \tag{3.10}$$

where d_i, the desired output of the network in response to the set of instances from B_i, is simply set to Y_i, the label assigned to B_i. Thus for positive bags $d_i = 1$, whereas for negative bags $d_i = 0$. The central idea behind the bag-level divergence of Equation 3.10 is to refer to any bag using the instance which produces the maximal output. This was proposed by Dooly *et al.* in [Dooly et al., 2003] where they showed that irrespective of the number of instances (positive or negative) in a bag, the bag can be adequately described by the instance with the maximal output.

The bag-level divergence of Equation 3.10 may be understood by noting that the term "$\max_j o_{ij}$" in it effectively represents the *bag-level output* of the network, and that Equation 3.10 simply computes the divergence of the bag-level output with respect to the bag-level label of Equation 3.3.

The ideal output of the network in response to any negative instance is 0, whereas for a positive instance it is 1. For *negative* bags, Equation 3.10 characterizes the *worst-case* divergence of all instances in the bag from this ideal output. Minimizing this effectively ensures that the response of the network to *all* instances from the bag is forced towards 0. In the ideal case, the system will output 0 in response to all inputs in the bag, and the divergence E_i will go to 0.

For *positive* bags, on the other hand, Equation 3.10 computes the *best-case* divergence of the instances of the bag from the ideal output of 1. Minimizing this ensures that the response of the network to *at least* one of the instances from the bag is forced towards 1. In the ideal case, one or more of the inputs in the bag will produce an output of 1 and the divergence E_i will go to zero.

The *overall* divergence on the training set is obtained by summing the divergences of all the bags in the set:

$$E = \sum_{i=1}^{N} E_i = \sum_{i=1}^{N}\frac{1}{2}\left(\max_{1\leq j\leq n_i} o_{ij} - d_i\right)^2 \tag{3.11}$$

The parameters of the network are trained using conventional backpropagation, with the difference that we now compute gradients of the divergence given in Equation 3.11, and that entire bags of data must be processed prior to updating the network parameters. During training once all instances in a bag have been fed forward through the network, the weight update for the bag is done with respect to the instance in the bag for which the output was maximum. The process is continued until the overall divergence falls below a desired tolerance.

Prediction using a trained network can now be done instance-wise as is done in standard feed-forward neural networks. Bag labels, if required, can be predicted based on the label obtained for the maximal-scoring instance in the bag.

3.5 MIL for Weakly Labeled AED

The recording level labels can be easily converted into bag-label pairs for multiple instance learning. To convert recording R_i to a bag, it is segmented into many short audio segments. Adjacent segments may overlap by design. Let the segments derived from R_i be $[I_{R_i^1}, I_{R_i^2}, \ldots, I_{R_i^K}]$. Each of these smaller segments is now treated as an individual instance within the bag R_i. Formally, a bag R_i is thus $R_i = \{I_{R_i^1}, I_{R_i^2}, ..., I_{R_i^K}\}$, where K depends upon the duration of recording R_i, the length of the individual segments, and the overlap between adjacent segments. If the weak label states that E is present in a recording R_i then R_i is a positive bag; otherwise, it is a negative bag. We are making an assumption that if the event E_i is present in R_i then, at least one of the segments contains the event and is a good representation of it. Hence, R_i will be a positive bag for the event. On the other hand, if an event is marked as not being present in a recording, then clearly *none* of the segments (instances) of the recording will be positive for that event, and hence overall that recording is a negative bag for the event. Hence, the weak labels for all the recordings can directly provide bag-label representations needed in MIL. Once bag-level pairs are created we can use any MIL method.

3.5.1 Temporal Localization of Events

Often, it is important to localize an event within the recording instead of just predicting whether the event is present in the recording or not. The MIL methods we discussed learn from bag-level labels; but once learning is complete, they can classify individual instances. Therefore, not only can we detect the presence of an event in a test recording (bag) but also in individual segments of the test recording.

Formally, if R_x is a test recording, which we may more explicitly represent as $R_x(t)$, where "t" indexes time. The individual segments in the recording as we described before are $I_{R_x^1}, I_{R_x^2}, ..., I_{R_x^K}$. The segment $I_{R_x^k}$ is the portion of R_x given by $I_{R_x^k} = R_x(t), \quad (k-1)l' \leq t < (k-1)l' + l$. Here, l is the length of the segment in seconds and l' denotes the amount by which the segment window is shifted with respect to the previous segment. In the special case of non-overlapping segments, $l' = l$. If an event E_i is detected in segment $I_{R_x k}$ it means this event can be localized to the time segment $\Big((k-1)l', \ (k-1) * l' + l\Big)$ in the recording R_x. The framework can thus generate information about the temporal location of events, and thus, we can obtain a complete description of the recording in terms of audio events. This is significant since the descriptive form of labeling was never present in the training data in the first place.

3.6 Experiments and Results

3.6.1 Features for Audio Segments

Before moving on to the evaluation of weakly supervised learning for AED, we describe the feature representation used for characterizing each audio segments. We use Mel-frequency cepstral coefficients (MFCCs) vectors to obtain low-level feature representations for audio segments. However, direct characterization of audio as sequences of MFCC vectors tends to be ineffective for the purposes of audio classification [Zhuang et al., 2010]; other secondary representations derived from these are required. One simple and yet very successful approach is the Bag of Audio Words feature representation, which quantizes the individual MFCC vectors into a set of codewords, and represents segments of audio as histograms over these codewords. For short audio segments, a more effective representation is obtained by using Gaussian Mixture Models (GMMs) to represent audio words and derive features based on it [Kumar et al., 2013b]. A robust feature representation is obtained by combining two features derived from GMM audio words. The first, which we represent as $\vec{F}$, is similar to bag-of-words characterizations such as [Van Gemert et al., 2010], and the second, which we represent as $\vec{M}$ is a characterization of the modes of the distribution of vectors in the segment.

As a first step to obtaining the $\vec{F}$ and $\vec{M}$ feature vectors for the audio segments, we train a *universal* Gaussian mixture model (GMM) on MFCC vectors from training audio data. This background GMM is used to extract the $\vec{F}$ and $\vec{M}$ features. Let us represent this universal GMM as $\mathcal{G} = \{w_k, N(x; \lambda_k)\}$, where w_k is the *a priori* probability or mixture weight of the k^{th} Gaussian in the mixture, $N(.)$ represents a Gaussian, and λ_k collectively represents the set of mean and covariance parameters of the k^{th} Gaussian.

$\vec{F}$ Features

For each audio segment we have a sequence of D-dimensional MFCCs vectors denoted by $\vec{x_t}$ where $t = 1$ *to* T. T is the total number of MFCC vectors for the given segment. For each component k of the background GMM we compute

$$Pr(k|\vec{x_t}) = \frac{w_k N(\vec{x_t}; \lambda_k)}{\sum\limits_{j=1}^{G} w_j N(\vec{x_t}; \lambda_j)}, \tag{3.12}$$

$$F(k) = \frac{1}{T} \sum_{i=1}^{T} Pr(k|\vec{x_t}) \tag{3.13}$$

The $\vec{F}$ feature vectors are $\vec{F} = [F(1), F(2), \cdots, F(G)]^{\top}$. Thus, $\vec{F}$ is G-dimensional vector representing a normalized *soft-count* histogram of the MFCC vectors in the recording. It captures how the MFCC vectors are distributed across components of $\mathcal{G}$. It is a variant of bag of audio word features where soft assignment is used in place of hard quantization.

$\vec{M}$ Features

A more detailed characterization can be obtained by actually representing the distribution of the feature vectors in the segment. To do so, we train a separate GMM for each audio segment by adapting the universal GMM $\mathcal{G}$ to the collection of MFCC vectors in the segment. The means of the universal GMM are adapted to each training segment using the maximum *a posteriori* (MAP) criterion as described in [Bimbot and et al., 2004]. This is done as follows for k^{th} component of the mixture

$$n_k = \sum_{t=1}^{T} Pr(k|\vec{x_t}), \tag{3.14}$$

$$E_k(\vec{x}) = \frac{1}{n_k} \sum_{t=1}^{T} Pr(k|\vec{x_t})\vec{x_t} \tag{3.15}$$

Finally, the updated means are computed as

$$\hat{\vec{\mu_k}} = \frac{n_k}{n_k + r} E_k(\vec{x}) + \frac{r}{n_k + r} \vec{\mu_k} \tag{3.16}$$

Table 3.1: Number of Positive Bags for each event

Event	Number of Bags	Event	Number of Bags
Cheering	171	Engine Noise	80
Children's Voices	33	Hammering	17
Clanking	13	Laughing	116
Clapping	102	Marching Band	24
Drums	25	Scraping	30

where $\vec{\mu_k}$ is the mean vector of k^{th} Gaussian and r is a relevance factor. The value of r controls the contribution of the original mean ($\vec{\mu_k}$) to the adapted mean. The means of all components are then appended to form the $G \times D$ vector $\vec{M}$ as $\vec{M} = [\hat{\vec{\mu_1}}^\top, \hat{\vec{\mu_2}}^\top, \cdots \hat{\vec{\mu_G}}^\top]^\top$.

The above two features are unsupervised methods of characterizing the statistically characterizing audio segments. $\vec{F}$ is a softer version of the bag of words features and are often used as standalone features. $\vec{M}$ is more detailed, though it is also more easily affected by inter-instance variability. Together, they give us a robust representation of both the coarse and fine structure of the signals in short audio segments. In our experiments, we have used $\vec{F}$ both in combination with $\vec{M}$ and as a standalone feature.

Once we have a robust set of features for representing events in short audio segments we can extract features for each audio segment (instances) of a recording (bag). Thus, a recording (bag) $R_i = \{I_{R_i1}, I_{R_i2}, ...I_{R_iK}\}$ in feature space becomes $R_i = \{\vec{x}_{R_i1}; \vec{x}_{R_i2}; ....\vec{x}_{R_iK}\}$ where $\vec{x}_{R_ij}$ are either the $\vec{F}$ features alone or the concatenated $\vec{F}$ and $\vec{M}$ vectors. The bags are then fed into the MIL algorithms.

3.6.2 Experimental Setup

We evaluated the proposed MIL framework on a portion of the TRECVID-MED 2011 database [32]. A subset of the MED dataset is annotated with ten audio events (Table 3.1). To be able to compare performance with the fully supervised case and to compute performance measures for temporal localization of events, our annotations include actual locations and duration of occurrences of these events. However, only the information regarding the presence or absence of these sounds is used in our MIL based framework. A total of 457 recordings (bags) are used in the experiments. This is over 22 hours of audio data. We henceforth refer to this set of recordings as the "dataset". The length of each recording in the dataset varies from a few seconds to several minutes with an average length of about 2.9 minutes. This implies that the number of instances in each bag also has a wide range.

Ideally, the length of each segment should be properly set keeping in mind the expected duration of the event of interest. However, we observed that segment length decided by heuristics work well. We report results using a segment length fixed to 1 second. The segments overlap by 50%. This results in well over 150,000 total instances. The names of the ten events and the total number of positive bags for each event are given in Table 3.1. Some of the recordings in the dataset do not contain any of the ten events; also a recording might be a positive bag for more than one event. Hence, the sum total of numbers in Table 3.1 is different from the total number of recordings in the dataset. For each event recordings where it is not present are used as the negative bags. The dataset is partitioned into four sets. Three of the sets were used to train the models, which were then tested on the fourth set. This is done in all four ways meaning each set

Table 3.2: AUC comparison with supervised SVM

Events	AUC ($miSVM$)	AUC ($supSVM$)
Cheering	0.632	0.682
Children's Voices	0.678	0.668
Clanking	0.714	0.727
Clapping	0.646	0.697
Drums	0.60	0.640
Engine Noise	0.623	0.671
Hammering	0.557	0.568
Laughing	0.527	0.741
Marching Band	0.551	0.558
Scraping	0.723	0.850
Mean	**0.625**	**0.680**

becomes a test set. This gives us results on the whole dataset. Hence, all results reported here are on the entire dataset.

All recordings were parameterized into sequences of 21-dimensional Mel Frequency Cepstrum Coefficient (MFCC) vectors. MFCC vectors were computed over analysis frames of 20ms, with an overlap of 50%(10ms) between adjacent frames. The $\vec{F}$ and $\vec{M}$ features were derived from these sequences of MFCC vectors. We trained two background GMMs with 64 and 128 Gaussian components respectively. The number of Gaussian component in the features is represented as the subscript in the feature such as $\vec{F}_{64}$, $\vec{M}_{64}$.

ROC curves are used to analyze the performance. AUC of ROC curves are used to compare results in different cases. Higher AUC values indicate better detection performance. We first show results for recognition of events at recording (bag) level using miSVM and NN-MIL. The instance level results or detection results measuring temporal localization of events are provided in Section 3.6.4.

3.6.3 Results

miSVM Results

For the miSVM framework linear SVMs are used in all experiments. LIBLINEAR [Fan et al., 2008] is used in the implementation of the miSVM framework. The slack parameter C in the SVM formulation is tuned by cross validation over the training set. The mean AUC (MAUC) over all events is shown in the last row of each table.

Comparison with supervised SVM

We start by showing the comparison of our proposed framework with fully supervised AED where the strong labels are available. For supervised learning, the timestamps in the annotations are used to obtain pure examples of each event, following which an SVM is trained using feature representations of these examples. Table 3.2 shows this comparison. The comparison is shown for $\vec{F}_{64}$ features. In Table 3.2 "*supSVM*" refers to supervised SVM. As is expected, supervised SVMs perform better than *miSVM*. However, there are several events for which the performance obtained with weak labels is comparable to that obtained with the strong labels. Although supervised SVM outperforms *miSVM*, miSVM can be easily scaled to a larger dataset which will

Table 3.3: AUC for different number of components in GMM (miSVM)

Events	AUC ($\vec{F}_{64}$)	AUC ($\vec{F}_{128}$)
Cheering	0.632	0.638
Children's Voices	0.678	0.633
Clanking	0.714	0.744
Clapping	0.646	0.667
Drums	0.60	0.636
Engine Noise	0.623	0.642
Hammering	0.557	0.587
Laughing	0.527	0.540
Marching Band	0.551	0.554
Scraping	0.723	0.735
Mean	**0.625**	**0.637**

Table 3.4: Effect of $\vec{M}$ features addition (miSVM)

Events	AUC ($\vec{F}_{64}$)	AUC ($[\vec{F}_{64}, \vec{M}_{64}]$)
Cheering	0.632	0.668
Children's Voices	0.678	0.723
Clanking	0.714	0.859
Clapping	0.646	0.680
Drums	0.60	0.639
Engine Noise	0.623	0.575
Hammering	0.557	0.660
Laughing	0.527	0.641
Marching Band	0.551	0.745
Scraping	0.723	0.744
Mean	**0.625**	**0.693**

improve performance and model generalization. However, the same cannot be easily done for supervised SVMs as obtaining *strongly* labeled data is extremely difficult.

Number of Gaussian Components

Table 3.3 shows AUC results for $\vec{F}_{64}$ and $\vec{F}_{128}$ features, obtained by training a 64 and 128 Gaussian GMM $\mathcal{G}$, respectively. It can be noted that there are several events for which increasing the number of Gaussians leads to about $2 - 4\%$ (up to 5.5% relative) absolute improvement in AUC values. At the same time, there are events such as *Cheering* and *Marching Band* where this improvement is not observed, or the performance goes down as in *Children's Voices*. A performance drop of about 4% is observed in this case. The optimal cluster size is known to be event specific [Kumar et al., 2013a], and this holds for MIL-based audio event detection as well.

Events	AUC
Cheering	0.668
Children's Voices	0.730
Clanking	0.859
Clapping	0.680
Drums	0.639
Engine Noise	0.642
Hammering	0.660
Laughing	0.685
Marching Band	0.745
Scraping	0.744
Mean	**0.704**

Table 3.5: Overall Results (miSVM)

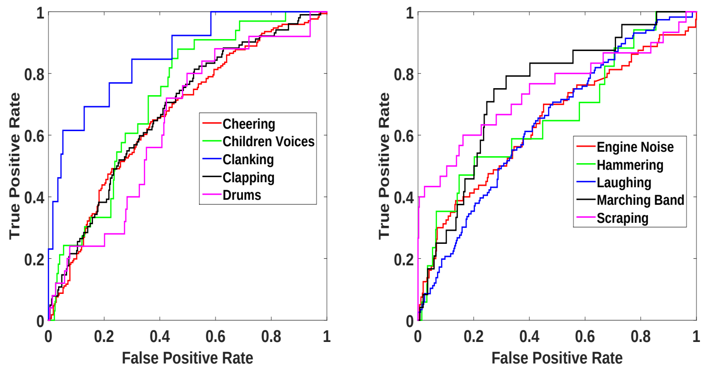


Figure 3.4: ROC Curves for using miSVM framework

Adding $\vec{M}$ Features

We now observe the effect of adding the $\vec{M}$ features to the system, along with $\vec{F}$. Table 3.4 shows a comparison of AUC values when only $\vec{F}$ is used and when it is combined with $\vec{M}$ features. The results shown are obtained with 64 Gaussian components in the GMM. It can be observed that adding $\vec{M}$ features obtained by *maximum a posteriori* adaptation leads to a considerable improvement of results for almost all events. Events for which $\vec{F}$ features alone results in poor performance such as *Hammering*, *Laughing* and *Marching Band* benefit significantly from the $\vec{M}$ features. Absolute improvements of 10.3%, 11.4% and 19.4% respectively are observed for these three events. This amount to around 18.5%, 21.6% and 35.2% relative improvements. For other events too absolute improvements in the range of $2.1\% - 14.5\%$ can be noted. The only exception is *Engine Noise* for which the *soft-count* $\vec{F}$ seems to be better. It is likely that although we observe improvements in miSVM, the actual improvements may be classifier dependent.

mi-SVM Recording Level Results

The overall bag-level recognition results using miSVM are shown in Table 3.5. This is the best result across different feature representations for audio segments. The corresponding ROC curves are in Figure 3.4. Events such as *Clanking*, *Children's Voices*, *Scraping* and *Marching Band* are easier to detect compared to other events such as *Drums*. The mean AUC over all events is 0.704 which validates the success of our proposed framework. A mean AUC of around 0.7 shows that audio event detects can indeed be trained using just weakly labeled data. Better results can be obtained by using a larger amount of training data which is much easier to obtain in the weakly labeled scenario.

NN-MIL Results

For the NN-MIL method, the number of hidden layers, the number of nodes in each hidden layer (n_{no}) needs to be set. We used a network with one hidden layer in all experiments. The network is trained for a total of 60 epochs. The learning rate is either fixed at 0.1 throughout training or 0.1 for the first 30 epochs and then reduced in each epoch until it reaches 0.01. Larger values of n_{no} are used for higher dimensional of input features. For $\vec{F}_{64}$ and $\vec{F}_{128}$ features 3 different

Events	AUC
Cheering	0.759
Children's Voices	0.767
Clanking	0.764
Clapping	0.781
Drums	0.601
Engine Noise	0.698
Hammering	0.603
Laughing	0.632
Marching Band	0.618
Scraping	0.785
Mean	**0.701**

Table 3.6: Overall Results (NN-MIL)

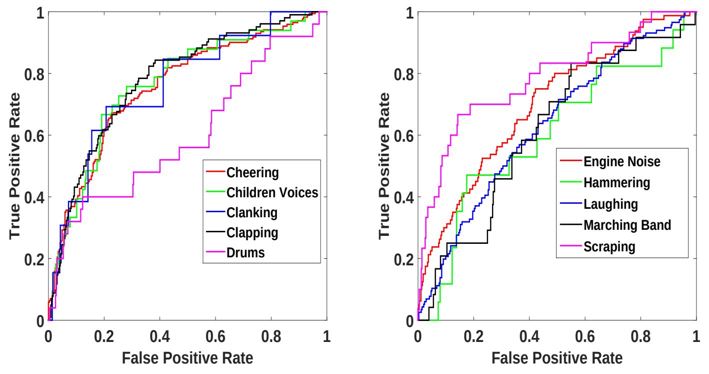


Figure 3.5: ROC Curves for using NN-MIL framework

values of n_{no} are used. These are $16, 50$ and 100 for $\vec{F}_{64}$ and $50, 100$ and 150 for $\vec{F}_{128}$. When both $\vec{F}$ and $\vec{M}$ are used, the values of n_{no} used in the experiments are 256 and 512. Training neural networks, in general, requires exhaustive tuning of parameters to get good results. Although results presented here show reasonable performance for the NN-MIL framework, we believe that better results can be obtained by more aggressive parameter tuning. In fact, parameter tuning may give better insight into the NN-MIL framework.

We present only the best results across the different settings. Table 3.6 shows overall results for NN-MIL framework. The corresponding ROC curves are shown in Figure 3.5. If we compare these results with the miSVM approach, we can observe that events such as *Scraping*, *Clanking* and *Children's Voices* are easier to detect in this case as well. Events such as *Drums* and *Hammering* are harder to detect using NN-MIL as well. The mean AUC remains almost same as miSVM; however, one can note a significant difference with respect to miSVM for several events. For *Cheering, Children's Voices, Clapping, Engine Noise*, NN-MIl is better than mi-SVM whereas for the rest miSVM is better. An analysis on a larger vocabulary of events might help differentiate the two approaches more clearly.

3.6.4 Temporal Localization of Events

To evaluate the performance on temporal localization task, we need the ground truth labels of all instances in all bags. The instances in the bags have been obtained through uniform segmentation in this work. Each instance is a one-second window segment of the recording which is moved in an overlapping manner to segment the recording. However, the annotations providing time stamps of events in the recording does not adhere to this uniform segmentation. Thus an event might start and end within a segment, and it can also start or end at any point in the segment. Hence, assigning ground truth labels of instances for a valid analysis is not straightforward. We use a simple heuristic to obtain ground truth labels. As described in Section 3.5.1 each segment represents a specific time duration of the recording. Looking into the actual annotations available, if an event can be marked to be present in at least 50% of the total length of the segment we call the ground truth label of that segment as positive; otherwise, it is negative.

Once the ground truth labels with respect to an event have been obtained for all instances for all bags, we can analyze the performance in the usual fashion. We again present ROC

Table 3.7: AUC for temporal localization of events

Events	AUC (miSVM)	AUC (NN-MIL)
Cheering	0.588	0.669
Children's Voices	0.665	0.705
Clanking	0.925	0.645
Clapping	0.585	0.626
Drums	0.680	0.628
Engine Noise	0.603	0.652
Hammering	0.542	0.572
Laughing	0.548	0.581
Marching Band	0.758	0.701
Scraping	0.684	0.724
Mean	**0.658**	**0.650**

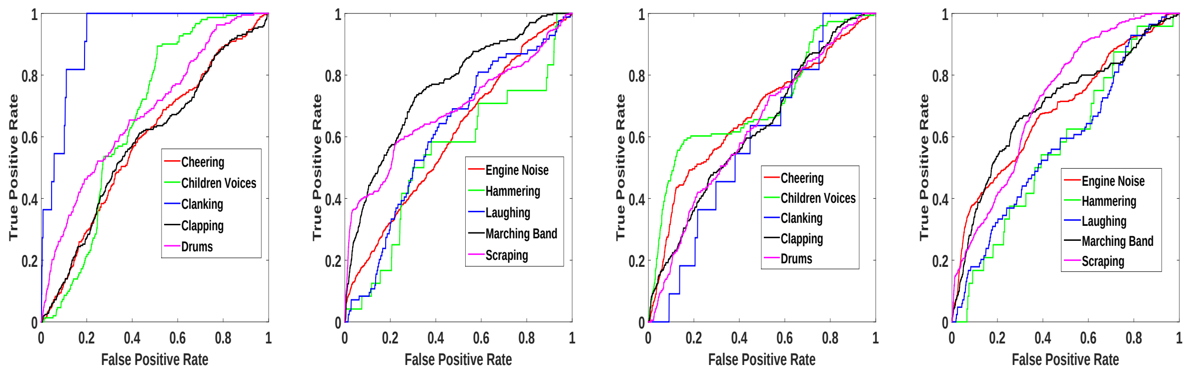


Figure 3.6: ROC Curves for Temporal Localization (First Two-miSVM, Last Two - NN-MIL)

curves and use AUC as the metric characterizing these curves. The best AUC values across all experiments for temporal localization using both *miSVM* and *NN-MIL* are shown in Table 3.7. The corresponding ROC curves are shown in Figure 3.6. The figures in the upper row are for *miSVM*, and the figures in the bottom row are for NN-MIL. Compared to bag-level results, about 5% drop in mean AUC is observed for both cases. For some events such as *Hammering* and *Laughing*, the performance is poor for both frameworks. For others, reasonable performance is obtained. Although these numbers are not exceptionally high, they are still significant since no temporal information was used during the training stage. Overall, AUC results validate that our proposed framework can work for temporal localization as well.

3.7 Scalable MIL Methods

In the previous sections, we presented MIL based methods for audio event detection using weakly labeled data. Other MIL algorithms can also be used. However, MIL algorithms have been known to suffer from scalability issues [Wu et al., 2017]. For example, the SVM based MIL methods are iterative methods and a quadratic program is solved in each iteration. Scalability is an important factor especially when we are trying to work on problems such as AED where the total number of instances will become very large even for a few hours of audio data. The primary reason MIL algorithms are time-consuming and do not scale to large data is that they try to learn in the *bag* domain, where the hypothesis space can be larger and more complex.

One idea to develop scalable MIL methods is that instead of trying to learn complex hypothesis for bag representation, we can map each bag into a single vector representation, usually in some higher dimensional space. This vector representation for each bag should try to encode as much non-redundant information as possible for the bag. Also, it should be computationally efficient for large-scale learning. Once this mapping can be done efficiently, the MIL problem effectively moves to the supervised learning paradigm because each bag is represented by a single vector and has a corresponding label. The scalability issue can be further addressed by using any scalable supervised learning algorithm. For example, linear SVMs are known to be computationally cheap and performs quite well in high dimensional space. Based on this idea a method called miFV was proposed in [Wei et al., 2014]. On the same idea, we proposed miSUP and miSUP_MN as two other scalable MIL methods [Kumar and Raj, 2016a].

3.7.1 miFV

miFV uses Fisher Vectors (FV) for encoding bags into the vector representations. FV originates from Fisher Kernels and is a state of art method for image retrieval and classification [Sánchez et al., 2013]. We show an outline of obtaining Fisher Vectors for each bag.

A Gaussian Mixture Model (GMM) is first trained using the instances of all bags. This GMM is learned over the instance space using all instances in the training bags. Let this K component GMM be represented as $\mathcal{M} = \{w_k, N(\boldsymbol{\mu}_k, \Sigma_k)\, k = 1 \textit{ to } K\}$ where w_k is the mixture weight, $\boldsymbol{\mu}_k$ is mean vector and Σ_k is the covariance matrix of k^{th} Gaussian. The Gaussians are assumed to have diagonal covariance with diagonal variance vector represented as $\boldsymbol{\sigma}_k^2$. Then for given a bag B_i with n_i instances we compute the following for each component of GMM

$$\gamma_j(k) = \frac{w_k N(\mathbf{x}_{ij}; \boldsymbol{\mu}_k, \Sigma_k)}{\sum\limits_{j=1}^{K} w_j N(\mathbf{x}_{ij}; \boldsymbol{\mu}_k, \Sigma_k)} \tag{3.17}$$

$$\alpha_{w_k}^{B_i} = \frac{1}{n_i\sqrt{w_k}} \sum_{j=1}^{n_i} (\gamma_j(k) - w_k) \tag{3.18}$$

$$\alpha_{\boldsymbol{\mu}_k}^{B_i} = \frac{1}{n_i\sqrt{w_k}} \sum_{j=1}^{n_i} \gamma_j(k) \left(\frac{\mathbf{x}_{ij} - \boldsymbol{\mu}_k}{\boldsymbol{\sigma}_k} \right) \tag{3.19}$$

$$\alpha_{\boldsymbol{\sigma}_k}^{B_i} = \frac{1}{n_i\sqrt{2w_k}} \sum_{j=1}^{n_i} \gamma_j(k) \left(\frac{(\mathbf{x}_{ij} - \boldsymbol{\mu}_k)^2}{\boldsymbol{\sigma}_k^2} - 1 \right) \tag{3.20}$$

Finally, the Fisher Vector is concatenation of $\alpha_{w_k}^{X_i}$, $\alpha_{\boldsymbol{\mu}_k}^{X_i}$, $\alpha_{\boldsymbol{\sigma}_k}^{X_i}$ for all K Gaussians. $\gamma_j(k)$ is simply the posterior probability of $\mathbf{x}_{ij}$ belongingness to k^{th} GMM component. This results in a $(2D+1)K$ dimensional representation where D is the dimension of instance space. An improved fisher vector (IFV) obtained by sign - square rooting and L_2 normalization of the original Fisher vector can also be used. Once fisher vector has been used one can potentially use any supervised learning method to train audio event detectors. In particular, Fisher vectors work remarkably well with linear SVMs. Its clear from Equations 3.20, 3.19 and 3.18 that once the GMM $\mathcal{M}$ has been obtained, the computation of Fisher vector is cheap and can be done efficiently. Combined with linear SVMs, the overall method is very efficient and scalable.

3.7.2 miSUP

Based on the same central idea of mapping bags to single vector representation, we propose an alternative scalable MIL method by describing a different method to encode bags. Similar to miFV we start with the same GMM $\mathcal{M}$ trained on the instances of bags. The single vector representation for a bag is then obtained by maximum a posteriori (MAP) adaptation of instances in the bag to the GMM $\mathcal{M}$. This essentially implies that the parameters of $\mathcal{M}$ are updated to effectively represent the instances in the bag. The encoding of a bag X_i is done by following steps.

For each component of the GMM we compute the posterior probabilities $\gamma_j(k)$ as in Eq 3.17. Then, the mean and variance updates are computed as

$$\beta_{\boldsymbol{\mu}_k}^{B_i} = \frac{\sum_{j=1}^{m_i} \gamma_j(k)\ \mathbf{x}_{ij} + r\boldsymbol{\mu}_k}{\sum_{j=1}^{m_i} \gamma_j(k) + r} \tag{3.21}$$

$$\beta_{\boldsymbol{\sigma}_k}^{B_i} = \frac{\sum_{j=1}^{m_i} \gamma_j(k)\ \mathbf{x}_{ij}^2 + r(\boldsymbol{\mu}_k^2 + \boldsymbol{\sigma}_k^2)}{\sum_{j=1}^{m_i} \gamma_j(k) + r} - (\beta_{\boldsymbol{\mu}_k}^{X_i})^2 \tag{3.22}$$

These updates can be derived using the general MAP estimation equations [Bimbot and et al., 2004][Gauvain and Lee, 1994] . The factor r controls the amount by which the parameters ($\boldsymbol{\mu}_k$, $\boldsymbol{\sigma}_k$) from $\mathcal{M}$ affect the new estimates $\beta_{\boldsymbol{\mu}_k}^{X_i}$ and $\beta_{\boldsymbol{\sigma}_k}^{X_i}$. Although there is a known update form for mixture weights, it does not directly come from MAP estimation and we do not employ mixture weight updates in our work. In fact weight terms are in general not used in practive for Fisher Vectors as well [Vedaldi and Fulkerson, 2008]. The Fisher Vectors are thus $2KD$ dimensional. We concatenate $\beta_{\boldsymbol{\mu}_k}^{X_i}$ and $\beta_{\boldsymbol{\sigma}_k}^{X_i}$ for all GMM components to obtain the $2KD$ dimensional extended vector representation for the bag B_i. These features are sometimes referred to as Supervectors and hence the name miSUP.

We show empirically later that the concatenation of only means updates (Eq 3.21) are sufficient to obtain reasonably good and comparable results. This is an important aspect of miSUP because the dimensionality of the bag representation will be now only *KD* instead of *2KD* in miFV. This reduction by half can be significant, if *KD* is large, and can speed up the learning process in the next stage where the classifiers are trained. It can also save a significant amount of storage space if we go into large-scale audio (multimedia) content analysis problems where feature vectors of possibly millions or billions of recordings (bags) are to be stored. We will refer to the mean only Supervector based miSUP method as miSUP_MN.

3.7.3 Experiments and Results

We consider a set of 8 events, namely *Cheering*, *Children Voices*, *Clapping*, *Crowd*, *Drums*, *Engine Noise*, *Laughing*, *Scraping* from the same dataset as before. Rest of the experimental setting remains same as in previous section. In this case, we use only $\vec{F}_{64}$ and $\vec{F}_{128}$ features. For non-scalable methods we use the two forms of SVM methods, miSVM and MISVM. Linear Kernels are used in all cases. For miFV, miSUP and miSUP_MN, the GMM component size is varied as $4, 8, 16, 32, 64, 128$. Recording level recognition results are presented.

Average Precision (AP) for each event is used as the performance metric. The Mean Average Precision over all events is also shown for each case. Table 3.8 shows the AP numbers for all events and methods. Best results across different K is presented. The first noticeable aspect in Table 3.8 is that miFV and miSUP are superior compared to miSVM and MISVM. There is an

Events	miSVM	MISVM	miFV	miSUP	miSUP_MN
Cheering	0.482	0.500	0.629	0.605	0.59
Children Voices	0.130	0.142	0.264	0.193	0.193
Clapping	0.383	0.395	0.492	0.494	0.477
Crowd	0.470	0.583	0.685	0.670	0.663
Drums	0.078	0.112	0.233	0.263	0.242
Engine Noise	0.330	0.389	0.608	0.583	0.540
Laughing	0.288	0.300	0.506	0.431	0.425
Scraping	0.449	0.305	0.562	0.539	0.511
MAP	**0.328**	**0.339**	**0.497**	**0.472**	**0.455**

Table 3.8: Average Precision for Each Event

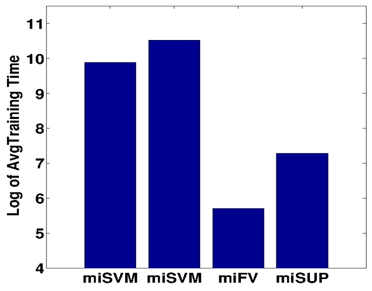


Figure 3.7: Mean Training Times (Log(sec))

K ⟶	4		8		16		32		64		128	
Events ↓	miFV	miSUP	miFV	miSUP	miFV	miSUP	miFV	miSUP	miFV	miSUP	miFV	miSUP
Cheering	0.629	0.566	0.629	0.579	0.603	0.605	0.61	0.58	0.598	0.582	0.624	0.574
Children Voices	0.227	0.182	0.199	0.158	0.174	0.193	0.176	0.185	0.152	0.142	0.172	0.125
Clapping	0.472	0.471	0.457	0.474	0.461	0.494	0.441	0.449	0.435	0.483	0.45	0.445
Crowd	0.664	0.644	0.685	0.646	0.676	0.67	0.674	0.599	0.647	0.601	0.643	0.613
Drums	0.207	0.205	0.233	0.144	0.152	0.181	0.217	0.126	0.197	0.19	0.204	0.106
Engine Noise	0.608	0.535	0.55	0.502	0.572	0.537	0.579	0.53	0.592	0.513	0.577	0.486
Laughing	0.465	0.403	0.506	0.41	0.479	0.406	0.487	0.431	0.471	0.413	0.468	0.416
Scraping	0.547	0.356	0.562	0.532	0.546	0.498	0.458	0.284	0.537	0.539	0.546	0.444
MAP	0.477	0.420	0.478	0.430	0.457	0.448	0.455	0.398	0.454	0.433	0.461	0.401

Table 3.9: AP Values For different K in GMM $\mathcal{M}$

absolute improvement of about 16−17% (around 50% in relative terms) in MAP value using these methods. For several events, miFV and miSUP improve AP by more than 75% over miSVM and MISVM. MISVM seems to be marginally better than miSVM. Also, miFV performs better than miSUP. We make a note of the fact that we used improved Fisher Vectors (IFV) [Sánchez et al., 2013] in miFV approach. It is possible that some normalization techniques for miSUP might lead to better results. The miSUP_MN method with only half the dimension of miFV and miSUP is competitive with both methods and gives comparable performance. Results for different K is shown in Table 3.9.

Analysis of Algorithms: We try to compare the average training times across all events for each MIL algorithm. For a given MIL algorithm the average training times for each event is noted and then a mean training time is obtained by averaging over all events. This comparison is shown in Figure 3.7. The y-axis shows the log of the mean training times in seconds. Clearly, miSUP and miFV are much faster (about 20 *to* 100 times) compared to miSVM and MISVM. The order of difference in time is actually higher for several individual events. This demonstrates the higher scalability of miFV and miSUP compared to miSVM and MISVM. Fisher Vectors were implemented using [Vedaldi and Fulkerson, 2008] which is possibly a very optimized implementation. With optimized implementation, miSUP should be closely comparable to miFV. For linear SVM classifiers, lower dimensional MISUP_MN representation does not lead to a substantial reduction in classifier training time and is similar to MISUP_MN. However, it is important to note that for other classifiers this gain can be significant.

3.8 Discussions and Conclusions

We introduced the idea of weak label learning for sound events. We showed that audio event detection using weakly labeled data can be formulated as a multiple instance learning problem and the recognition and detection performance numbers show that weakly supervised learning for sound events is a promising avenue. Weak label learning opens up several new frontiers for investigation but most importantly it offers us a way to scale audio event detection.

Finding correct event boundaries in audio recordings is an important task and we saw that the MIL based method is capable of doing that. However, in the current setup with a fixed window size, the detection boundaries are only a rough estimate. In our experiments, we observed that performance for temporal localization of events is lower than the recording level recognition of events. The window size itself is a hyper-parameter and may need to be adjusted for different events. Shorter segment size might be more desirable for short duration events such as Hammering, whereas longer segment sizes might be more suited for events with long-term characteristics.

We saw that MIL algorithms can be computationally expensive and may not be suited for large datasets. This forces us to think about the scalability of the MIL algorithms as scaling audio event detection is our primary goal. To address this, we proposed some scalable MIL algorithms. The scalable MIL methods encode all instances of the bag into a single vector representation. The scalable methods turned out to be not only computationally less expensive compared to non-scalable ones, but also lead to better recognition results. One concern, however, regarding these scalable methods is the temporal localization of events. The encoding method is not suited for encoding just a single instance, and hence temporal localization of events is an issue in this case. One possible solution is to train a separate event detector model using the high confidence segments (from the positive bags) obtained from the initial model trained on the weakly labeled data. The process can be iterated several times for improvement. The problem is similar to image object localization using weakly labeled data [Cinbis et al., 2017], and we can try to adopt methods from these computer vision works.

A large number of other works have followed up on our idea of weakly labeled audio event detection. Some of the works have investigated more into the multiple instance learning framework itself, [Tseng et al., 2017, Wang et al., 2018]. Others, [Su et al., 2017, Xu et al., 2017], have proposed deep neural network based approaches, which under certain assumptions again fall under the general rubric of multiple instance learning. Audio event detection using weakly labeled data has also been introduced in the annual IEEE AASP challenge on detection and classification of sound events (DCASE) [2]. A large-scale audio event dataset has been developed by Google [Gemmeke et al., 2017] to further accelerate research in this direction.

The significance of weakly supervised learning of audio event primarily lies in the fact that weakly labeled data can be obtained automatically from the web without any manual annotation effort. One obvious choice for obtaining audio data from the web is YouTube or any other video sharing websites. One can potentially use the top k retrieved results corresponding to a text query (related to the sound) as weakly labeled data. YouTube topic API can also be used for obtaining weakly labels. In other cases, it might have to be inferred from the metadata associated with the videos. In these cases, our sound related knowledge mined from the web in the previous chapter can play a crucial role. They can be used for filtering the noisy weak labels directly obtained from YouTube as well. Nevertheless, creating weak labels for a given audio recording obtained from

[2]http://dcase.community/

the web is a challenge in itself. Even more important is the fact that unless manually inspected weak labels obtained from any source or method will always contain some label noise. This poses serious learning issues and needs to properly addressed. We will look into some of these problems in subsequent chapters.

Once we register that weakly labeled audio event detection is the way to scale audio event detection, it is easy to note the importance of weakly supervised deep learning methods for audio event detection. Deep learning methods have been found to be an effective way to learn from a large amount of data. In this chapter, we presented shallow methods for AED using weakly labeled data. In the next chapter, we present deep learning methods for AED using weakly labeled data. We will present convolutional neural networks based approaches which achieve state of the art performances on large scale audio event detection with weakly labeled data. We will also see how multimedia data from the web can be directly used without manual annotation efforts.

Chapter 4

Deep Learning for Weakly Labeled Audio Event Detection

We must form our minds by reading deep rather than wide.

- Quintilian

Deep learning methods have led to remarkable improvement in performances of systems on machine learning problems. Perhaps an overused statement but an apt one given its widescale adoption in a variety of tasks in the last few years. Deep learning methods rely on large datasets, and the availability of large-scale labeled datasets have been crucial to their success. However, as pointed out earlier, lack of large-scale datasets have been a major constraint for sound event detection. The weakly labeled learning showed us a promising way to scale audio event detection by easing the labeling process. In this chapter, we will describe some deep learning methods for audio event detection using weakly labeled audio data.

4.1 Introduction

The remarkable improvement in speech recognition and computer vision tasks around the year 2012 [Hinton et al., 2012, Krizhevsky et al., 2012] using deep learning led to investigations of neural network methods for audio event detection as well [Espi et al., 2012]. In [Espi et al., 2012], a tandem connectionist approach [Hermansky et al., 2000] along the lines of speech recognition was used for AED. Multilayer perceptron (MLP) is used to generate the posterior features which are then fed to a GMM - HMM system. Other works explored different feature representations for audio and a deep neural network (DNN) as classifier [Ashraf et al., 2015b, McLoughlin et al., 2015].

However, the bulk of the works have been on using Deep Convolutional Neural Networks (CNNs) for audio event detection [Espi et al., 2015, Lidy and Schindler, 2016, Phan et al., 2016, Piczak, 2015a, Salamon and Bello, 2017, Takahashi et al., 2016, Zhang et al., 2015]. Audio signals for the purposes of classification and recognition are often characterized by some form of time-frequency representations such as spectrograms, logmel spectrograms, modulation spectrograms, constant-Q transforms (CQT), etc. CNNs allows classification systems to be built directly over these representations, instead of extracting some higher level vector representations using them.

The idea is that deep networks with successive non-linear mappings can automatically learn representations which will be suitable for recognition purposes.

[Zhang et al., 2015] extracted a fixed size (2 dimensional) features from spectrograms to use with CNNs. The CNNs themselves are fairly straightforward, some convolutional layers followed by some fully connected layers. [Espi et al., 2015] used CNN to extract features from spectrogram patches. [Piczak, 2015a] presented CNN architectures for classifying sounds using logmel spectrograms. Logmel features have been perhaps the most popular input features for CNN based audio event classification systems. [Lidy and Schindler, 2016] on the other hand used Constant-Q transforms with convolutional neural networks and showed that it could perform better than logmel spectrograms. [Salamon and Bello, 2017] investigated different data augmentation methods such as time stretching, pitch shifting, background noise addition, to improve the performance of a CNN based system.

All of the above works rely on strongly labeled audio data and can be argued to be on a small scale, considering the size and vocabulary of datasets used. Since the introduction of weak label learning [Kumar and Raj, 2016b], and release of a large scale audio event dataset [Gemmeke et al., 2017], deep learning methods for weakly labeled data have also been proposed. In this chapter, we will describe our methods for large-scale weakly labeled audio event detection. Our methods are primarily based on convolutional neural networks; however, the broad framework we propose can be applied in a variety of ways. We propose different variations of our general framework and achieve state of the art performance on weakly labeled datasets. We show that our methods can perform well on manually created weakly labeled dataset as well as on datasets obtained directly from the web, where no manual labeling was done. Later in this chapter, we investigate factors which play important roles in weak label setting. *Label density* is one such factor and is an inherent characteristic of weakly labeled data. We show how these factors can affect performance in the weakly supervised training of neural networks.

4.2 Weakly Supervised Deep Networks

Most of the discussion in this chapter will revolve around convolutional neural networks, and we will assume audio recordings are characterized by some matrix representations. These representations can be some time-frequency representations such as spectrograms or logmel spectrograms or some vector embeddings which have been obtained for every small segment of the recording. In either case, audio recordings are represented by a $m \times n$ dimensional matrix.

[Hershey et al., 2017] performed large-scale audio event detection using the web data. They used a massive amount of audio recordings from YouTube to train well known CNN architectures such as AlexNet [Krizhevsky et al., 2012], VGG [Simonyan and Zisserman, 2014] and ResNet [He et al., 2016] for sound event detection. The labeling of the recordings was done automatically using the knowledge graph and did not involve manual annotators. Moreover, the sound classes are unfiltered and had several categories which do not represent sounds, for example, *Web page*. However, it presents a simple method for learning from weakly labeled data and shows the limit of training from weakly labeled web data using a simple approach. We first describe this approach, referred to as *Strong Label Assumption Training* in this work. This approach despite being reasonably simplistic can give good performance in several cases.

Let us assume that we have a dataset of N weakly labeled recordings and a total of C sound events in the vocabulary. Throughout this chapter, unless otherwise specified, we will assume a multi-label setting. This implies more than one sound event can be present in any given audio

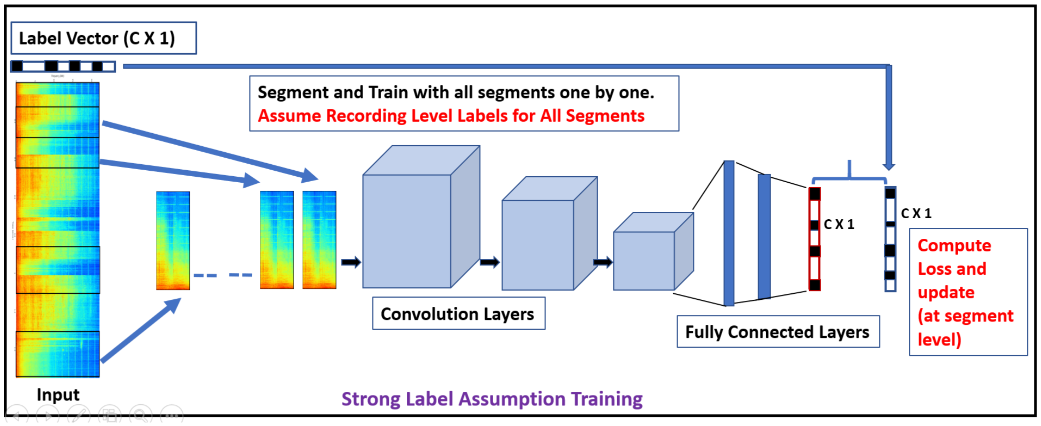

Label Vector (C X 1)
Segment and Train with all segments one by one.
Assume Recording Level Labels for All Segments
Convolution Layers
Fully Connected Layers
C X 1
C X 1
Compute Loss and update (at segment level)
Input
Strong Label Assumption Training

Figure 4.1: Schema of Strong Label Assumption Training

recording. Let the i^{th} recording and its label be represented by $(\mathcal{R}_i, Y_i)$. The label vector, Y_i, is a C dimensional vector. It is 1 at index j, if C_j^{th} sound event is present, otherwise 0.

Training any network requires loss computation and then computing gradients with respect to the loss and then back-propagating it to update the network parameters. Since the timestamps for events are not known, we cannot extract out the event specific portion from $\mathcal{R}$ to train the network in a supervised fashion. A straightforward way to handle this problem is through strong label assumption training which is presented next.

4.2.1 Strong Label Assumption Training

The main idea behind strong label assumption training (SLAT) [Hershey et al., 2017] is that we ignore the fact that the recordings are weakly labeled, and instead, assume that the event marked to be present in the recording runs throughout the whole duration recording. Hence, the assumption is that the labels are in fact strong and train the network accordingly.

Let $S_k^{\mathcal{R}_i}$, $k = 1 \; to \; K$ be small segments (say 1 second) obtained by chunking $\mathcal{R}_i$ into small pieces. The segments may or may not overlap, though most of the analysis done in this work employs overlapping segments. f represents the network function such that $f(X)$ represents the output of the network $(\mathcal{N}_{slat})$ for an input X and $\mathcal{L}$ is the loss function employed to compute the divergence between the network output and the desired output. Let us assume that we have designed a network, which takes as input fixed duration segments $S_k^{\mathcal{R}_i}$, and produces output for each segment.

Under the strong label assumption, the desired output for any segment $S_k^{\mathcal{R}_i}$ is Y_i. Then, the loss for the whole recording $\mathcal{R}_i$ is computed as the total loss over all segments of $\mathcal{R}_i$. Eq 4.1 shows the total loss for $\mathcal{R}_i$.

$$Loss(\mathcal{R}_i) = \sum_{i=1}^{K} \mathcal{L}(f(S_k^{\mathcal{R}_i}), Y_i) \tag{4.1}$$

Often, the duration of the recordings can vary. However, this is not a concern here as the segments are of fixed size and the network $\mathcal{N}_{slat}$, can be any convolutional neural network architecture. [Hershey et al., 2017] used well established CNN architectures as AlexNet [Krizhevsky et al., 2012], VGG [Simonyan and Zisserman, 2014] and ResNet [He et al., 2016]; these network architectures have been shown to perform well on computer vision tasks. Hence, the overall process is fairly simple, $\mathcal{N}_{slat}$ takes in fixed sized segments as inputs, and then the loss function is computed using the network output and the label Y_i of the recording. Figure 4.1 shows the general schema of the strong label assumption training. The network consists of some convolutional layers followed by some fully connected layers, the last layer FC3 being the output layer.

Strong label assumption training offers a simple way to train audio event detection models using weakly labeled data. It is also flexible in terms of network architecture choice, $\mathcal{N}_{slat}$ can be any CNN architecture.

Strong label assumption training despite being simple is not an efficient approach for weak label learning and can result in a significant amount of label noise while training. An audio event, say *door bell ringing*, may be present for only a few seconds in a recording which may be several minutes long, a fact that is ignored in assuming that the label is strong. Moreover, segment-wise training of CNN is cumbersome and computationally inefficient. Segmenting recordings is a preprocessing step in itself, and often one must experiment with different segment sizes, which would require repeated preprocessing of the data. Also, a significant portion of computational operations done by the network is common over different segments, and in segment-wise training these operations are repeated. We propose CNN based approaches which can address these issues [Kumar and Raj, 2017b, 2018b, Kumar et al., 2018].

4.2.2 Weakly Labeled Training

The SLAT method is based on looking at the recordings at the segment level, similar to multiple instance learning. Here also, the main idea was to look at the recordings at the segment level and then design the framework around that. This approach can be argued to be the general idea behind any weakly supervised method. Each input is considered as a collection of data points rather than a single data point and the collection as a whole has a label using which the algorithm is expected to learn. For example, we saw that MISVM does this by redefining the margin with respect to the instance which gives the maximal output. Our weakly supervised training is also a bottom to top approach where the network is designed to compute loss at recording level using segment level prediction.

Similar to the previous case, let us assume that $f(S_k^{\mathcal{R}_i})$, $k = 1, 2, ...K$ be the network outputs for segments of the recordings. $f(S_k^{\mathcal{R}_i})$ is a C dimensional vector. The main idea behind the weak label training is to map these segment level outputs to the recording level, that is one vector which represents the class wise output for the whole recording. Once we have the recording level outputs for all classes, we can compute the loss function with the recording label Y_i. This is formulated in Eq 4.2.

$$Loss(\mathcal{R}_i) = \mathcal{L}(g(f(S_1^{\mathcal{R}_i}),\ f(S_2^{\mathcal{R}_i}),\ ....,\ f(S_K^{\mathcal{R}_i}))), Y_i) \tag{4.2}$$

In Eq 4.2, the function $g()$, maps the segment level predictions $(f(S_i^{\mathcal{R}}))$ to the recording level prediction. The loss is then computed using this recording level prediction with respect to the recording level label, Y_i. The mapping function $g()$ can in principle be any function which looks at predictions on segments and then uses that knowledge to produce output for the whole recording. This formulation is essentially capturing the simple intuition that *weak labels*, that is the labels for the full recordings, essentially comes from presence or absence of events at lower levels, in this case at the segment level. The role of the mapping function is to look at these segment level outputs and obtain the recording level output using them. The general schema for

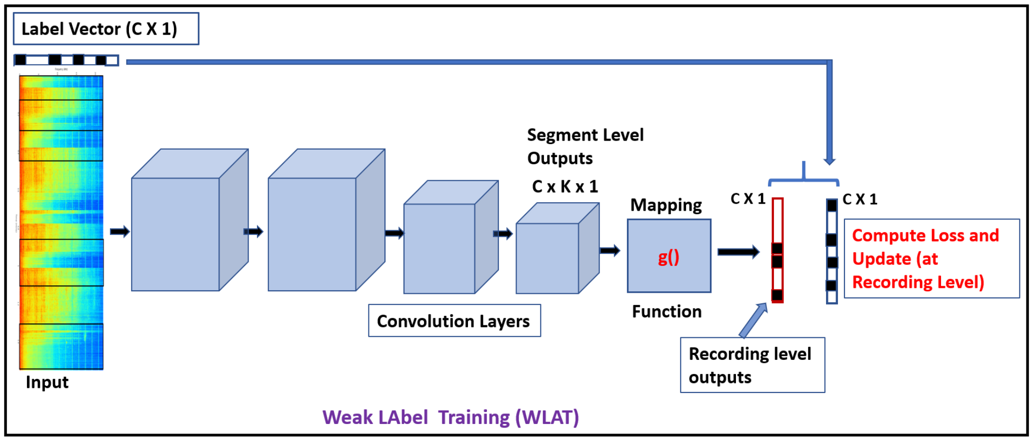


Figure 4.2: Schema of Weak Label Training

Weakly LAbeled Training (WLAT, read as "dub-lat") is shown in Figure 4.2.

Note that in Figure 4.2, the whole audio recording is input to the network and we obtain segment level output in one forward pass. We will discuss this characteristic of the network architecture later on. For now, let us assume that the network automatically produces segment level outputs which are then mapped by the function $g()$ to obtain recording level predictions. K denotes the number of segments obtained for a given input.

The function $g()$ can be any function which produces a C dimensional output vector using the outputs from the previous layer. The function $g()$ can be very complex or something simple. Here, we lay down some simple forms $g()$ can take.

Linear Functions: In this case the mapping function $g()$ is a linear function, $g(\vec{w}_j, \vec{f}_j) = \vec{w}_j^T \vec{f}_j$. Note, that we overloading the representation and the vector $\vec{f}_j$ represents the segment level outputs for the j^{th} class. $\vec{w}_j$ is the weight parameter for the j^{th} class and is a K dimensional vector. We discuss four different forms for the parameter w.

- **Sparse $\vec{\mathbf{w}}_j$**: In this case, $\vec{w}_j$ is a sparse and in fact a one hot vector. This one hot vector represent the segment which gives the maximal output for the class C_j. Hence, $\vec{w}_j$ is given by

$$\vec{w}_j^k = \begin{cases} 1, & \text{if } k = argmax_{l=1\,to\,K} \vec{f}_j^l \\ 0, & otherwise \end{cases} \tag{4.3}$$

  The mapping function g, here, in simple terms considers the maximal output across all segments as the recording level output for each class. It is inspired from the multiple instance learning neural network we presented in the previous chapter.

- **Dense Fixed $\vec{\mathbf{w}}_j$**: The one hot weighing vector above is an extreme case where only one segment for each class contributes to the final output. This might not be the most efficient thing to do as most of the segments essentially do not contribute to the loss function and update computations. Hence, we suggest a dense weighing vector where all segments play a role in the recording level output. One simple form $g()$ can take is the averaging function. The weights $\vec{w}_j$ for each class is then given by

$$\vec{w}_j^k = \frac{1}{K} \, for \ k = 1 \ to \ K \tag{4.4}$$

The mapping function in this case is essentially taking the average of segment level predictions for all classes. Clearly, the weights here are fixed parameters which can change depending on size of input (that is K) at any given step. However, these are not learned parameters.

g() as Attention Functions: Taking an average or 'max' across all segments are ad hoc ways of setting the parameter $\vec{w}$ of the mapping function g. Though these methods are well motivated, and empirically works well, the parameter $\vec{w}_j$'s can, in fact, be learned during network training. Making the weights $\vec{w}$ learnable forces them to pay attention to the appropriate segments. The approach is similar to attention like methods which have been very useful in other deep learning works [Cho et al., 2014, Xu et al., 2015a]. We propose two simple ways to bring attention into the WLAT framework through the mapping functions $g()$.

- **Single Learnable $\vec{\mathbf{w}}$**: In this case, we consider a single weight vector to weigh segments across all classes. Hence, $\vec{w}_j = \vec{w}$ for all j. $\vec{w}$ is a learnable parameter which is updated at each backpropagation step during network training. The parameter $\vec{w}$, in this case, is not used to directly weigh the segments (the number of which can vary across different input recordings) but is used to compute another weight vector $\vec{w}^s$, which is used for actually computing the weighted recording level outputs.

  Let F be the matrix representing segment level outputs for all classes. F is a $K \times C$ dimensional matrix, where each column of F is f_j, that is the segment level outputs from the network for the input recording. Eq 4.5 shows the steps for computing recording level predictions in this case.

$$\begin{aligned} \vec{w}^s &= \sigma(F\vec{w}) \\ o_j &= f_j^T \vec{w}^s \end{aligned} \tag{4.5}$$

  $\sigma(\vec{x})$ is the *softmax* function given by $\sigma(\vec{x})_i = \frac{e^{x_i}}{\sum_{i=1}^{K} e^{x_i}} \ for \ i = 1, ..., K$. Thus the mapping function g first computes an overall weight of each segment. Then the segments for each class are weighed by this weight $\vec{w}^s$, to obtain the recording level prediction. The trainable parameter $\vec{w}$ of the function g is updated in each backpropagation step. It is expected it will allow the overall network to focus on segments which might be more important than the others.

- **Class Dependent $\vec{\mathbf{w}}$'s**: This is an extension to the previous one, we now aim to compute different $\vec{w}^s$ for different classes. The argument is that the segments which should be *attended to* for different classes is expected to be different and hence the function g should maintain C such parameter vectors, one for each class. The computation (in Eq. 4.6) remains same as before except that the process is done for each class.

$$\begin{aligned} \vec{w}_j^s &= \sigma(F\vec{w}_j) \\ o_j &= f_j^T \vec{w}_j^s \end{aligned} \tag{4.6}$$

Hence, the parameter of the function g is $C \times C$ dimensional matrix, W. The computations can then be rewritten as

$$\begin{aligned} W^s &= \tilde{\sigma}(FW) \\ O &= W^s \odot F \\ o_j &= \sum_{k=1}^{K} O_{kj} \end{aligned} \tag{4.7}$$

In Eq 4.7, $\tilde{\sigma}()$ applies $softmax$ on each class, that is normalizing in the segment dimension for each class. The matrix W^s represents the importance of each segment for each class. Hence, each column of the product FW is separately recognized. $\odot$ is element wise product operation. The final recording level prediction for each output is obtained by the weighted sum of segment level outputs, where the weights are given by the matrix W^s. The learnable parameter W of $g()$ is updated along with other network parameters.

Note that, during actual training adding a regularization term for W might help avoid overfitting and improve performance. Overall this method of learning weights can be compared to attention like frameworks which have been developed in several tasks.

Sequential Mapping: The previous mapping functions do not consider the segment level outputs as a sequential or ordered structure, they treat it as a simple collection of outputs. Given that some sounds can have long-term characteristics, a sequential mapping might be useful. To treat the segment level output as a sequential structure, we can use a recurrent neural network (RNN) as the mapping function $g()$. The RNN takes in the sequence of segment level outputs and then outputs the recording level predictions.

4.2.3 Characteristics of WLAT

In this section, we discuss some characteristics of the WLAT method. For a better understanding, we will take a concrete network architecture to discuss WLAT. Note that, the network architectures in the later sections will be different from this one and this is just for the purposes of discussion here. Figure 4.3, shows an example architecture for the proposed method.

The layer blocks, C1 to C5 consist of convolutional layers followed by a max pooling layer. The convolutional layers consist of filters of size 3×3, operated with a stride of 1. The inputs at all layers are padded, with padding size set to 1. The number of channels used in each layer is: {B1: 32, B2: 64, B3: 128, B4: 256, B5:256}. The max pooling is done over a window of size 2×2 moving with a stride of 2.

Layers F1, F2, and F3 are also convolutional layers with 512, 512 and C numbers of channels respectively. F1 has convolutional filters of size 4×4; F2 and F3 has filters of size 1×1. A stride of 1 is used without any padding on the inputs of these layers. ReLU activation is used in all convolutional layers except for F3. F3 is the segment level output layer with sigmoidal activation output. The segment level output layer is followed by the mapping function g. For the purposes of discussion here we will assume that sparse $\vec{w}$'s which is same as considering the max across all segments for each as recording level output.

We will assume here that the recordings are represented by their logmel spectrograms, and these are the inputs to the network. So a recording R is represented by, $\mathcal{R} \in R^{N \times 128}$, where N is the number of logmel frames.

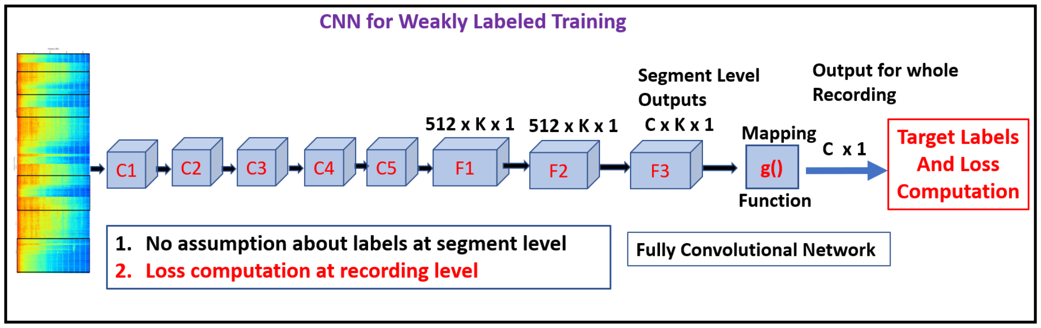

CNN for Weakly Labeled Training
C1
C2
C3
C4
C5
512 x K x 1
F1
512 x K x 1
F2
Segment Level Outputs
C x K x 1
F3
Mapping
g()
Function
Output for whole Recording
C x 1
Target Labels And Loss Computation
1. No assumption about labels at segment level
2. Loss computation at recording level
Fully Convolutional Network

Figure 4.3: An example CNN architecture for weak label training.

The *first* point to note about is that WLAT does not make strong level assumptions by design. It aims to obtain a recording level prediction through the segment level outputs and then uses that to compute the loss for the input. The *second* point worth noting is that the network is fully convolutional and can thus handle recordings of variable lengths. This ensures that the training data consisting of audio recordings of different durations is not a concern. The network produces outputs for all segments in one forward pass for any given input.

The *third* characteristic to be noted is that the network design controls the segment size, hop size and thus the number of segments produced for any input. Consider, an input recording with 1024 logmel frames, that is $\mathcal{R} \in R^{N \times 128}$. Given the network design in Figure 4.3, one can easily compute the output size after each layer. The output size at F3, which feeds into the mapping function is $C \times 29 \times 1$, that is the number of segments is 29. The network is designed to produce outputs at every 128 frames segment moving by 32 frames. The convolutional layers in C1 - C5 gives the same size output as their input, except for the channel dimension. Hence $1 \times 1024 \times 128$ input at C1 will produce output of size $32 \times 1024 \times 128$. The hop size of 32 frames is obtained by the max-pooling layers in C1 - C5, they reduce the dimension by a factor of 2. Finally, the convolutional layer F1, with filter size 4×4 also plays a role in deciding the segment and hop size. The segment size in terms of actual time duration can be computed using the sampling frequency of the audio recording and the STFT window and overlap sizes used in logmel computations. In the current example, assuming 44100 Hz sampling frequency and 1024 (23 ms) point FFT window moving by 512 (11.5 ms) points, the segment size of 128 frames comes out to be approximately 1.5 seconds. The segment hop size is approximately 0.375 seconds.

Finally, the network can do temporal localization of the events as well. The segment level predictions can be used to locate the event in the input as well. The segments, as we saw above, corresponds to specific frames of the input and hence we can project back the predictions for a segment to the input frames. These give us the probability of classes in those frames. We will show examples of such localizations in results sections.

4.2.4 Loss Function

Often several audio events are simultaneously present in an audio recording. Hence, most of our experiments will consider a multi-label task, that is more than one sound labels are possible for each recording. The sigmoid outputs of the network can be considered as class specific posteriors for any given input. Binary cross entropy function as shown in Eq 4.8 is then used to compute loss with respect to each class.

$$l(y_c, p_c) = -y_c * log(p_c) - (1 - y_c) * log(1 - p_c) \tag{4.8}$$

In Eq 4.8 y_c and $p_c = \mathcal{N}(\mathcal{X})$ are the target and the network output for c^{th} class respectively. The overall loss function is the mean of losses over all classes, Eq 4.9

$$\mathcal{L}(\mathcal{X}, y) = \frac{1}{C}\sum_{c=1}^{C} l(y_c, p_c) \tag{4.9}$$

4.3 Experiments and Results: Weakly Labeled Learning

4.3.1 Datasets

We consider two different web data sources in our experiments.
Urbansounds (US): Urbansounds [Salamon et al., 2014] dataset gives us human annotated weakly labeled data. The source of audio recordings in this dataset is Freesound website [fre]. 10 events, namely *Air Conditioner, Car Horn, Children Playing, Dog Barking, Drilling, Engine Idling, Gunshot, Jackhammer, Siren and Street Music* were manually marked in the recordings. Partial time stamps information for each recording is available in the dataset. We do not use that information, and all experiments rely only on weak labels. A total of 1302 recordings, amounting to about 27 hours is present in the dataset, with recording length varying from a few seconds to upto 10 minutes. The dataset comes pre-divided into 10 folds. We use the first 4 for training (533 recordings), next 2 (262 recordings) for validation and last 4 (502 recordings) as the testing set.

YouTube Training Set: The importance of weakly labeled learning lies in being able to obtain labeled data directly from the web without any human labeling effort and be able to train robust AED models from these data. To show this, we collect training data for the 10 sound events directly from Youtube.

Collecting videos from YouTube and automatically getting their weak labels is a challenging task in itself. We propose a simple yet effective strategy to collect weakly labeled data from YouTube. The intuition is based on observations from chapter 2, where we looked into the problem of mining sound concepts from the text. We found that the keyword “sound” is often associated with audio events when people describe it. Hence, we form the text query for searching on YouTube by adding the keyword “sound” to the event name, e.g. *children playing sound.* This leads to a considerable improvement in the retrieved queries for any given sound event. We then select the top 125 videos under 10 minutes duration retrieved by YouTube and mark them to contain that event. The duration of the recordings varies from 0.6 seconds to 10 minutes, with an average duration of 2.1 minutes. The total audio data collected is around 48 hours. Clearly, this data contains label noise, both false positives, and false negatives. We will refer to this data as *webly labeled*, as the data has not been manually labeled. The weak labels are automatically labeled.

Audioset Test Set: US dataset is a relatively clean dataset. Often it is required to recognize events in low-quality consumer generated data, which are more like the noisy web data we use for training here. To test our approach on data of such form and nature, we used the “Eval” set from Audioset dataset [Gemmeke et al., 2017]. The source of Audioset is also YouTube. 9 events out of the 10 events are present in the Audioset (except *street music*), and we will present results only for 9 events. A total of 761 test recordings exist with durations of 10 seconds for most cases.

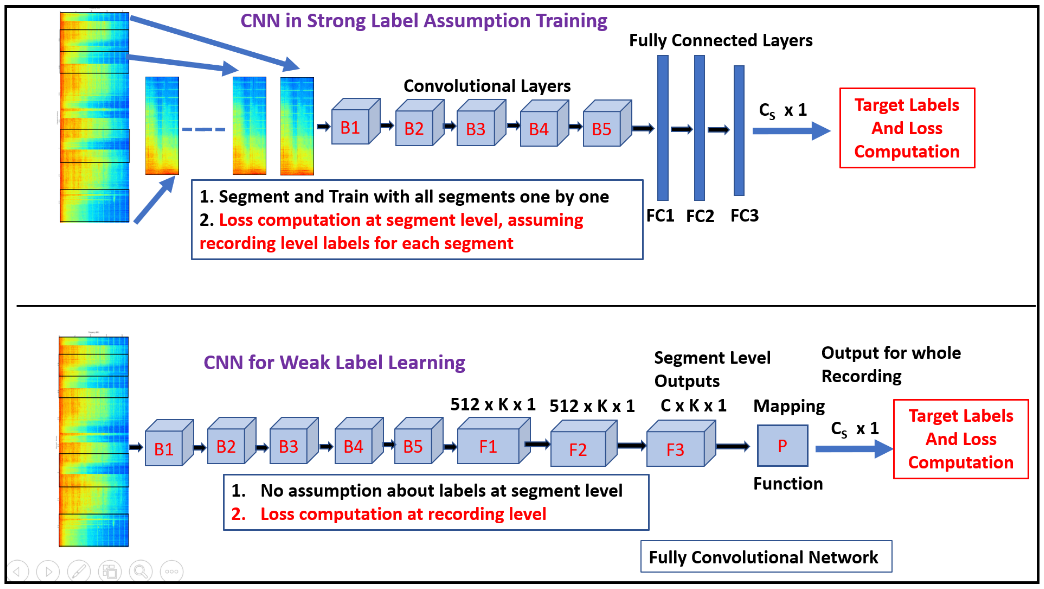

CNN in Strong Label Assumption Training
Fully Connected Layers
Convolutional Layers
B1
B2
B3
B4
B5
C_S x 1
Target Labels And Loss Computation
1. Segment and Train with all segments one by one
2. Loss computation at segment level, assuming recording level labels for each segment
FC1
FC2
FC3
CNN for Weak Label Learning
Segment Level Outputs
Output for whole Recording
512 x K x 1
512 x K x 1
C x K x 1
Mapping
F1
F2
F3
P
Function
C_S x 1
Target Labels And Loss Computation
1. No assumption about labels at segment level
2. Loss computation at recording level
Fully Convolutional Network

Figure 4.4: Top:SLAT and Bottom:WLAT Architectures used in Experiments (4.3)

4.3.2 Multi-Scale Acoustic Features

We employ logmel spectra as acoustic features for training the network. We extract logmel features at multiple scales for training the network. The different scales, here, refer to different FFT windows sizes. Different FFT sizes lead to features at different frequency resolution. The sampling rate for all recordings is 44100 Hz. FFT window sizes used are 23ms (1024), 46ms (2048), and 92ms (4096). The hop size is fixed to 11.5ms (512) for all four window sizes, and 128 mel-bands are used to extract mel spectra.

During training, the network is simply trained on all feature variants. Essentially, the multi-scale feature training serves as a data augmentation method. It exposes the network to different frequency representations which lead to better learning. During test stage, the logmel spectra for the test recordings at each FFT scale are forwarded through the network, and then the average of output scores across all FFT sizes is considered as the final output. Note that, due to mel-spectra feature extraction at three different FFT scales, the total experimental data is 3 times in all cases, that is up to 144 hours of training data in case of the YouTube training set.

4.3.3 Network Architectures

WLAT: The weak label training architecture used in experiments is shown in the bottom panel of Figure 4.4. The layer blocks, B1 to B4 consist of *two* convolutional layers followed by a max pooling layer. The convolutional layers consist of filters of size 3×3, and convolution operations are done with a stride of 1. The padding size is set to 1 for inputs at all layers. The number of channels used in each layer within each block is as follows: {B1: 32, B2: 64, B3: 128, B4: 256}. Note, both layers in each block have these many channels. The max pooling is done over a window of size 2×2 moving with a stride of 2.

The B5 block is again a convolutional block except that it contains only one convolutional layer, with 256 channel outputs, which is again followed by the max pooling layer. Layers F1, F2, and F3 are also convolutional layers with 512, 512 and C numbers of channels respectively.

FFT scales used in training	MAP		MAUC	
	$\mathcal{N}_{slat}$	$\mathcal{N}_W$	$\mathcal{N}_{slat}$	$\mathcal{N}_W$
1024	0.641	0.715	0.899	0.930
1024, 2048	0.662	0.734	0.905	0.927
1024, 2048, 4096	0.697	0.750	0.905	0.935

Table 4.1: Performance of $\mathcal{N}_W$ and $\mathcal{N}_{slat}$ and effect of multi-scale training

F1 has convolutional filters of size 4×4; F2 and F3 has filters of size 1×1. A stride of 1 is used without any padding on the inputs of these layers. ReLU activation is used in all convolutional layers except for F3. F3 is the segment level output layer with sigmoidal activation output. The segment level output layer is followed by the mapping function g.

We use linear mapping functions in experiments here, (a) Sparse $\vec{w}_j$'s and (b) Dense Fixed $\vec{w}_j$'s. The first one essentially takes maximum across all segments for each class whereas the second takes an average of all segments for each class. In the architecture here, these can be implemented through pooling layers (max or mean), where the pooling operator pools over the segments. That is the $C \times K \times 1$ output after F3 is pooled to produce $C \times 1$ output which is then used to compute the loss. This network will be referred to as $\mathcal{N}_W$.

SLAT: We compare WLAT with SLAT [Hershey et al., 2017]. $\mathcal{N}_{slat}$ is trained on recording segments where each segment is assigned the recording level label. The fully connected layers FC1, FC2 and FC3, has 512, 512 and C numbers of nodes respectively. Hence, the network configuration matches $\mathcal{N}_W$ to make the comparison fair. During the prediction stage, the recording level score for each class is obtained by applying $max()$ or $avg()$ over segment scores. Finally, averaging across all 3 logmel spectra features, similar to the weak label case is done.

4.3.4 Metrics and Experimental Setup

Similar to [Hershey et al., 2017], we will use Area under ROC curves (AUC) and Average Precision (AP) as performance metrics (Section 1.6). We will report Mean AUC (MAUC) and Mean AP (MAP) over all classes as the overall metrics for comparison.

Pytorch[1] is for implementation of our neural networks. The network is trained with Adam optimizer [Kingma and Ba, 2014]. The validation set is used for tuning hyperparameters such as learning rate.

4.3.5 Urbansounds Results

Table 4.1 compares $\mathcal{N}_W$ and $\mathcal{N}_{slat}$. Table 4.1 shows that the proposed weak label learning network outperforms the strong label assumption training by a considerable margin. For networks trained on just 1024 point FFT scale features, $\mathcal{N}_W$ outperforms $\mathcal{N}_{slat}$ by 11.5% (relative) in terms of MAP and 3.45% (relative) in terms of MAUC. Augmenting the data by adding features at different scales is helpful for both $\mathcal{N}_W$ and $\mathcal{N}_{slat}$. For $\mathcal{N}_W$, the MAP goes up by 2.65% by adding 2048 point FFT features and by 5.6% when trained on features at all three scales.

The results presented in Table 4.1 for $\mathcal{N}_W$ uses $avg()$ as mapping function g. Table 4.2 compares the results for $\mathcal{N}_W$ using $max()$ and $avg()$ as mapping function. All 3 scales of features are used. We observe that $avg()$ mapping outperforms the $max()$ mapping. The sparse weighing

[1]https://pytorch.org/

MAP		MAUC	
$g = max()$	$g = avg()$	$g = max()$	$g = avg()$
0.700	0.750	0.904	0.935

Table 4.2: Performance of $\mathcal{N}_W$ for different mapping functions

Event Name	AP		AUC	
	$\mathcal{N}_{slat}$	$\mathcal{N}_W$	$\mathcal{N}_{slat}$	$\mathcal{N}_W$
Air Conditioner	0.507	0.477	0.807	0.817
Car Horn	0.693	0.834	0.884	0.957
Children Playing	0.774	0.879	0.951	0.978
Dog Bark	0.859	0.918	0.911	0.944
Drilling	0.669	0.622	0.931	0.922
Engine Idling	0.444	0.540	0.795	0.871
Gunshot	0.832	0.929	0.983	0.990
Jackhammer	0.685	0.703	0.940	0.939
Siren	0.703	0.694	0.902	0.954
Street Music	0.800	0.907	0.949	0.978
Mean	**0.697**	**0.750**	**0.905**	**0.935**

Table 4.3: AP and AUC for each event using $\mathcal{N}_{slat}$ and $\mathcal{N}_W$

MAP		MAUC	
128 (1.5 sec)	64 (0.75 sec)	128 (1.5 sec)	64 (0.75 sec)
0.750	0.712	0.935	0.922

Table 4.4: Performance of $\mathcal{N}_W$ designed for different mapping functions

through *max*() can be thought of as picking one segment for each class which contributes to the loss function computation and updates. The *avg*() function on the other hand, allows all segments to contribute to the recording level outputs and hence loss and updates.

Table 4.3 shows event wise results for $\mathcal{N}_W$ and $\mathcal{N}_{slat}$. For most events $\mathcal{N}_W$ outperforms $\mathcal{N}_{slat}$. The difference is considerable for several events, up to $20-21\%$ for events such as *Car Horn* and *Engine Idling*. Interestingly, there are a couple of events, *Air Conditioner* and *Engine Idling* for which there is a small drop in performance and the network with strong label assumption training does better. Overall, however, training the network with the proposed method outperforms $\mathcal{N}_{slat}$ by around 7.6%.

The segment size at which the outputs are predicted is another factor which can play a role in the performance. In the previous tables, the network was designed for segment size 128 frames $\approx$ 1.5 *seconds*. Table 4.4, compares the results for $\mathcal{N}_W$ designed for segment size of 128 frames and of 64 frames $\approx$ 0.75 *seconds*. The network with segment size 64 frames is designed by changing the filter size in layer F1 to 2×4. Overall, the 128 frames segment size performs better than 64 frames segment size by about 5.3% in terms of the MAP. However, the role of segment size can be event specific. *Air Conditioner*, which has long-term more or less stationary characteristics, is better with longer segment size. 1.5 s segment size is better than 0.75 s segment size by more than 35% for *Air Conditioner*. At the same time, for events such as *Engine Idling*

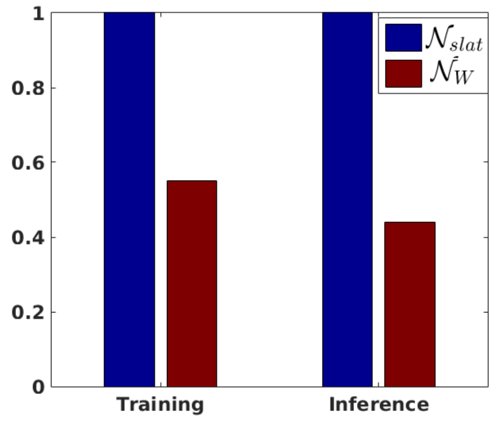


Figure 4.5: Comparison of Computational time for $\mathcal{N}_W$ and $\mathcal{N}_{slat}$
$\mathcal{N}_W$ is almost twice faster compared to $\mathcal{N}_{slat}$ during both training and inference.

Event Name	AP		AUC	
	$\mathcal{N}_{slat}$	$\mathcal{N}_W$	$\mathcal{N}_{slat}$	$\mathcal{N}_W$
Air Conditioner	0.106	0.207	0.610	0.720
Car Horn	0.269	0.440	0.773	0.854
Children Playing	0.347	0.317	0.828	0.893
Dog Bark	0.704	0.800	0.944	0.977
Drilling	0.151	0.296	0.736	0.801
Engine Idling	0.371	0.424	0.773	0.796
Gunshot	0.589	0.755	0.795	0.927
Jackhammer	0.101	0.135	0.510	0.668
Siren	0.753	0.730	0.852	0.805
Mean	**0.377**	**0.456**	**0.757**	**0.827**

Table 4.5: AP and AUC for each event using $\mathcal{N}_{slat}$ and $\mathcal{N}_W$ with YouTube Training

and *Siren* which have shorter salient characteristics, 0.75 s segment size is better than 1.5 s segment size by around 18% and 9% respectively.

Computational Expense: Figure 4.5 shows the average training and inference time comparison for the two methods. The training time is normalized for comparison. $\mathcal{N}_W$ is over 45% faster than $\mathcal{N}_{slat}$ during training and more than 55% during inference. It shows that the proposed $\mathcal{N}_W$ is not only better and more convenient (no segmentation or other such preprocessing) but is also computationally much better.

4.3.6 YouTube Results

Table 4.5 shows the performance of the networks when trained on the webly labeled YouTube recordings. The manually labeled Audioset data serves as the test set. The results correspond to training and testing with features at all three scales. Once again we see that $\mathcal{N}_W$ outperforms $\mathcal{N}_{slat}$. For several events such as *Air Conditioner*, *Car Horn*, *Drilling* improvements in the range of 63% to 96% in relative terms for AP can be seen. The MAP over all events improves by around 21%. This gain is considerably higher than that observed in US dataset. This shows that for

Event Name	AP		AUC	
	$\mathcal{N}_W$ (US)	$\mathcal{N}_W$ (YT)	$\mathcal{N}_W$ (US)	$\mathcal{N}_W$ (YT)
Air Conditioner	0.155	0.207	0.713	0.720
Car Horn	0.280	0.440	0.816	0.854
Children Playing	0.631	0.317	0.937	0.893
Dog Bark	0.610	0.800	0.928	0.977
Drilling	0.215	0.296	0.777	0.801
Engine Idling	0.373	0.424	0.738	0.796
Gunshot	0.699	0.755	0.888	0.927
Jackhammer	0.113	0.135	0.586	0.668
Siren	0.804	0.730	0.858	0.805
Mean	**0.418**	**0.456**	**0.805**	**0.827**

Table 4.6: Comparing $\mathcal{N}_W$ trained on Urbansound and on *YouTube* training set.

the webly labeled data weak label training is crucial and strong label assumptions should not be made.

Note that, the overall performance numbers are much lower compared to results on the Urbansounds dataset, where the recordings are relatively cleaner, and the presence of noise and other events are relatively low. The "label noise" is absent as the labels are manually obtained. The *YouTube* training and test set, on the other hand, are much noisier and contains other audio events which makes them more challenging as far as recognition is concerned. The training set most likely contains label noise as no manual checking was performed.

We compare the performance of the model trained on Urbansounds dataset to that of the model trained on *YouTube* on *Audioset* test set. Table 4.6 shows this comparison. We notice that the *YouTube* training set which is the only *webly* labeled, without any manual labeling effort outperforms the network trained on Urbansounds dataset. A relative improvement of around 9% in terms of MAP and 2.5% in terms of MAUC is observed. For several events the difference is much higher, for example for *Car Horn* and *Dog Bark* the improvement is by more than 57% and 31% respectively. However, there are events such as *Children Playing* and *Siren*, where the network trained on Urbansounds set is better compared to that trained on *YouTube*. So we see that recognizing events in unstructured recordings such as those on YouTube, it is important to train on recordings of such nature. However, training from YouTube data without manual labeling raises additional challenges. We will investigate those later in this chapter.

4.3.7 Temporal Localization

Even though we learn from weakly labeled data, where temporal information is not available during training, it is often essential to locate events during the inference stage. The previous results show recognition at the recording level. Computing objective measures for temporal localization requires ground truth locations of events in the recording. Neither Urbansounds test set nor Audioset test accurately provides ground truth locations of events. Urbansounds provides only partial ground truth locations, again not allowing us to compute any objective measures.

The segment level predictions at layer F3 in $\mathcal{N}_W$ can be easily mapped back to the frame level predictions since we can map back each segment level output to the receptive fields in input recordings. Figure 4.6 shows a couple of examples of localization as given by our method. The

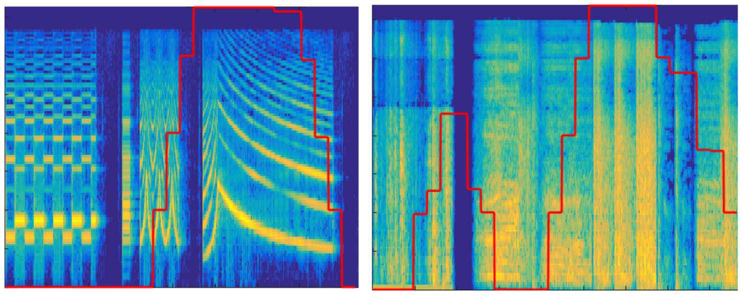


Figure 4.6: Temporal Localization Examples. Left: An Example of Siren Sound, Right: An Example of Gunshot Sound.

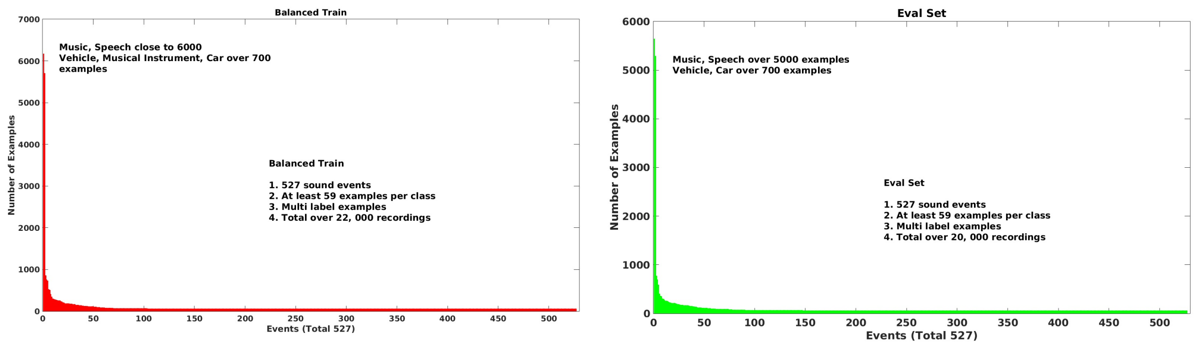


Figure 4.7: Number of examples for different sound events in training (Red) and test (Green) sets.

red line shows the output activation corresponding to the sound event as it changes with time. Note that as the event starts the activation goes up for the duration of the event. This variation of activation thus shows that the method can locate events in the recordings.

4.4 Experiments and Results: Large Vocabulary Weakly Labeled Learning

We now describe experiments and results on a large vocabulary weakly labeled dataset. We use Audioset [Gemmeke et al., 2013] , which consists of weakly labeled examples for 527 sound events. To the best of our knowledge, this is currently the largest number of sound events in any publicly available dataset. The dataset consists of weakly labeled recordings from YouTube. The labeling was done manually, and the maximum duration of the recording is 10 seconds. The dataset consists of a total of over 2 million recordings and comes pre-divided into training and evaluation sets. We will use the *Balanced Training set* of Audioset[2] for training, *Eval set* for evaluation and a subset of *Unbalanced training set* as validation set. Henceforth, we will refer to these sets as the training, evaluation (or test) and validation sets.

The training set consists of a total of around 22,000 audio recordings. Some of the videos were not available for download at the time we downloaded them. Most of the recordings are of 10 seconds duration, although a few are less than 10 seconds as well. The total training data is approximately 60 hours of audio data. Test set consists of a total of around 20,000 recordings. The training and test sets consist of at least 59 examples per event. Audioset is a multi-label

[2]https://research.google.com/audioset/

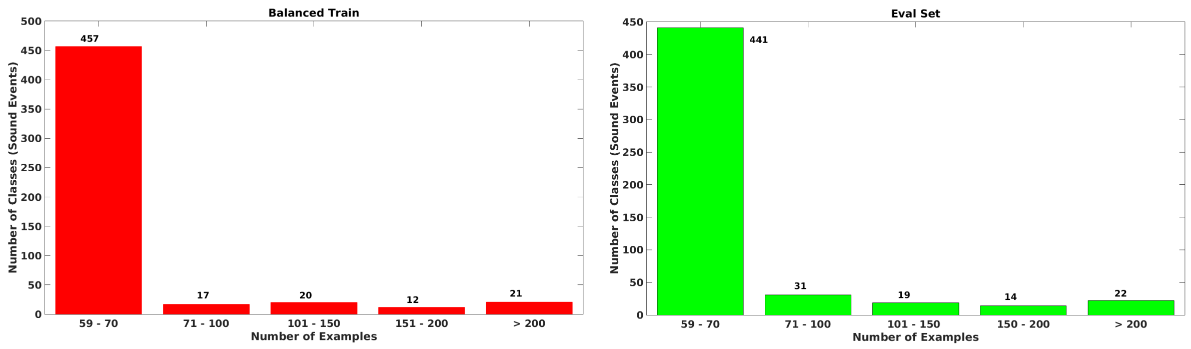


Figure 4.8: Number of Events vs Number of Examples (Distribution of examples and events) (Red) and eval (Green) sets

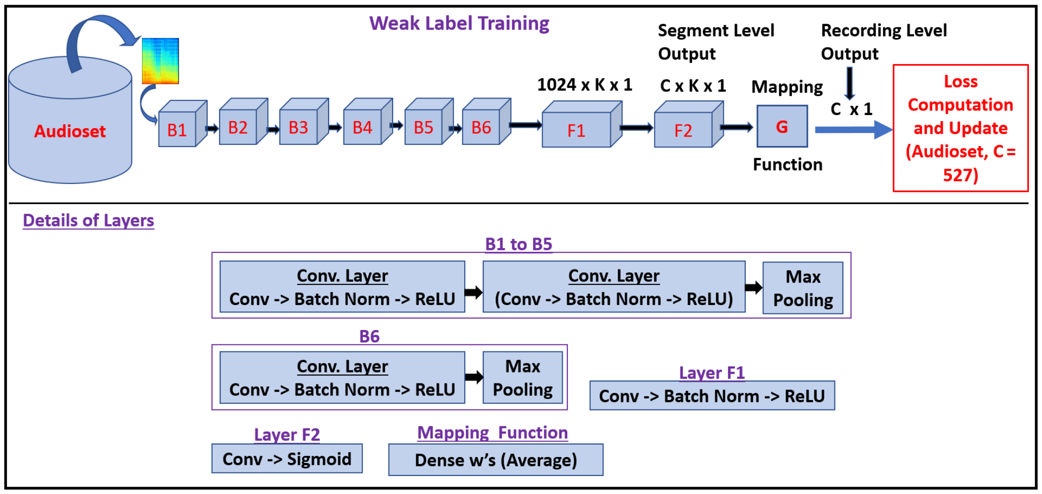


Figure 4.9: Network Architecture for Large Vocabulary Experiments (4.4.1)

dataset and on an average 2.7 labels per recording are present. Figure 4.7 and 4.8 shows the number of examples for each event and the distribution of examples with events. From Fig. 4.8, we can see that for over 50 sound events the number of recordings available for training are more than 100.

4.4.1 Experiments with Logmel Acoustic Features

All audio recordings are sampled at 44.1 KHz sampling frequency. 128 mel bands are used. A window size of 23 ms (1024 point FFT) and overlap of 11.5 ms (512 point hop size) is used for obtaining mel features. Hence, the input to the networks are recordings, R, represented by their logmel spectrograms ($\mathcal{R} \in R^{N \times 128}$). All experiments are done in Pytorch. Adam optimization is again used for training networks. The validation set is used for tuning parameters and selecting the best model. One again average precision (AP) and area under ROC curves (AUC) will be used as performance metrics. Mean average precision (MAP) and Mean area under ROC curves (MAUC) over all classes will be used as overall evaluation metrics.

MAUC		MAP	
$\mathcal{N}_{\mathbf{A}}^{\mathbf{slat}}$	$\mathcal{N}_{\mathbf{A}}^{\mathbf{wlat}}$	$\mathcal{N}_{\mathbf{A}}^{\mathbf{slat}}$	$\mathcal{N}_{\mathbf{A}}^{\mathbf{wlat}}$
0.915	0.927 (**+1.3%**)	0.167	0.213 (**+27.5%**)

Train Time		Inference Time	
$\mathcal{N}_{\mathbf{A}}^{\mathbf{slat}}$	$\mathcal{N}_{\mathbf{A}}^{\mathbf{wlat}}$	$\mathcal{N}_{\mathbf{A}}^{\mathbf{slat}}$	$\mathcal{N}_{\mathbf{A}}^{\mathbf{wlat}}$
1.0	0.61 (-39%)	1.0	0.67 (-33%)

Table 4.7: WLAT vs SLAT on Audioset
Comparison of performances (Left) and computational times (Right) of $\mathcal{N}_{\mathbf{A}}^{\mathbf{slat}}$ and $\mathcal{N}_{\mathbf{A}}^{\mathbf{wlat}}$

Network Architecture

Figure 4.9 shows the network architecture and the details of different layers. The blocks of layers B1 to B5 consists of two convolutional layers followed by a max pooling layer. B6 consists of one convolutional layer, followed by a max pooling layer. The convolutional layers consist of batch normalization before non-linear activation function. ReLU ($max(0, x)$) [Nair and Hinton, 2010] is used in all layers from B1 to B6. 3×3 convolutional filters are used in all layers from B1 to B6. Stride and padding values are fixed to 1. The number of filters employed in different blocks are as follows, {*B1: 16, B2:32, B3:64, B4:128, B5:256, B6:512* }. Note that B1 to B6 consists of two convolutional layers followed by max pooling. Both convolutional layers in these blocks employ the same number of filters. The max pooling operation is applied over a window of size 2×2. Logmel inputs are treated as single channel inputs.

F1 is a convolutional layer with 1024 filters of size 2×2. Once again ReLU activation is used. Stride is again fixed to 1, and no padding is used. Layer F2 represents segment level output. It is a convolutional layer consisting of C filters of size 1×1. C is the number of classes in the dataset. Sigmoid non-linear activation function is used at this secondary output layer. We again use the linear mapping function g, through dense $\vec{w}$'s. This is again implemented through a pooling player, which performs mean pooling along the segment level output. Note that, this network is designed for 128 frames segment size and 64 frames segment hop size. These sizes approximately correspond to 1.5 seconds and 0.75 seconds duration respectively. We will refer to this network as $\mathcal{N}_{A}^{wlat}$.

Once again, we will compare performance with an analogous network trained through SLAT. We will refer to this network as $\mathcal{N}_{A}^{slat}$. $\mathcal{N}_{A}^{wlat}$ is similar to $\mathcal{N}_{A}^{wlat}$ except that F1 and F2 are now fully connected layers of size 1024 and CS. Training is done with fixed size segments of 128 logmel frames as inputs, segments overlap by 64 frames. The loss is computed for each input segment by using recording level labels.

Results

Table 4.7 shows Mean AUC (MAUC) and Mean AP (MAP) over all 527 classes in Audioset. An absolute improvement of 1.2 (1.3% relative) in MAUC and 4.6 (27.5% relative) in MAP is obtained using $\mathcal{N}_{A}^{wlat}$. The top right table shows relative computational times, normalized for comparison. $\mathcal{N}_{A}^{slat}$ is 33% faster on an average during inference. Hence, more suitable for real applications. During training, on an average it is 39% faster for 1 full pass over training data. Once again, it establishes that the proposed WLAT method not only outperforms but is also computationally better than SLAT.

Table 4.8 shows comparison for 10 sound events for which $\mathcal{N}_{A}^{slat}$ achieved least and highest APs. For low performance classes, on an average $\mathcal{N}_{A}^{wlat}$ doubles the AP (0.0097 to 0.0203). For easier sound classes 8.5% relative improvement is obtained using $\mathcal{N}_{A}^{wlat}$.

Lowest 10			**Highest 10**		
Event	$\mathcal{N}_{\mathbf{A}}^{\mathbf{slat}}$	$\mathcal{N}_{\mathbf{A}}^{\mathbf{wlat}}$	Event	$\mathcal{N}_{\mathbf{A}}^{\mathbf{slat}}$	$\mathcal{N}_{\mathbf{A}}^{\mathbf{wlat}}$
Scrape	0.0058	0.0092	Music	0.728	0.749
Crackle	0.0078	0.0097	Siren (Civil Defense)	0.671	0.641
Man Speaking	0.0080	0.0202	Bagpipes	0.646	0.786
Mouse	0.0092	0.0368	Speech	0.631	0.661
Buzz	0.0095	0.0077	Purr (Cats)	0.575	0.600
Squish	0.0102	0.0122	BattleCry	0.575	0.651
Gurgling	0.0111	0.0125	Heartbeat	0.559	0.569
Door	0.0115	0.0685	Harpsichord	0.544	0.630
Noise	0.0116	0.0107	Ringing (Campanology)	0.538	0.690
Zipper	0.0121	0.0161	Timpani	0.538	0.528
Mean	**0.0097**	**0.0203**	**Mean**	**0.600**	**0.651**

Table 4.8: AP comparison for 10 sound events with lowest and highest APs using baseline $\mathcal{N}_S^{slat}$.

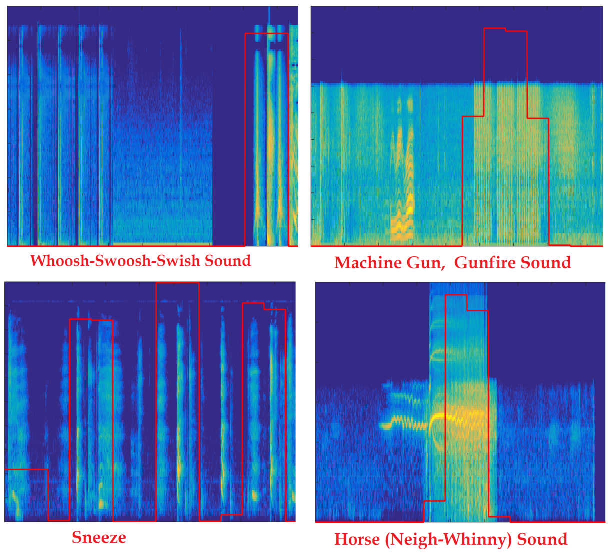


Figure 4.10: Temporal Localization Examples from Audioset.

<u>Temporal Localization</u>

Audioset does not provide temporal information of events, and hence objective measures for temporal localization cannot be computed. Figure 4.10 shows some examples of temporal localization from Audioset test set. The name of the sound event labeled in the recording is shown below each recording. The red line in each figure shows the output for that class with time. We observe that the output for the event responds to the start and end of the event. Note that segment and hop sizes can affect the localization of the event.

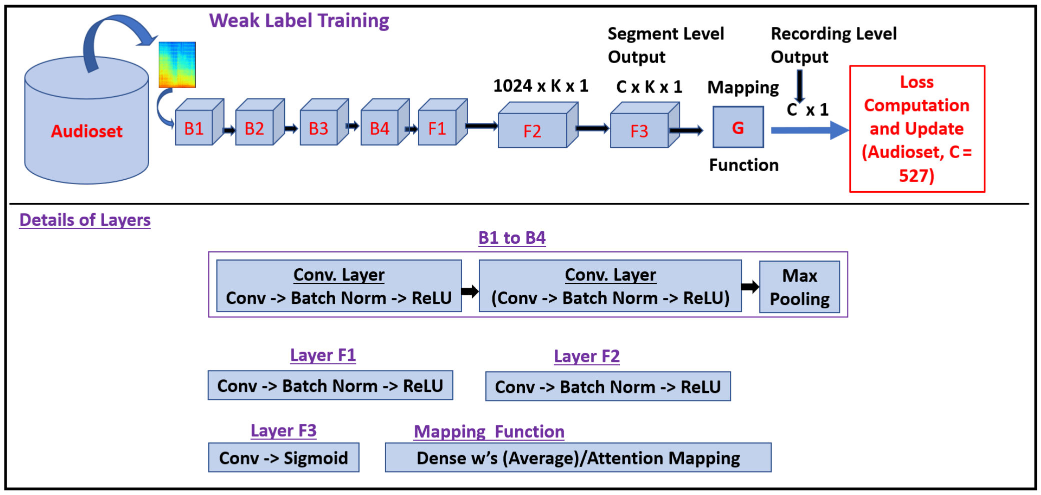


Figure 4.11: Network Architecture for Large Vocabulary Experiments on Embeddings (4.5)

4.5 Experiments and Results: Google Embeddings and Attention Like Mapping Functions

Google provides precomputed embeddings for the Audioset dataset. The embeddings were obtained by first training a VGG-like network on a massive amount of audio recordings from YouTube. The dataset used to train this VGG-like model contains over 5 million hours of audio data. The details of the embeddings can be found in [Gemmeke et al., 2017][3]. The embeddings are 128-dimensional quantized vectors. The embeddings correspond to every 1 second of audio recording. Hence, a 10 seconds audio recording will have an embedding vector for every 1 second, leading to 10×128 matrix representation. Hence, in this case, each audio recording is represented by $\mathcal{R} \in R^{N \times 128}$, where N is the number of embedding vectors for the recording.

4.5.1 Network Architecture

Figure 4.11 shows the network architecture and the details of the different layers used for training with embedding representations. The architecture is in a similar vein to the one used for logmel spectrograms. The blocks of layers B1 to B4 consists of two convolutional layers followed by a max pooling layer. The structure within these blocks is the same as the one in Figure 4.9, each convolutional layer has batch normalization followed by ReLU activation. The filter or kernel size, stride, and padding are again 3×3, 1 and 1 respectively. The number of filters employed in these blocks is as follows, {*B1: 64, B2:128, B3:256, B4:512* }. Both convolutional layers in these blocks employ the same number of filters. The max pooling operation is applied over a window of size 1×2 which is moved with a stride of 1×2.

F1 is a convolutional layer with 1024 filters of size 1×8. Once again ReLU activation is used. Stride is again fixed to 1, and no padding is used. Layer F2 is again a convolutional layer with a total 1024 filters and the size of filters are 1×1. F1 and F2 also apply batch normalization before the non-linear activations. F3 is the segment level output layer. It is a convolutional layer with C filters of size 1×1 and sigmoid non-linear activations.

[3]https://github.com/tensorflow/models/tree/master/research/audioset

Method	MAP	MAUC
ResNet-Attention [Xu et al., 2017]	0.220	0.935
ResNet-SPDA [Zhang et al., 2016]	0.219	0.936
M&mnet [Chou et al., 2018]	0.226	0.938
M&mnet (Multiscale) [Chou et al., 2018]	0.232	0.940
$\mathcal{N}_D^{wlat}$	0.228	0.927
$\mathcal{N}_S^{wlat}$	0.226	0.935
$\mathcal{N}_C^{wlat}$	0.231	0.931

Table 4.9: Comparison different methods
Comparison of performances of WLAT with different mapping functions and other works

We will consider three different mapping functions. g, with dense $\vec{w}$'s which takes an average across all segments to map segment level outputs to recording level outputs. We will also employ the attention like mapping functions. We use $g()$ with single learnable $\vec{w}$ as well as $g()$ with class dependent $\vec{w}$. We observed that regularizing these parameters helps in achieving slightly better performances. The regularization is done by adding L2 or L1 norm terms of these $\vec{w}$ parameters to the loss term. We will refer to the networks as $\mathcal{N}_D^{wlat}$, $\mathcal{N}_S^{wlat}$ and $\mathcal{N}_C^{wlat}$ for networks using mapping functions with dense fixed $\vec{w}$'s, single learnable $\vec{w}$ and class dependent learnable $\vec{w}$'s, respectively.

4.5.2 Results

Table 4.9 shows the results using the embeddings as representations for the audio recordings and the three different networks. We also compare our performance with other methods. The performance numbers for other methods have been taken from [Chou et al., 2018]. Our method performs on par or better than most of the other methods. Note that ResNet-Attention [Xu et al., 2017], ResNet-SPDA [Zhang et al., 2016], M&mnet [Chou et al., 2018], are all using some form of attention in training the network on the weakly labeled Audioset. M&mnet (Multiscale) relies on features at multiple scales. M&mnet without multi-scale features is outperformed by our method.

Overall our proposed method is able to achieve state of the art performance. Our proposed idea of first producing segment level outputs and then using a mapping function to map these segment level outputs to recording level outputs is entirely generic and can be applied with any network architecture. Other forms of mapping functions can be developed which might lead to better performance.

4.6 Closer Look at Weakly Labeled Learning

The previous sections presented deep learning methods for audio event detection using weakly labeled data. We presented a generic framework for training deep convolutional neural networks using weakly labeled audio data. We showed evaluated our performance on large-scale audio event detection problem as well as a *webly* labeled dataset, where the weak labels were obtained automatically without any manual effort. In summary, we showed that deep learning methods can be successfully applied to audio event detection using weekly labeled dataset.

However, weakly labeled learning comes with its own set of challenges. A deeper understanding and analysis of the problems and challenges arising in large-scale AED using weak labels is necessary. In this section, we take a closer look at AED using weakly labeled data. We define, study and analyze some of the factors which affect learning from weakly labeled data [Shah et al., 2018b]. We describe different ways in which "label noise" naturally manifests itself into weak label paradigm for sound events. These characteristics of weak label learning are then analyzed empirically to understand how they affect the generalization capabilities of the trained model. This analysis and understanding are not only desirable but necessary for further development of audio event detection using weakly labeled data.

The basis of this study lies in the primary advantage of weakly labeled data. The primary significance of weak label learning is that it shows a way to scale AED by easing the data labeling process. This includes mining data from the web and automatically labeling them without human labelers. However, mining multimedia data from the web for sound events often leads to noise in the training data. This aspect of weakly supervised learning has been explored in the field of computer vision, [Feng et al., 2014, Lu et al., 2017, Tang et al., 2009] to cite a few.

However, for sounds, the impact and relevance of factors such as noise in the data are yet to be analyzed, understood and explored. For sounds, weakly labeled data from the web (such as *YouTube*) contains noise primarily in two forms, *Signal noise* and *Label noise*. We use the term *signal noise* to refer to a variety of signal degrading factors. Often, consumer-generated web audios or videos are recorded under unstructured conditions. Occurrences of sound events of interests are often corrupted by either noise and/or events of which are not of interests. Collectively referring to them as signal noise, this signal noise does make the problem much more challenging. The focus here, however, is on analyzing the other type noise in weakly supervised learning of sounds, namely, the *Label noise*.

Label noise in weakly labeled audio (video) occurs naturally in different ways. We use the term "label noise" to loosely refer to multiple labeling related factors which affect learning in weakly supervised learning. Here, we consider two important factors. The first one is the *Label Density*. Label density signifies the portion of the recording (out of the total length) during which the tagged sound event is actually present in the recording. A recording with lower label density is a weakly representation of a sound event compared to another recording with high label density.

The second form of *label noise* comes from the actual wrong labeling of the audio recordings. Sound events can often be hard to interpret, and this might lead to wrong labels, even when the labeling is done manually. This becomes a bigger problem when we work on a large scale, with a large number of audio events. For example, *Audioset* which have been manually labeled; the quality of the labels has been roughly estimated to be less than 90%[4] for several events. The problem can grow by several folds when we try to mine the data directly from the web and obtain weak labels by exploiting the associated metadata. In this case, a considerable portion of the weak labels is expected to be wrong. We refer to this form of label noise as *label corruption noise*.

Clearly, *label density* noise is an inherent characteristic of weak label learning. Corrupt labels can also easily occur, and hence often the two forms of noises are simultaneously present. This is especially true for weakly labeled data obtained from the web in which no manual labeling effort was employed. In the following sections, we use the network $\mathcal{N}_A^{wlat}$ described in Section 4.4.1 for analysis. The input acoustic features are also logmel spectrograms, same as before.

[4]https://research.google.com/audioset/dataset/index.html

4.6.1 Label Density

Label density signifies how much of a given audio recording actually contains the tagged event. In weakly labeled data, we only know whether an event is present or not. We do not know what portion of the recording contains the marked event. Hence, the concept of *label density* and correspondingly *label density noise* is naturally embodied into weakly label learning. We formally define label density (LD) of a recording R with respect to an audio event E as

$$LD_R^e = \frac{Duration\,(seconds)\ \ for\ which\ E\ ispresent\ in\ R}{Length\ (seconds)\ of\ R} \tag{4.10}$$

Correspondingly, label density noise (LDN) is defined as $LDN_R^e = 1 - LD_R^e$. Label density noise is a measure of *weakness* of a recording with respect to a given event. In other words, how weak are the labels for a given weakly labeled audio recording? A one-minute long audio recording where a sound event is present for only a couple of seconds is a much weaker representation of the event, compared to a similar audio recording where the event is present for almost three-quarters of the duration of the recording.

Understanding how label density affects the learning process in a weakly supervised paradigm is important. However, designing an evaluation strategy to measure the impact of label density on the learning process is not straightforward. Any such method would require one to measure the label density of all recordings with respect to each event. This would require manually obtaining the duration for which a given event is present in the audio recording. Clearly, this cannot be done for a large scale dataset such as *Audioset*. Hence, we consider an alternative view on label density, which allows us to study its impact on weakly labeled audio event detection empirically.

The alternate view is based on the intuition that if we mine weakly labeled audio from the web, then the expected density of labels for a given audio event will be lower for long duration audio recordings. In other words, on average long duration audio recordings will have higher "label density noise" compared to shorter audio recordings.

Taking this alternate view, we design our experiment as follows. We consider the *Audioset* dataset for experiments. *Audioset* provides weak labels for YouTube audio recordings. Weak labeling was manually done on audio recordings of 10 seconds duration (a small fraction are of less than 10 seconds). These 10 seconds audio segments were actually obtained from longer YouTube videos.

Let us represent the Audioset training data by tuples of the forms $(R^i, YT^i_{id}, S^i, E^i, L^i)$. R^i is the i^{th} recording in Audioset and YT^i_{id} is the YouTube id of R^i. S^i is the start time of R^i in YT^i_{id} and E^i is the end time of R^i in YT^i_{id}. That is R^i is a portion of the video YT^i_{id} on YouTube, starting at S^i and ending at E^i. R^i's are weakly labeled (manually). L^i represents the set of audio events present in R^i. Hence, this labeling tells us that the set of events in L^i are somewhere present in YT^i_{id} between S^i and E^i. We request the readers to take a quick look at the label sets provided by Audioset[5] to understand this aspect.

R_i's are mostly 10 seconds long recordings and will have label densities likely on the higher side. For each YT^i_{id} in training set, we consider the audio from $S^i - 10$ seconds to $E^i + 10$ seconds. Clearly, the set of L^i events are present in this audio as well, and hence we obtain 30 seconds long weakly labeled audio recordings. Those YT^i_{id} where the total duration of YouTube video is less than 30 seconds, we simply consider the whole recording. Henceforth, we refer to this weakly labeled set as *Audioset-At-30*. It is expected that on an average the label density of events in Audioset-At-30 will be lower than Audioset.

[5]https://research.google.com/audioset/download.html

Table 4.10: Effect of label density on performance

Training Set	**MAP**	**MAUC**
Audioset	0.213	0.927
Audioset-At-30	0.172 (**-19.2%**)	0.904 (-2.5%)
Audioset-At-60	0.165 (**-22.5%**)	0.908 (-2.9%)

Similarly, we obtain *Audioset-At-60* by considering $S^i - 25$ seconds to $E^i + 25$ seconds for each $(YT^i_{id}, S^i, E^i, L^i)$. Once again, YT^i_{id} which are of less than 60 seconds are taken entirely. *Audioset-At-60* is expected to have even lower label density or higher label density noise, compared to Audioset-At-30. We train our weakly labeled method on all 3 sets to understand how label density affects the learning process. Since our networks are designed to handle audio recordings of variable length, variable lengths of recordings is not a problem.

Analyzing Label Density

The experimental setup for this part follows the procedure outlined above. For the training set in the Audioset, we created two additional sets, namely, *Audioset-At-30* and *Audioset-At-60*. Audioset-At-30 and Audioset-At-60 are approximately 180 and 360 hours of audio data respectively. These two additional training sets have higher label density noise compared to the original Audioset. We train $\mathcal{N}_A^{wlat}$ on original Audioset, Audioset-At-30 and Audioset-At-60. The validation and evaluation sets are kept the same as before. The original Audioset gives label density at 10 seconds segment. Hence, we are analyzing the performance of models trained on datasets for which weak labels are available at 10 seconds, 30 seconds and 60 seconds granularity.

Table 4.10 shows the effect of label density change in MAP and MAUC numbers. The MAP value decreases by a sharp **12**% in relative terms as we move from weak labels at 10 seconds long audio recordings in the Audioset to 30 seconds long audio recordings in Audioset-At-30. The drop in performance is evident in both MAP and MAUC numbers. Going from 30 to 60 seconds leads to an additional drop of **4**% in the MAP. No significant change in MAUC is observed. This shows an interesting behavior. As we increase the recording durations from 10 to 30 seconds, the model is not able to learn well from relatively reduced label density. One possible explanation might be that increasing the data leads to model getting more robust. Note that the test set in all cases are the same. Larger training data have been known to improve the robustness of deep learning models. Hence, even though the performance does go down due to reduced label density, training on a larger amount of data reduces the impact to a certain extent. A deeper analysis in the future might give better insights.

Table 4.11 shows 10 events for which maximum drop (relative) in AP is observed for Audioset-At-30 compared to the Audioset. For the Table 4.11, we considered only those whose AP is more than the MAP. Similarly, Table 4.12 shows the same for Audioset-At-60. For these ten events, on an average around 62% drop in AP is seen for Audioset-At-30 and 65% for Audioset-At-60. Note that the set of events for which maximal drop is observed for Audioset-At-30 and Audioset-At-60 are different. Interestingly, most of the events in these two lists, except for *Pink Noise, Music of Bollywood, Music Africa*, have very unique and specific characteristics and are easily identifiable by humans. One can then argue that such events are more prone to suffer from label density noise.

Table 4.11: 10 Audio Events with highest drop in AP for Audioset-At-30 (w.r.t Audioset)

Events	**Audioset - AP, (AUC)**	**Audioset-At-30 - AP, (AUC)**	**AP Drop (% Drop)**
Speech synthesizer	0.225, (0.953)	0.037, (0.882)	0.187, (83.506)
Air horn truck horn	0.290, (0.979)	0.051, (0.950)	0.239, (82.310)
Vehicle horn, honking	0.238, (0.955)	0.084, (0.923)	0.155, (64.913)
Afrobeat	0.261, (0.977)	0.104, (0.967)	0.157, (60.088)
Whip	0.274, (0.927)	0.112, (0.876)	0.162, (59.117)
Chopping (food)	0.285, (0.920)	0.117, (0.886)	0.168, (58.887)
Music of Bollywood	0.309, (0.933)	0.140, (0.933)	0.169, (54.765)
Sizzle	0.339, (0.976)	0.155, (0.958)	0.183, (54.122)
Toot	0.290, (0.963)	0.137, (0.961)	0.153, (52.813)
Pour	0.238, (0.957)	0.115, (0.940)	0.123, (51.717)
Mean	**0.275, (0.954)**	**0.105, (0.927)**	**0.170, (62.224)**

Table 4.12: 10 Audio Events with highest drop in AP for Audioset-At-60 (w.r.t Audioset)

Sound Events	**Audioset - AP, (AUC)**	**Audioset-At-60 - AP, (AUC)**	**AP Drop (% Drop)**
Chopping (food)	0.285, (0.920)	0.064, (0.871)	0.220, (77.348)
Pink noise	0.349, (0.974)	0.093, (0.942)	0.257, (73.448)
Cash register	0.205, (0.917)	0.056, (0.918)	0.149, (72.722)
Whip	0.274, (0.927)	0.078, (0.877)	0.196, (71.705)
Music of Africa	0.225, (0.966)	0.082, (0.919)	0.143, (63.512)
Keyboard (musical)	0.303, (0.955)	0.116, (0.907)	0.186, (61.524)
Afrobeat	0.261, (0.977)	0.102, (0.955)	0.159, (61.038)
Speech synthesizer	0.225, (0.953)	0.089, (0.905)	0.136, (60.401)
Electronic organ	0.205, (0.916)	0.089, (0.847)	0.116, (56.491)
Artillery fire	0.218, (0.963)	0.102, (0.939)	0.117, (53.482)
Mean	**0.255, (0.947)**	**0.087, (0.908)**	**0.168, (65.167)**

4.6.2 Corrupted Labels Noise

Corrupted labels or wrong labels is possible due to a variety of reasons. Sound events are often hard to interpret which might present difficulties in labeling recordings even when manually done. This is especially true for YouTube quality audio recordings where "signal noise" can create further difficulties in understanding and labeling the audio events. For the Audioset, the authors themselves checked a random sample of 10 recordings for each event and provided a confidence on the correctness of quality of labels [6]. Interestingly, this confidence is less than 90% for several events. This problem is hard to address though; manual labeling is the best one can do as far as labeling is concerned. Hence, we assume that labels of Audioset are not corrupted and have 0 label noise.

However, quality of labels is adversely affected to a much larger extent when we mine audio from the web and automate the weak labeling by using metadata associated with audios (videos). Both false positive, as well as the false negative assignment of labels, are possible. That is, we may mark an event to be present in the recording when it is not. Similarly, we might miss the presence of an event and mark the recording as not containing the given sound event.

We analyze this form of label noise by corrupting the labels in Audioset. We treat the labels from Audioset as perfect or at 0% corruption. We note that this is actually not true. However, Audioset has been manually labeled, and it is the best one can possibly do for a collection of over 500 sound events. Hence, we consider Audioset as having perfect labels without any noise.

[6]https://research.google.com/audioset/dataset/index.html

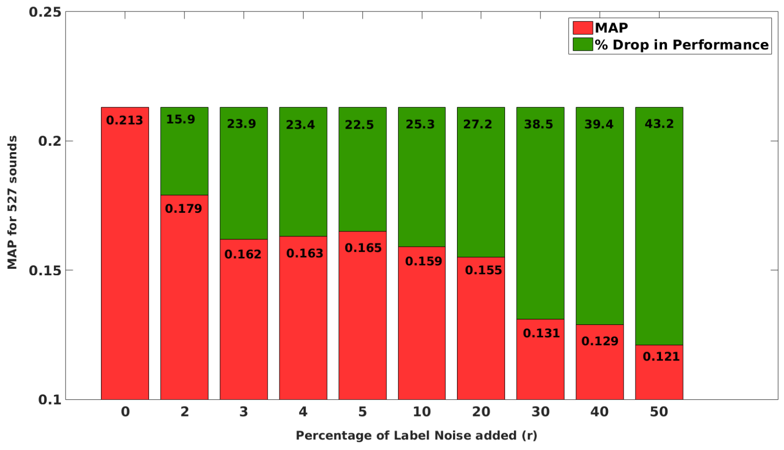


Figure 4.12: Effect of corruption of labels on performance
x-axis represents corruption level r in %. y-axis show MAP and % reduction in MAP compared to no corruption.

Table 4.13: Weakly Labeled Audio in the Wild. Comparison with Audioset

Training Set	**MAP**	**MAUC**
Audioset-40	0.419	0.913
YouTube-Wild	0.127	0.700

We then gradually increase the amount of corrupted labels by manually corrupting the assigned labels in Audioset. More specifically, for a portion of the labeled tuples $(YT^i_{id}, S^i, E^i, L^i)$, we corrupt L^i by changing the events in L^i. The corruption process is done in a stratified way, such that around r% labels for each event get corrupted. r% includes both false positive and false negative label noises. Moreover, we perform the corruption in a way such that the number of recordings marked to contain the recording remains consistent with respect to original labels. We perform analysis on different values of r.

Analysis of Labels Corruption

The analysis of label noise in terms of corrupted labels follows the procedure outlined above. We use 9 different values of r, the level of artificially induced corruptions in labels. Correspondingly, 9 different training sets are obtained. $\mathcal{N}_A^{wlat}$ is trained on each of these training sets. The validation and evaluation sets remain as before. Fig. 4.12 shows MAP values for different values of r. $r = 0$ corresponds to the original training set from Audioset.

Adding around 2% noise in labels leads to around 15.9% drop in MAP performance. A further 8% drop in performance occurs by adding an extra percentage of noise. As the noise keep increasing, the performance keeps going down. At around 30% noise, the model goes on to loose as much as 38.5% in performance. At 50% noise, the model ends up losing 43.2% in performance.

Overall, we can state that the corruption of labels is a crucial factor which can adversely affect the performance of the system. Depending on the amount of label noise a considerable drop in performance is possible. Still, a deep learning model like WLAT seems robust to a certain extent. Future works on AED using weak labels, especially those relying on YouTube data should factor in the possibility of corrupted labels and model accordingly.

Table 4.14: Best 10 and Worst 10 performing events (ordered by AP) for YouTube-Wild

Events (Best 10)	**YouTube-Wild**	**Audioset-40**	**Events (worst 10)**	**YouTube-Wild**	**Audioset-40**
Guitar	0.316, (0.748)	0.755, (0.961)	Engine	0.048, (0.488)	0.330, (0.896)
Siren	0.291, (0.731)	0.735, (0.970)	Rail transport	0.042, (0.564)	0.441, (0.929)
Animal	0.289, (0.712)	0.579, (0.882)	Violin, fiddle	0.041, (0.692)	0.446, (0.945)
Chicken-rooster	0.268, (0.824)	0.281, (0.908)	Tools	0.039, (0.540)	0.356, (0.872)
Vehicle	0.243, (0.566)	0.482, (0.828)	Bus	0.039, (0.779)	0.052, (0.818)
Emergency vehicle	0.234, (0.728)	0.613, (0.961)	Train	0.035, (0.476)	0.427, (0.925)
Laughter	0.233, (0.881)	0.612, (0.960)	Motorboat, speedboat	0.035, (0.732)	0.060, (0.845)
Drum	0.224, (0.747)	0.572, (0.948)	Truck	0.032, (0.512)	0.134, (0.878)
Drum kit	0.218, (0.855)	0.530, (0.974)	Race car, auto racing	0.030, (0.622)	0.157, (0.884)
Crowd	0.191, (0.693)	0.681, (0.979)	Motorcycle	0.026, (0.599)	0.066, (0.826)
Mean	**0.251, (0.748)**	**0.584, (0.937)**	**Mean**	**0.037, (0.600)**	**0.247, (0.882)**

4.6.3 Weakly Labeled Audio In the Wild

We now consider the situation where both of the above forms of *label noises* can occur in abundance. We directly obtain audio from YouTube for a given sound event and then train our model using these data. We collect the data using a very simple approach. The videos are retrieved by using search queries of the forms "<sound event name > sound" on YouTube. Adding the word "sound" leads to a considerable improvement in the retrieval of relevant videos on YouTube.

For each event, we consider the top 50 retrieved videos (under 4 minutes duration) and mark these to contain the event. If a video is retrieved for multiple events, they are accordingly multi-labeled. We call this training set *YouTube-wild*. In *YouTube-wild* label noises in all forms occur. Since the retrieval is not perfect, marking all top 50 videos to contain the event introduces false positives in labels, implying we are assigning a positive label to the recording even when the event is not present. False negative labels also get naturally introduced in the process, as a retrieved video V for an event, e_1, might contain another event, e_2, as well. However, unless V was retrieved for e_2 also, we do not know this, and we mark e_2 to be not present in V. Label density can again be very low for these unverified recordings.

Several of the events in *Audioset* have vague names and broad meaning. The automated labeling process described in the previous paragraph leads to mostly non-relevant results. Hence, for this part, we worked with a smaller subset of events; selecting those who have a more definite description and meaning. That is, those for which retrieval leads to somewhat meaningful and relevant set of audio recordings. The selection process also factors in the total number of examples available for the event in Audioset. We select events for which more examples are available in Audioset. We do this because we wanted to analyze how *YouTube-wild* compares with *Audioset*, which is manually labeled. It is desirable that we work with events for which a higher number of examples are available in Audioset, to better understand where *YouTube-wild* training stand in comparison to Audioset. YouTube-wild is collected and labeled without manual effort and contains long duration audio recordings, against Audioset, which is manually labeled, and weak labels are over relatively short 10 seconds recordings. Since the source of both is YouTube, signal noise is expected to be similar. The two forms of "label noise", however, are going to make the most difference.

Analyzing YouTube-Wild

We selected a total of 40 sound events and obtained training examples for them from YouTube by the process described previously. This set, called *YouTube-Wild* is used as the training set for

Table 4.15: 10 Events with highest relative drop in performance

Events	**Audioset-40 - AP (AUC)**	**YouTube-Wild**	**AP Drop (% Drop)**
Train	0.427, (0.925)	0.035, (0.476)	0.392, (91.861)
Violin fiddle	0.446, (0.945)	0.041, (0.692)	0.405, (90.727)
Rail transport	0.441, (0.929)	0.042, (0.564)	0.399, (90.386)
Singing	0.659, (0.947)	0.089, (0.529)	0.569, (86.420)
Keyboard (musical)	0.568, (0.949)	0.084, (0.735)	0.484, (85.168)
Railroad car, train wagon	0.451, (0.934)	0.071, (0.697)	0.380, (84.313)
Water	0.616, (0.935)	0.118, (0.621)	0.498, (80.817)
Bass drum	0.541, (0.973)	0.119, (0.782)	0.422, (78.070)
Cymbal	0.577, (0.977)	0.151, (0.833)	0.426, (73.911)
Pigeon, dove	0.526, (0.956)	0.140, (0.771)	0.386, (73.388)
Mean	**0.525, (0.947)**	**0.089, (0.670)**	**0.436, (83.506)**

training $\mathcal{N}_A^{wlat}$. It consists of a total of 1906 videos for 40 sound events, totaling around 60 hours of audio. The average duration of audio recordings is around 1.92 minutes, with a maximum duration of 240 seconds. In comparison, Audioset has mostly 10 seconds long audio recordings. All audio recordings are sampled to 44.1 kHz sampling rate. *YouTube-Wild* training list will be released and made available for future works in this area. We refer to this subset of Audioset with these 40 sounds as *Audioset-40*.

We train $\mathcal{N}_A^{wlat}$ (with $C = 40$) on YouTube-wild. We also train $\mathcal{N}_A^{wlat}$ ($C = 40$) on Audioset-40. The subset of Audioset validation and evaluation sets which contains only recordings belonging to only these 40 events are used as validation and evaluation set in these experiments.

Table 4.13 compares performance on Audioset-40 and YouTube-Wild. One can observe the considerable difference between the performance of these training sets. Audioset-40 is manually labeled and is also expected to have reasonably good label density as recordings are of 10 seconds duration. On the other hand, YouTube-wild is not manually labeled or even verified and is expected to have a large number of corrupted labels. Moreover, YouTube-wild consists of very long duration recordings (up to 4 minutes) and hence even those where the weak labels are correct the label density can be very low. Together these two forms of noises severely affects the learning process. Learning from web data without any manual labeling remains a major challenge. Future works on large-scale AED needs to develop algorithms which can address these challenges.

Table 4.14 shows 10 best and worst performing events for YouTube-Wild. Corresponding Audioset-40 numbers for these events are shown in the Table. Even for the 10 events where YouTube-Wild does well, compared to Audioset-40 an average drop of 55% in AP is noted. At the same time, there are a few events such as *Chicken-Rooster* among the best 10 and *Bus* among worst 10, where both sets can be argued to have similar performances.

Table 4.15 shows 10 events for which the highest percentage drop in performance is observed. Corresponding AUC values are also shown in parenthesis. As expected the average drop over these cases is very high, higher than 80%. Interestingly, 3 of the classes (*Train, Rail Transport, and Railroad Car-Wagon*) are broadly expected to fetch similar results from YouTube. For all of these three classes, the performance drop is substantial.

4.7 Summary and Conclusions

In this chapter, we described deep learning methods for audio event detection using weakly labeled data. More specifically, our methods primarily relied on convolutional neural networks and then incorporated the constraints of the weak label in the learning process. The fundamental idea is that we take a bottom to top approach, where the prediction at recording level is done through predictions at segments level. In WLAT, the network is designed to produce segment level outputs directly, and then a mapping function maps these segment level outputs to recording level outputs. The overall framework is entirely generic and can be applied through different network architectures. We discussed several mapping functions, including some which bring attention like behavior into the framework. Other forms of mapping functions can also be developed.

Our method embodies several desirable characteristics from the perspective of weakly labeled audio event detection. It can handle recordings of variable length, and the network design controls segment sizes over which segment level outputs are produced. This removes any additional pre-processing step. Our method is able to achieve state of the art performances. Compared in particular to the SLAT method, our WLAT method performance better and is also computationally less expensive resulting in improved training and inference times.

We also dived deeper into the nature of weak label learning. We specifically pointed out two factors which we are expected to run into when we try to learn from weakly labeled data. The problem of label density noise is inherent to the nature of weak labels. Corrupted labels are also expected to occur often. Given that other deep learning methods for weakly labeled dataset has been proposed [Chou et al., 2018, McFee et al., 2018, Tseng et al., 2017, Xu et al., 2017], it would be interesting to see which of these more robust to these factors label noises. It is essential that future works address these issues while learning from weakly labeled data.

Chapter 5

A Unified Framework: Combining Weakly and Strongly Labeled Data

You make different colors by combining those colors that already exist.

-Herbie Hancock

In the previous chapter, we developed deep learning methods for audio event detection using weakly labeled dataset. Towards the end of the chapter, we delved deeper into weak label learning for sounds and discussed factors which are expected to play important roles in weakly supervised learning of sounds. Specifically, *label density* and *noisy/corrupted labels* were the two main factors we investigated. In this chapter, we introduce another novel framework for training sound event detectors. We describe a unified learning framework, which uses both weakly and strongly labeled data. We call this framework (WEA)kly and Strongly Labeled learning (WEASL, pronounce as *weasel*).

5.1 Introduction

Our unified framework aims to simultaneously exploit weakly and strongly labeled data. Training with strongly labeled data can otherwise be referred to as fully supervised learning because positive and negative examples of audio events are available (extracted through available time stamps). In the weakly labeled data, labels are available only at recording level and only weakly supervised learning is possible. Our WEASL framework [Kumar and Raj, 2017a] offers a unique form of learning which can leverage labeled data in both forms.
The motivation behind WEASL can be understood through the following three points.

- The problem with strongly labeled data is that a large amount of strongly labeled data is hard to collect. Nonetheless, in a lot of cases strongly labeled data are available or strong labels can be created, though in a small amount. Weakly labeled data, on the other hand, can be obtained on a much larger scale. Given labeled data in the two forms, it is desirable to have a learning framework which can use them simultaneously. The labeled data in both forms can together help in learning robust models for audio events.
- Weakly supervised learning offers us an opportunity to use the massive amount of mul-

timedia data on the web, example from *YouTube*. Weak labels for a recording can be automatically obtained; for example, through the metadata associated with the recording. The retrieval engines of these websites often rely on the metadata and can directly give us a list of videos for any query term, further easing the weak labeling process. Irrespective of how the weak labels are obtained, such weak labels are always expected to be noisy. Consider, for example, the sound event *barking*. A query search of *barking* on YouTube also returns recordings such as *Hillary Clinton literally barks at Republicans* among the top results which obviously has nothing to do with the acoustic event barking. Thus, getting rid of label noise entirely under these circumstances is extremely difficult and close to impossible.

  Another form of noise we discussed before is label density noise. A weakly labeled set with low label density or high label density noise can again make the learning process very difficult. We believe that WEASL can address these problems to a certain extent. A small amount of data can be strongly labeled. These "pure" examples can be appropriately exploited to improve the overall training of event detectors. It can be used along with the large pool of weakly labeled recording through a unified learning framework which can harness labeled data in both forms. The effects of label noises while learning weakly labeled data can be mitigated by the strongly labeled data or in other words the presence of well-labeled data can make weakly supervised learning tolerant to label noises.

- A large portion of web audio or multimedia data are consumer generated. The recording conditions, styles, and sophistication vary significantly among such recordings, resulting in large within-category variations among different instances of an event, making the fundamental learning problem challenging. Moreover, the audio signal itself might be very noisy. Another overlapping event or noise might degrade the event of interest. Overall, learning from web data presents yet another challenge in the form of what we can refer to as "signal noise". Once again, a small amount of "pure" examples from strongly labeled data can be appropriately exploited to counter these signal noise issues.

The above points lay down the motivations behind our unified learning framework. To the best of our knowledge, this is the first such framework for sounds which leverages labeled data in both forms. Some works in the field of computer vision have considered labeled data in both forms for object detection in images. [Hoffman et al., 2015] used strong labels as an auxiliary data source and the optimization function jointly optimizes both weak label loss and the strong label loss. The weak label loss follows the MISVM formulation we had discussed before. [Li et al., 2018] proposed another object detection method using both strongly and weakly labeled images. This mixed supervised detection, as they called it, tries to learn domain invariant features from the strongly labeled data and then attempts to transfer the knowledge to weak categories. An important point to note for these works is that the strongly and weakly labeled data are for different object categories. [Xu et al., 2015c] tries to use images from the web to aid strongly labeled data in *part detection* in images.

Similar to the above works, our goal is to learn from labeled data in both forms. Our WEASL framework retains the basics of MIL to work with the weakly labeled data. Audio recordings are converted into bags by segmenting into small segments, and positive (+1) or negative (-1) labels are assigned to the bags according to the weak labels available for the recordings. So the weakly labeled in our unified framework is once again a collection of labeled bags.

We propose two approaches for the unified learning framework. The first one called *naive-WEASL* follows a naive approach to the blended learning. It adapts the strongly labeled data in the MIL framework to learn simultaneously from labeled data of both forms. The second and the more important approach is a generic formulation for WEASL, under which potentially a variety of methods can be devised. The central idea behind the second approach is that WEASL can be formulated as semi-supervised (SSL) learning with constraints. Under this formulation, one can adopt a variety of SSL methods for WEASL. We will refer to this generic approach as just *WEASL*.

5.2 Naive WEASL

The fundamental idea behind weakly labeled audio event detection was that it can be formulated as a multiple instance learning problem. The weakly labeled recordings are converted into labeled bags; by segmenting the recordings and using appropriate feature representations for the segments. The bags are labeled as positive for an event if the weak labels mark the event to be present in the recording and otherwise negative. The segments of the recordings are instances, the labels for which are not known.

In WEASL, along with the weak data in the form of labeled bags, we also have strongly labeled data. The strongly labeled recordings are essentially a collection of labeled exemplars for each event. The naive approach to learn from labeled data in both of these forms is to treat strongly labeled data as a special case of weakly labeled data. In this formulation, each labeled instance coming from the strongly labeled set is considered as a bag with one instance only. The bag label is the same as instance label. So the strongly labeled data also becomes a collection of labeled bags. Once this is done, any MIL approach can be applied.

MIL methods either incorporate the weak labels through additional constraints in the learning algorithm or the loss function is modified to handle the weak labels appropriately. The fact that for a positive bag all we know is that it contains at least one positive instance needs to be factored in somehow. The strongly labeled data in naive-WEASL are considered as labeled bags with a single instance. Hence, any MIL method will treat the strongly labeled data as weakly labeled data. Since each such bag contains only one instance, the constraints imposed by MIL methods will end up satisfying the supervised label constraints. For example, if we use miSVM as the MIL method, then the bags formed from strongly labeled will satisfy label constraints as in conventional supervised SVM. Hence, data in both strong and weak forms gets used appropriately.

5.3 Generalized WEASL

naive-WEASL is a simple way of unifying strongly and weakly labeled data. We desire a more systematic framework for combining strongly and weakly labeled data. In this section, we present a more general and methodical framework for WEASL. Figure 5.1 outlines our general method for WEASL.

The main idea here is that it formulates the problem as a constraint form of semi-supervised learning. In MIL, negative bags are known to contain only negative instances, meaning, labels for all instances in negative bags are known. Thus, instances in negative bags can be considered as similar to labeled instances from strongly labeled data. For positive bags, on the other hand, this is not true.

We undertake instances in positive bags as *unlabeled* but with certain "label constraints". The constraint on *this unlabeled data* is that they are grouped into bags, and within each bag of

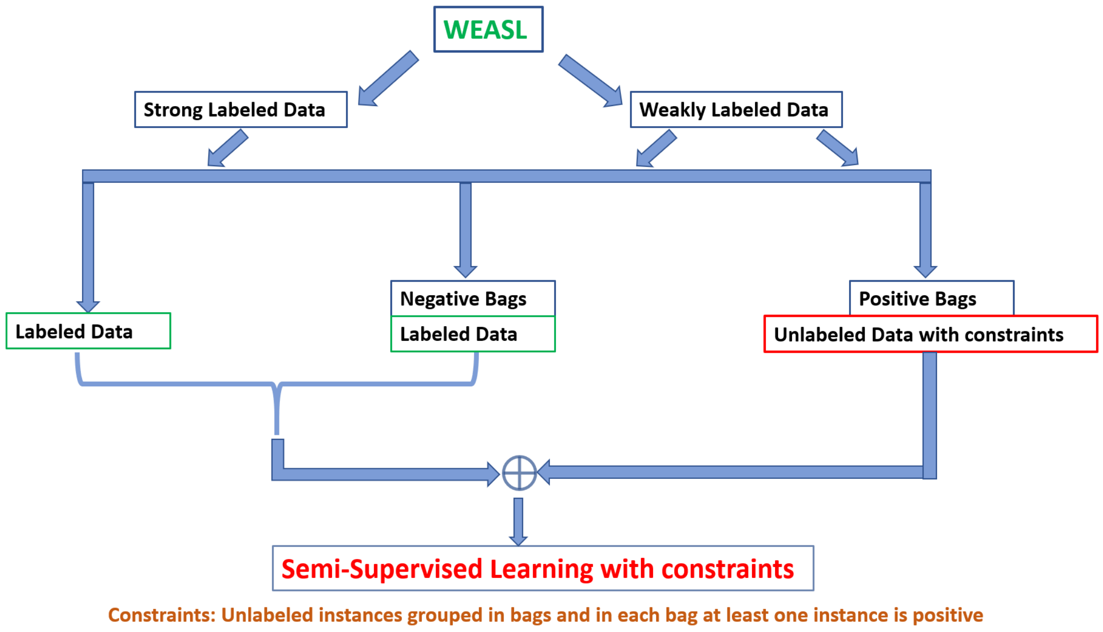

WEASL
Strong Labeled Data
Weakly Labeled Data
Labeled Data
Negative Bags
Labeled Data
Positive Bags
Unlabeled Data with constraints
Semi-Supervised Learning with constraints
Constraints: Unlabeled instances grouped in bags and in each bag at least one instance is positive

Figure 5.1: Unified Framework for Weakly and Strongly Labeled learning (WEASL)

instances at least one instance is positive. Looking at the whole thing together, we have labeled instances on one side and unlabeled instances with certain label constraints on the other side.

Hence, this general form of WEASL can be formulated as semi-supervised learning with constraints on the *unlabeled* data. A variety of methods for semi-supervised learning have been proposed over the years [Belkin et al., 2006, Bennett et al., 1999, Jia and Zhang, 2008, Zhou and Xu, 2007, Zhu and Goldberg, 2009, Zhu et al., 2003]. We start by adopting one of the most popular methods for semi-supervised learning, Manifold Regularization on Graphs [Belkin et al., 2006] for WEASL. We name this variant of WEASL as *graph-WEASL*.

5.4 graph-WEASL

All instances in the negative bags are of negative labels and hence, from here on in our mathematical formulation we will consider them to be part of the strongly labeled dataset. Thus, we have labeled instances from the strongly labeled data and the negatively labeled bags. The unlabeled set with constraints are instances from the positive bags.

Let us represent the supervised dataset as $\mathcal{D}_s = \{(\mathbf{x}_1, y_1), ...(\mathbf{x}_n, y_n)\}$. $y_i \in \{-1, 1\}$ is label for instance $\mathbf{x}_i$ and n is the total number of instances in $\mathcal{D}_s$. The weakly supervised data $\mathcal{D}_w$, is in form of positive bags. Let $\mathcal{D}_w = \{B_1, ..B_T\}$ be the set of T bags where $B_t = \{(\mathbf{x}_{t1}, y_{t1}), ..., (\mathbf{x}_{tm_t}, y_{tm_t})\}$ is a positive bag of instances. Labels y for instances in the bags are unknown but at least one of them is $+1$. $m = \sum_{t=1}^{T} m_t$ is the total number of instances in all positive bags.

Let us represent the whole data $\mathcal{D}$ as $\mathcal{D} = \{(\mathbf{x}_1, y_1), ...(\mathbf{x}_n, y_n), (\mathbf{x}_{n+1}, y_{n+1}), ...(\mathbf{x}_{n+m}, y_{n+m})\}$. Without loss of generality, we have assumed that instances are ordered such that first n are from $\mathcal{D}_s$ and $n + 1$ to $n + m$ are from $\mathcal{D}_w$. Instances $n + 1$ to $n + m_1$ are from bag B_1 and so on. We will denote the start and end indices of instances from bag B_t in $\mathcal{D}$ as p_t and q_t. The instance space is denoted by $\mathcal{X}$. The total number of instances in $\mathcal{D}$ is $N = n + m$.

Labels for the first n instances in $\mathcal{D}$ are known. The labels for the rest of the instances are unknown but constraint by the following relationship

$$max(y_{p_t}, y_{p_{t+1}}, ..., y_{q_t}) = 1 \ \forall \ t = 1 \ to \ T \tag{5.1}$$

The goal is to learn the function mapping $f : \mathcal{X} \rightarrow R$, which maps the instance space to a decision score. f is assumed to be smooth and let us denote the Reproducing Kernel Hilbert Space (RKHS) of f as $\mathcal{H}$.

Since the labels for instances $\mathbf{x}_{n+1}$ to $\mathbf{x}_{n+m}$ are essentially unknown and yet constrained by Eq 5.1, we can formulate this learning process as a constrained form of semi-supervised learning. A particularly well-known method for semi-supervised learning is manifold regularization on graphs [Belkin et al., 2006]. This graph based semi-supervised learning method is inductive, and that is one of the reasons for choosing it.

5.4.1 Manifold Regularization approach for WEASL

In graph based semi-supervised learning, all instances are assumed to be connected by a graph $G = (V, E)$, where the vertices V are instances in the data. Here, we assume a kNN graph [Zhu and Goldberg, 2009] where a vertex $\mathbf{x}_i$ is connected to another vertex $\mathbf{x}_j$ by a non-zero weight w_{ij} if $\mathbf{x}_i$ is among the k-nearest neighbour of $\mathbf{x}_j$ and vice versa. The edge weight w_{ij} is then defined by Gaussian Kernel, $w_{ij} = exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2})$. σ is the bandwidth parameter for weights. Clearly, when $\mathbf{x}_i$ and $\mathbf{x}_j$ are not connected $w_{ij} = 0$. The overall graph is parametrized through a symmetric weight matrix W whose elements are w_{ij}. Finally, the unnormalized graph laplacian L is defined by $L = D - W$, where D is diagonal matrix and $D_{ii} = \sum_j w_{ij}$.
Manifold regularization on graphs for SSL solves the following optimization problem

$$\min_f \ \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 + \lambda_1 ||f||_{\mathcal{H}}^2 + \lambda_2 ||f||_I^2 \tag{5.2}$$

The first term is simply the squared loss over the labeled instances. The first regularization term $||f||_H^2$ is the standard RKHS norm which is used to impose smoothness conditions on f. The second penalty term $||f||_I^2$ is a regularization term for intrinsic structure of data distribution. This terms ensures that the solution is smooth with respect to data distribution as well. Together the two regularization terms control the complexity of the solution over both RKHS and intrinsic geometry of data distribution.

For WEASL, we need to factor in the weak label information of positive bags in the above optimization problem. To do this, we solve following optimization problem

$$\begin{aligned} \min_f \ & \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 + \lambda_1 ||f||_{\mathcal{H}}^2 + \lambda_2 ||f||_I^2 \\ & + \frac{\lambda_3}{T} \sum_{t=1}^{T} (1 - \max_{j=p_t,\dots,q_t} f(x_j))^2 \end{aligned} \tag{5.3}$$

In the above formulation, the last term is the squared loss for each positive (+1) bag. To compute the loss for each bag the output of each bag is determined by the maximal output instance.

Unlike the case of semi-supervised learning, the above WEASL formulation is a non-convex optimization problem. To solve the optimization problem, we rewrite the optimization problem in Eq 5.3 using slack variables.

$$\begin{aligned}\min_{f,\xi}\ & \sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 + \lambda_1||f||_{\mathcal{H}}^2 + \lambda_2||f||_I^2 + \lambda_3\sum_{t=1}^{T}\xi_t^2 \\ s.t\ & 1 - \max_{j=p_t,\dots,q_t} f(x_j) \leq \xi_t,\ \ t = 1,...,T \\ & \xi_t \geq 0,\ \ t = 1,...,T\end{aligned} \tag{5.4}$$

ξ_t are the slack variables for loss on positive bags. Also, note that we have factored in the normalization terms (n and T) in the regularization parameters. To solve the above problem we need a finite dimensional form for f.

Using Representer Theorem [Smola and Schölkopf, 1998], the solution to the above problem can be expressed as $f(\mathbf{x}) = \sum\limits_{i=1}^{N} \alpha_i k(\mathbf{x}, \mathbf{x}_i)$, where $k(\cdot,\cdot)$ is the reproducing kernel of $\mathcal{H}$. Let us denote the $N \times N$ kernel gram matrix over the training data $\mathcal{D}$ with K.

Now, let Y be a N dimensional label vector where $Y = [y_1, y_2, ...y_n, 0, 0....0]$. Y_i is label for the first n instances, which are labeled, and 0 for the rest. Also, let J be $N \times N$ diagonal matrix where $J_{ii} = 1$ for $i = 1\ to\ n$ and $J_{ii} = 0$ for $i = n+1\ to\ (n+m)$.

Using the expression for f, the squared loss term for labeled instances can be written as $\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 = (Y - JK\alpha)^T(Y - JK\alpha)$. The intrinsic norm $||f||_I$ is estimated using the graph laplacian matrix L by $||f||_I^2 = \frac{1}{N^2}f^TLf$ [Belkin et al., 2006]. Hence, $||f||_I^2 = \frac{1}{N^2}\alpha^TKLK\alpha$. So our final optimization problem becomes

$$\begin{aligned}\min_{\alpha,\xi}\ & (Y - JK\alpha)^T(Y - JK\alpha) + \lambda_1\alpha^TK\alpha \\ & + \lambda_2\frac{1}{N^2}\alpha^TKLK\alpha + \lambda_3\sum_{t=1}^{T}\xi_t^2 \\ s.t\ & 1 - \max_{j=p_t,\dots,q_t} K_j'\alpha \leq \xi_t\ \ t = 1,...,T \\ & \xi_t \geq 0,\ \ t = 1,...,T\end{aligned} \tag{5.5}$$

In 5.5, K_j is the j^{th} column of kernel matrix K.

5.4.2 Optimization Solution

The objective function in the optimization problem in Eq 5.5 is a convex differentiable function. The first set of constraints $(1 - \max\limits_{j=p_t,\dots,q_t} K_j'\alpha \leq \xi_t)$ are non-convex but differences of two convex functions. Convex Concave Procedure (CCCP) [Smola et al., 2005], is a well known method of sequential convex programming to handle problems like this. It is an iterative method in which the non-convex function is converted into a convex function using Taylor series approximation at the current solution.

For an objective or constraint in the form of $g(x) - h(x)$, where $g(x)$ and $h(x)$ are convex, a convex approximation at $x^{(k)}$ is obtained as $g(x) - h(x^{(k)}) - \nabla h(x^{(k)})(x - x^{(k)})$, where $\nabla h(x^{(k)})$ is gradient of $h(x)$ at $x^{(k)}$.

$max()$ is a non-smooth function and hence we need the subgradient of $max()$ for the Taylor series expansion. The subgradient of $\underset{j=p_t \cdots q_t}{max} K'_j\alpha$ can be defined as [Cheung and Kwok, 2006]

$$\partial(\underset{j=p_t,\ldots,q_t}{max} K'_j\alpha) = \sum_{j=p_t}^{q_t} \delta_{tj} K_j \tag{5.6}$$

The δ_{tj}'s are defined as

$$\delta_{tj} = \begin{cases} \frac{1}{r_t}, & \text{if } K'_j\alpha = \underset{u=p_t,\ldots,q_t}{max} K'_u\alpha \\ 0, & \text{otherwise} \end{cases} \tag{5.7}$$

r_t is the number of instances which maximizes the output $K'_j\alpha$ in t^{th} bag. Hence, all instances in the bag for which maximum is achieved are active in the subgradients. Now, we can rewrite the non-convex constraints in k^{th} iteration of CCCP using the above subgradient. The non-convex constraint $1 - \underset{j=p_t,\ldots,q_t}{max} K'_j\alpha$ in k^{th} iteration becomes

$$1 - \underset{j=p_t \cdots q_t}{max} K'_j\alpha \approx 1 - (\underset{j=p_t \cdots q_t}{max} K'_j\alpha^{(k)} + \sum_{j=p_t}^{q_t} \delta_{tj}^{(k)} K'_j(\alpha - \alpha^{(k)})) \tag{5.8}$$

Hence, the final optimization problem to solve in k^{th} iteration of CCCP is

$$\begin{aligned} \underset{\alpha,\xi}{min}\ & (Y - JK\alpha)^T(Y - JK\alpha) + \lambda_1\alpha^T K\alpha \\ & + \lambda_2 \frac{1}{N^2}\alpha^T KLK\alpha + \lambda_3 \sum_{t=1}^{T} \xi_t^2 \\ & s.t \\ & 1 - (\underset{j=p_t \cdots q_t}{max} K'_j\alpha^{(k)} + \sum_{j=p_t}^{q_t} \delta_{tj}^{(k)} K'_j(\alpha - \alpha^{(k)})) \leq \xi_t \\ & \qquad\qquad\qquad\qquad t = 1, \ldots, T \\ & \xi_t \geq 0,\ t = 1, \ldots, T \end{aligned} \tag{5.9}$$

The objective function in optimization problem of Eq 5.9 is convex and the constraints are linear. The overall problem is a convex QP problem. In CCCP, the above optimization problem is iteratively solved until convergence.

Once α has been obtained the output corresponding to any test point $\mathbf{x}_{test}$ can be obtained as $f(\mathbf{x}_{test}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_{test}, \mathbf{x}_i)$. It is worth noting that graph-WEASL can also predict output corresponding to each instance in a bag. Hence, our WEASL approach can also be used for temporal localization of acoustic events in a recording. The bag-level prediction is done by using the $max()$ function over instance outputs.

5.5 Experiments and Results

5.5.1 Experimental Setup

We evaluate the proposed WEASL frameworks on both audio event and acoustic scene recognition tasks. In our experiments, we add a small amount of strongly labeled data to the pool of weakly labeled data to learn event or scene detectors using WEASL and compare it with weakly supervised learning. For weakly supervised learning we use miSVM approach. For naive-WEASL, we again use miSVM approach to integrate the strongly and weakly labeled data.

The details specific to the audio event and scenes are given in the respective sections. We describe the common experimental set up here. All audio recordings in our experiments are sampled at $44.1KHz$ sampling frequency. 20 dimensional MFCC features along with delta and acceleration coefficients are used to parametrize audio recordings. The $\vec{F}$ features over the MFCCs as described in Section 3.6.1 are used to represent segments or instances in the bags. The GMM component size K is 64 or 128.

Exponential Chi-square (χ^2) kernels in form of $exp(-\gamma d(x, y))$ have been known to work remarkably well with histogram features, including for detection of acoustic concepts [Rawat et al., 2013][Kumar and Raj, 2016c]. $d(x, y)$ is χ^2 distance. Hence, we use exponential χ^2 kernels for miSVM and naive-WEASL. The parameter γ is set as the inverse of mean χ^2 distance between training points. The slack parameter C in SVM training is selected through cross-validation on the training data.

For graph-WEASL, the kNN graph is constructed with k as 20 or 40. We evaluate and show results for both cases. The bandwidth σ for graph weights is set to 1.0 for all experiments. The kernel K is again exponential χ^2 kernel as used in miSVM and naive-WEASL. The parameter λ_3 in graph-WEASL is simply set as $\lambda_3 = n/T$, that is the ratio of the number of supervised instances to the number of positive bags. The other two regularization parameters λ_1 and λ_2 are selected through cross-validation over a grid of 10^{-3} to 10^3. We use [Grant and Boyd] for solving the QP problem in Eq 5.9.

5.5.2 Audio Event Recognition

We consider a set of 10 acoustic events namely, *Chainsaw (C), Clock Ticking (CT), Crackling Fire (CF), Crying Baby (CR), Dog Barking (DB), Helicopter (H), Rain (RA), Rooster (RO), Seawaves (SE), Sneezing (SN).* The events are part of ESC-10 [Piczak, 2015b] dataset which provides us strongly labeled data for these events. We obtain the weakly labeled training data from YouTube.

For each event, we use the event name as the search query on YouTube. To get more sound-oriented results the keyword "sound" is attached to each event name (e.g. Chainsaw sound). We consider the audio from the top 60 returned video results, from which we filter out very long recordings. Finally, we are left with 40 weakly labeled recording for all acoustic events except *Crackling Fire* (35) and *Rain*(10). Recordings for Rain are relatively longer, and hence the total duration of audio for it is in a similar range compared to others. The total duration of all 365 recordings is around 5.1 hours. Audio recordings are segmented to form bags and instances, and we use the average duration of each event in the strongly labeled data as segment size.

We need a large test dataset for a comprehensive evaluation of all methods. For all audio events, we obtain test data from *Freesound* [1]. The number of test recordings for each event are as follows: Chainsaw - 78, Clock Ticking - 69, Crackling Fire - 66, Crying Baby - 88, Dog Barking - 100, Helicopter - 39, Rain - 76, Rooster - 83, Sea Waves - 48, Sneezing - 75. A total of 722 *recordings* spanning over 13 *hours* is used for evaluation. Note that results given in the next paragraphs are bag-level or recording-level prediction results. The recording level prediction scores are obtained by taking a $max()$ over segment level scores.

[1]www.freesound.org

Table 5.1: Results (AP) using miSVM and naive-WEASL

Events	$K = 64$		$K = 128$	
	miSVM	naive-WEASL	miSVM	naive-WEASL
Chainsaw	0.571	0.671	0.436	0.649
Clock Ticking	0.563	0.658	0.542	0.689
Crackling Fire	0.382	0.458	0.421	0.522
Crying Baby	0.558	0.630	0.613	0.691
Dog Barking	0.237	0.348	0.442	0.520
Helicopter	0.363	0.384	0.393	0.431
Rain	0.263	0.414	0.252	0.374
Rooster	0.392	0.444	0.466	0.533
Seawaves	0.164	0.162	0.176	0.171
Sneezing	0.320	0.402	0.327	0.424
MAP	**0.381**	**0.457**	**0.407**	**0.500**

The ESC-10 dataset contains 40 positive examples for each event, resulting in a total of 400 supervised data (positive or negative) for any given event. The dataset comes pre-divided into 5 folds. We include recordings from 4 out of 5 folds for training. This amounts to adding about 25 minutes of strongly labeled data to the pool of weakly labeled data. The ratio of the total duration of weakly to strongly labeled data is around 12.5. Experiments are run all 5 ways (leaving one fold in each case), and the average across all 5 runs are reported. We use Average Precision (AP) as the performance metric. The mean average precision (MAP) over all events are also shown.

Table 5.1 shows AP values for different audio events using miSVM and naive-WEASL. Results for acoustic features using both $K = 64$ and $K = 128$ Gaussians in the GMMs are shown. One can observe that adding a small amount of strongly labeled data to the pool of weakly labeled data is extremely helpful. WEASL even in the naive formulation gives a considerable improvement over only weakly supervised miSVM. Improvements in MAP of around **20**% for $K = 64$ and **22.8**% for $K = 128$ is observed. As far as individual events are concerned, for several events such as *Chainsaw, Crackling Fire, Rain, Sneezing* relative improvements in the range of $25 - 50\%$ in AP can be observed.

Table 5.2 shows AP values for graph-WEASL approach. We observe that graph-WEASL improves over naive-WEASL another $8.6 - 9.6\%$. This amounts to about **31.5**% and **33.5**% improvement over miSVM for $K = 64$ and 128 and respectively. For graph-WEASL, the performance remains more or less consistent for the two values of k in the kNN graph. Overall, our results show that a small amount of strongly labeled can play a significant role in reducing the effect of signal-noise and label-noise in weakly labeled data obtained from the web.

5.5.3 Recognition of Strongly Labeled Events Set

One crucial advantage of weakly supervised AED using methods such as miSVM is that we can obtain temporal locations of events in audio recordings. We showed this in Chapter 3. The WEASL framework we presented here can do the same as well.

The procedure remains the same as in MIL. We can predict on each instance of the bag and then the temporal localization can be done through predictions on the segments. Our goal is

Table 5.2: Results (AP) using graph-WEASL

Events	$C = 64$		$C = 128$	
	$kNN = 20$	$kNN = 40$	$kNN = 20$	$kNN = 40$
Chainsaw	0.534	0.531	0.578	0.574
Clock Ticking	0.669	0.672	0.713	0.713
Crackling Fire	0.571	0.584	0.579	0.574
Crying Baby	0.749	0.741	0.767	0.772
Dog Barking	0.305	0.305	0.439	0.439
Helicopter	0.448	0.458	0.565	0.526
Rain	0.421	0.403	0.382	0.414
Rooster	0.612	0.610	0.695	0.678
Seawaves	0.178	0.174	0.194	0.198
Sneezing	0.523	0.523	0.519	0.519
MAP	**0.501**	**0.500**	**0.543**	**0.541**

Table 5.3: Temporal Localization Results

Events	$K = 64$			$K = 128$		
	miSVM	naive-WEASL	graph-WEASL	miSVM	naive-WEASL	graph-WEASL
Chainsaw	0.455	0.646	0.497	0.372	0.640	0.668
Clock Ticking	0.702	0.766	0.846	0.704	0.856	0.800
Cracking Fire	0.715	0.841	0.914	0.691	0.878	0.886
Crying Baby	0.861	0.923	0.980	0.846	0.931	0.983
Dog Barking	0.510	0.726	0.772	0.621	0.810	0.834
Helicopter	0.691	0.722	0.741	0.733	0.776	0.829
Rain	0.130	0.498	0.622	0.086	0.516	0.586
Rooster	0.827	0.883	0.967	0.838	0.926	0.957
Seawaves	0.514	0.612	0.693	0.563	0.662	0.714
Sneezing	0.683	0.828	0.916	0.693	0.888	0.960
MAP	0.609	0.745	0.795	0.615	0.788	0.822

to evaluate temporal localization performances for WEASL as well. However, the test set used in the previous section has been obtained from Freesound and are weakly labeled. Temporal localization evaluation implies instance level evaluation of all methods. Hence, we design an experiment which is similar to the evaluation of predictions at the segment level.

We evaluate our framework on strongly labeled data. Each recording is an exemplar of the event. We relate this test to temporal localization experiments. The procedure for temporal localization would have been to recognize the event in each segment of the recording. For strongly labeled data each recording is just one instance, rather than a bag.

We use the left out fold from the ESC-10 dataset for evaluating instance-level performance. Results accumulated over all 5 runs are reported.

Table 5.3 shows AP for instance-level prediction of the sound events. Once again we notice that WEASL based approaches give $22.3 - 33.6\%$ improvements over learning from only weakly labeled data. Graph-based WEASL is once again superior to other methods. Overall this shows that our proposed method is suitable for instance level prediction as well.

5.5.4 Acoustic Scene Recognition

The procedure for acoustic scenes remains same as acoustic events. We work with a total of 15 acoustic scenes from DCASE [Mesaros et al., 2016] dataset, which is also the source of strongly labeled data in our experiments. Weakly labeled training data is again obtained from YouTube through a procedure similar to the audio events. In this case, we create weakly labeled data of 40 recordings per scene, totaling 600 recordings which span over 27 hours. Once again we obtain test data from Freesound. A total of 928 test recordings ($\sim$ 38 hours) are present in the test set. The test data contains an average of 61 recording per scene with a minimum of 53 for *Forest Path* and a maximum of 71 for *Residential Area*. Segment size is the average duration of scenes in strongly labeled data, which is 30 seconds for all of the scenes. The strongly labeled data from DCASE comes pre-divided into 4 sets, and our experimental approach is the same as before. We use 3 out of 4 folds in WEASL and perform experiments all four ways. As before, average results across all 4 runs are reported.

It is worth noting that acoustic scenes are acoustically more complex. Intra-class variation is far greater than the audio events. Both training and the test data contains a significant amount of within-class variations, which makes recognition of acoustic scenes from weakly labeled data a much harder problem. This is evident in the low average precision for most classes. Moreover, the weakly labeled training data from YouTube is also noisier than the audio events.

Table 5.4 shows AP values for different methods. For acoustic scenes, we note that the performance of miSVM and naive-WEASL is almost similar as far as the MAP is concerned. Graph-based WEASL, on the other hand, gives a **20**% relative improvement over these methods. graph-WEASL improves AP for almost all scenes. For several scenes such as *Grocery Store, Home, Library, Metro Station, Train, Tram* AP improves by **50**% or more in relative terms. This should be especially noted for *Home*, which turns out to be the hardest scene to detect. graph-WEASL, in this case, more than *doubles* the average precision.

5.6 Discussions and Conclusions

In this chapter, we described a novel learning framework called Weakly and Strongly Labeled (WEASL) framework which leverages labeled data in both weak and strong forms. Within the general framework presented, a variety of other WEASL methods can be developed.

One approach can be a modification of the current graph-WEASL method. It can be modified to utilize loss functions other than squared loss. Hinge loss is a common loss function used in a variety of machine learning algorithms. We can replace the square loss in Eq 5.3 with Hinge Loss. This would amount to solving the following optimization problem

$$\begin{aligned} \underset{f}{min} \ \frac{1}{n}\sum_{i=1}^{n} H(y_i, f(\mathbf{x}_i)) + \lambda_1 ||f||^2_{\mathcal{H}} + \lambda_2 ||f||^2_I \\ + \frac{\lambda_3}{T}\sum_{t=1}^{T} H(1, \underset{j=p_t,\ldots,q_t}{max} f(x_j)) \end{aligned} \tag{5.10}$$

$H(y, f(x))$ is the Hinge loss with y as true label and $f(x)$ as decision output. Hinge loss is a convex function, and the above optimization problem can be solved by following a procedure similar to current graph-WEASL.

Table 5.4: Acoustic Scene Results (AP). graph-WEASL (gWEASL), naive-SWSL (nWEASL)

Events	$C = 64$			$C = 128$		
	miSVM	nWEASL	gWEASL	miSVM	nWEASL	gWEASL
Beach	0.129	0.132	0.153	0.140	0.150	0.122
Bus	0.087	0.099	0.110	0.094	0.104	0.106
Cafe	0.264	0.220	0.256	0.272	0.246	0.301
Car	0.066	0.083	0.078	0.069	0.085	0.079
City Center	0.137	0.122	0.121	0.132	0.129	0.121
Forest	0.074	0.076	0.081	0.090	0.095	0.092
Grocery Store	0.053	0.061	0.081	0.059	0.066	0.098
Home	0.048	0.060	0.116	0.049	0.067	0.098
Library	0.073	0.106	0.113	0.070	0.086	0.088
Metro Station	0.090	0.096	0.133	0.086	0.096	0.119
Office	0.105	0.093	0.080	0.099	0.091	0.069
Park	0.115	0.119	0.133	0.129	0.153	0.142
Residential Area	0.092	0.112	0.099	0.085	0.113	0.111
Train	0.070	0.066	0.105	0.074	0.068	0.092
Tram	0.106	0.132	0.144	0.111	0.144	0.157
MAP	**0.101**	**0.105**	**0.120**	**0.104**	**0.113**	**0.120**

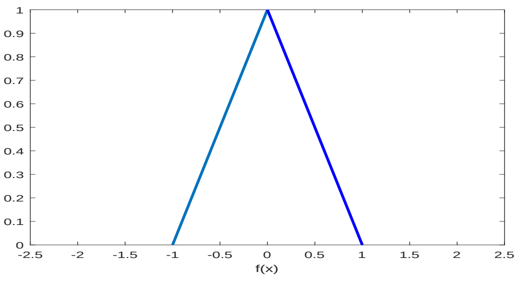


Figure 5.2: Hat Loss in semi-supervised SVM

One can also use semi-supervised SVMs for WEASL. The semi-supervised SVM solves the following optimization problem

$$\min_{f} \frac{1}{n}\sum_{i=1}^{n} H(y_i, f(\mathbf{x}_i)) + \lambda_1 ||f||^2_{\mathcal{H}} + \lambda_2 \sum_{i=n+1}^{N} \hat{H}(y_i, f(\mathbf{x}_i)) \tag{5.11}$$

H is the usual Hinge loss over the labeled data and $\hat{H}$ is a non-convex hat shaped loss function (Figure 5.2) over predicted $f(x)$ for the unlabeled data point. Once again the positive bags can be treated as unlabeled data, with certain "label constraints". Hence, similar to our graph-WEASL framework we can modify the above optimization function for WEASL. This is shown in Eq 5.12. Solving Eq 5.12 is more complicated than the previous cases due to the non-convex $\hat{H}$, however, it can still be done.

$$\underset{f}{min}\ \frac{1}{n}\sum_{i=1}^{n} H(y_i, f(\mathbf{x}_i)) + \lambda_1 ||f||^2_{\mathcal{H}} + \lambda_2 \sum_{i=n+1}^{N} \hat{H}(y_i, f(\mathbf{x}_i)) + \frac{\lambda_3}{T}\sum_{t=1}^{T} H(1, \underset{j=p_t,\ldots,q_t}{max} f(x_j)) \quad (5.12)$$

Optimization problems Eq 5.10 and Eq 5.12 are two other methods within the general framework of WEASL. Deep learning methods for WEASL can also be explored, although, it might not follow the general outline of WEASL we presented here. However, deep learning methods might do well with larger datasets. The graph-based semi-supervised method we presented here might lead to computational issues for very large datasets. One simple way to explore deep learning for WEASL can be through transfer learning. Here, strongly and weakly labeled data can be considered as two different domains and then transfer learning based methods can be used to transfer knowledge from one to the other. We present deep learning methods to transfer knowledge from one domain to another for sounds in the next chapter, although not in the context of weakly and strongly supervised learning.

Chapter 6

Transfer Learning for Sounds

If you have knowledge, let others light their candles in it.

-Margaret Fuller

In the previous chapters, we introduced the idea of weakly labeled learning for sounds. Weak label learning led to large-scale audio event detection and in this chapter, we show how large-scale weakly supervised learning of sounds can be further utilized in solving other tasks.

6.1 Introduction

Weak labeling addresses data availability constraints to a large extent. However, creating large datasets along the lines of Audioset is still not easy. Moreover, in some cases, it might just be difficult to collect large amounts of labeled data in any form. For examples, there are sound events which are inherently rare. Deep learning methods such as those based on CNN are not directly useful in such cases. However, as pointed out by Ellis et al. in Future Perspectives [Ellis et al., 2018], one can attempt to address this problem by transferring knowledge from a model trained on a large dataset. Motivation also comes from computer vision where deep CNN models have been successfully used to transfer knowledge from one domain to another as well as from one task to another [Oquab et al., 2014, Yosinski et al., 2014]. This approach, more generally referred to as transfer learning [Pan and Yang, 2010, Weiss et al., 2016] remains more or less unexplored in the context of sound events and scenes.

Transfer learning is an important and extremely useful learning framework in machine learning. Humans often apply knowledge from one domain to another in their daily life. This comes naturally to us as we have highly evolved association capabilities. The idea of using knowledge from one domain or task to another in machine learning is grounded in our natural abilities. Hence, we expect that if two domains or tasks are related then a model trained on one should be helpful in solving the other task. Its significance increases even further when labeled data for one of the task is scarce. In these cases, training a new model on a small amount of labeled data might not lead to good generalization. However, the knowledge from the first model which trained on a larger dataset could be very useful. The general idea of transfer learning is shown in Figure 6.1.

The model trained on the first task, shown in Figure 6.1 as the source model, is used in

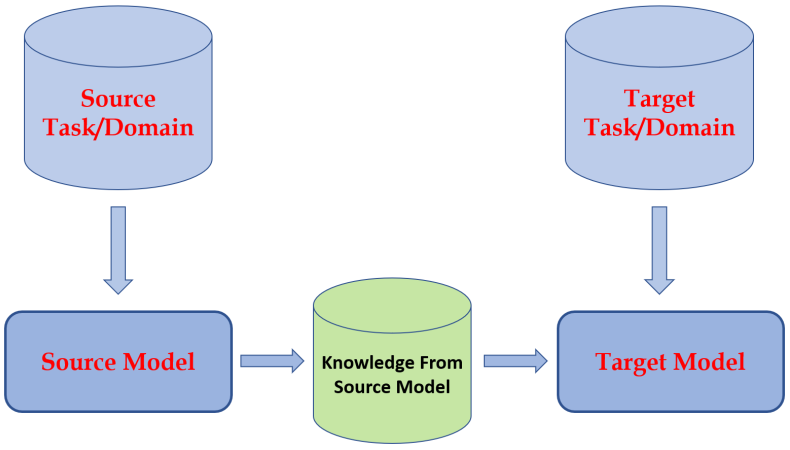


Figure 6.1: The basic idea behind transfer learning

learning the target task. We follow [Pan and Yang, 2010] in giving a formal definition of transfer learning. Let $\mathcal{D}_S$ and $\mathcal{T}_S$ be source domain and task. Let $\mathcal{D}_T$ and $\mathcal{T}_T$ be the target domain and task. Let $f_S()$ and $f_T()$ be source and target domain predictive functions.

Given the source domain and task $\mathcal{D}_S$ and $\mathcal{T}_S$, the goal of transfer learning is to improve the learning of target function $f_T()$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$. In this setting, either $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$. That is, either the source and target domains are different, or the source and target tasks are different. They can both be different as well.

The domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ is defined by an instance space, $\mathcal{X}$ and a marginal distribution $P(X)$, $X = \{x_1, ..., x_n\} \in \mathcal{X}$ over a set of data points. The source domain data is defined as $D_S = \{(x_{S_1}, y_{S_1}), ..., (x_{S_n}, y_{S_n})\}$, a collection of data points from $\mathcal{X}_S$ and their labels from $\mathcal{Y}_S$. Similarly, target domain data is $D_T = \{(x_{T_1}, y_{T_1}), ..., (x_{T_n}, y_{T_n})\}$. The task consists of learning the predictive function $f()$, given the data points $x_i \in X$ and their labels $y_i \in Y$. If the domains are different then $\mathcal{X}_S \neq \mathcal{X}_T$ or $P_S(X) \neq P_T(X)$. If the tasks are different then either $\mathcal{Y}_S \neq \mathcal{Y}_T$ or the conditional probability distributions are different, $P(Y_S|X_S) \neq P(Y_T|X_T)$. The overall goal of transfer learning is to use the knowledge from the source domain and task, to improve the learning of the target task. The assumption behind most of the successful transfer learning works is that the source and target domain are related and knowledge transfer is meaningful. However, this might not be necessarily true and one needs to be careful to avoid *negative transfer*.

6.1.1 Related Works

Transfer learning has been widely used in a large number of computer vision and natural language processing (NLP) tasks. It has been used for text classification, sentiment analysis and classification, named entity recognition, to mention a few [Dai et al., 2007, Guo et al., 2009, Long et al., 2014, Melville et al., 2009, Pan et al., 2010]. In computer vision, transfer learning has been applied to object recognition and video concept or multimedia event detection [Duan et al., 2012a, Lampert et al., 2009, Lim et al., 2011, Shao et al., 2015, Tommasi et al., 2010, Yang et al., 2007].

Deep learning models are capable of exploiting large datasets, and this makes them well suited for knowledge transfer through them. Hence, a large number of recent transfer learning works have used deep neural networks to transfer learning [Long et al., 2015, Oquab et al., 2014, Sharif Razavian et al., 2014, Zeiler and Fergus, 2014]. Transferability of features have been done and also exhaustive empirical analysis have been done [Kornblith et al., 2018, Yosinski et al., 2014], which have lead to better insights into transfer learning for computer vision tasks. Convolutional neural networks, being the dominant neural network framework in computer vision, have been used in most of the works.

Transfer learning can be done through deep networks in a variety of ways. However, two popular approaches have been to (a) *fine-tune* the network for the target task [Kornblith et al., 2018, Tajbakhsh et al., 2016], and (b) extract features or learn representations using the source network for the data in the target domain [Oquab et al., 2014, Sharif Razavian et al., 2014]. In the first case, the source network itself is adapted to solve the target task. In the second case, the source network serves as a feature extraction system; the features are then further used with a different learning algorithm, most commonly a simpler one which does not require a massive amount of data to learn. It also possible that the networks trained on the source task are fine-tuned first before it is used to extract features for the target task. In this case, the idea is to learn discriminative representations by using the labeled data from the target task. Multi-task learning [Caruana, 1997] is another popular form of transfer learning where two different domains simultaneously teach each other.

Transfer learning in computer vision has been successful because of the availability of large datasets such as Imagenet, which provides a large collection of labeled examples for a large number of visual objects. This allows one to train deep models which can learn enough information from the source data to be useful in solving other tasks. For sounds, the primary problem has been lack of such large-scale datasets; by large scale, we imply both, the vocabulary of sound events as well as the overall dataset size. The vocabulary of sound events is critical here. To successfully transfer knowledge across different domains and tasks, the larger the class set, the better it is. Similar to computer vision, a deep network trained on a large dataset with a wide variety of sound events can be useful in solving audio tasks beyond sound events.

Compared to computer vision, there has been little work on transfer learning for audio tasks. Most of the works are in speech and music domain [Coutinho et al., 2014, Deng et al., 2013, Huang et al., 2013, Van Den Oord et al., 2014]. [Huang et al., 2013] proposed a multilingual DNN, with shared hidden layers for speech recognition. The framework follows the multi-task learning approach. [Van Den Oord et al., 2014] used supervised pre-training on a large dataset to learn discriminative features for music classification tasks. [Deng et al., 2013, Van Den Oord et al., 2014] again learns features through transfer learning using deep denoising autoencoders. They apply it in the domain of emotion recognition in speech. In another early work [Kumar et al., 2014], we looked into how a model trained on one set of sound events performs on another set of sound events. The context, in this case, was to explore the idea of sound objects, rather than explicit transfer learning. [Ntalampiras, 2017] interestingly explores emotions in music and sounds. They use a transfer learning framework where sounds and music aid each other for the automated understanding of emotions evoked through sounds and music. [Lim et al., 2016] used fine-tuning for transfer learning from speech to sounds. They first train a network using speech data and then fine-tune it for sound event classification. [Wang and Metze, 2017] used the SoundNet architecture from [Aytar et al., 2016] to obtain representations for audio recordings.

To circumvent the problem of lack of large sound event datasets, Soundnet [Aytar et al., 2016] proposed to transfer knowledge from vision models for audio tasks. They use CNN models trained for visual objects and scenes to teach a feature extractor network for audio. However, it remains to be seen how a more direct approach of audio to audio knowledge transfer can be done.

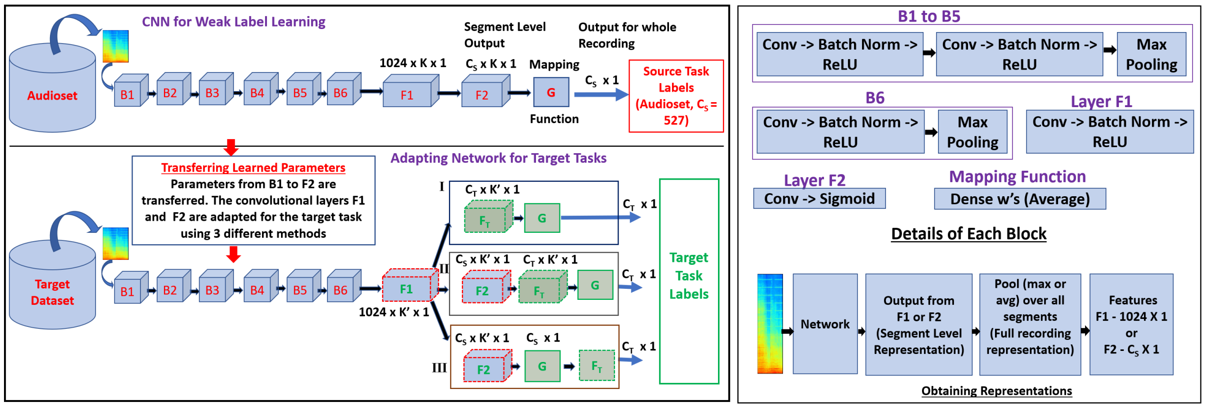

CNN for Weak Label Learning
Audioset
B1
B2
B3
B4
B5
B6
F1
F2
G
1024 x K x 1
Segment Level Output
C_S x K x 1
Mapping Function
Output for whole Recording
C_S x 1
Source Task Labels (Audioset, C_S = 527)
Transferring Learned Parameters
Parameters from B1 to F2 are transferred. The convolutional layers F1 and F2 are adapted for the target task using 3 different methods
Adapting Network for Target Tasks
Target Dataset
1024 x K' x 1
I
II
III
C_T x K' x 1
C_S x K' x 1
C_S x 1
C_T x 1
F_T
Target Task Labels
B1 to B5
Conv -> Batch Norm -> ReLU
Max Pooling
B6
Layer F1
Layer F2
Conv -> Sigmoid
Mapping Function
Dense w's (Average)
Details of Each Block
Network
Output from F1 or F2 (Segment Level Representation)
Pool (max or avg) over all segments (Full recording representation)
Features F1 - 1024 X 1 or F2 - C_S X 1
Obtaining Representations

Figure 6.2: Transfer Learning Framework
Top Left and Right: Deep CNN ($\mathcal{N}_A^{wlat}$) for Weakly Labeled Audio. ***Bottom Left:*** Adapting CNN for target task. 3 different methods (I, II, III) are proposed. Parameters from B1 to F1 (or up to F2) are transferred. F1 and (or) F2 onwards are adapted for target task. Newly added layers are shown in green outline. Layers which are updated during task adaptive training are shown in dashed outline. ***Bottom Right***: Obtaining representations for audios. Network can be $\mathcal{N}_S$ or one of $\mathcal{N}_T^I$, $\mathcal{N}_T^{II}$, $\mathcal{N}_T^{III}$.

In our work [Kumar et al., 2018], we proposed methods to effectively transfer knowledge from a CNN based sound event model trained on a large dataset (Audioset). We first train a deep CNN model on Audioset, a dataset which provides weakly labeled audio examples from YouTube for 527 sound events. The network is trained on the weakly labeled dataset using the methods proposed in the previous chapter. We then use the network to learn representations for the target tasks. Our method achieves state of the art performance on sound event classification task on data from a different domain. Moreover, we show that transfer learning can not only be used for acoustic scene classification but instead can also establish commonsense relations between scenes and sound events.

6.2 Transfer and Representation Learning

We present methods for transfer learning using a deep convolutional neural network trained on a large scale dataset. More specifically, we consider Audioset, which contains weakly labeled recordings for 500 sound events. Hence, the source domain and task are Audioset and sound event detection on Audioset, respectively. We use the methods described in the previous chapter to train a deep CNN on Audioset. Specifically, we use the network $\mathcal{N}_A^{wlat}$ (Figure 4.9), trained on Audioset as the source model, from which knowledge is used for the target task.

Figure 6.2 shows the overall process for transfer learning. Parameters from the network, $\mathcal{N}_A^{wlat}$ are transferred to learn representations for the target task. Segment level outputs from F1 ($1024 \times K \times 1$) and F2 ($C_S \times K \times 1$) serve as the base representations for the audio recordings. These segments level representations are then mapped to full recording level representations. We apply either $max()$ or $avg()$ for this mapping. Finally, we obtain 1024 (F1) or C_S (F2) dimensional representations for full recordings.

During $\mathcal{N}_A^{wlat}$ training on the source task, the blocks from B1 to B6 embed knowledge from source audio data into $F1$, which is then mapped to source labels by filters in $F2$. This makes

F1 well suited for transfer learning, where it can be used to train classifiers for the target task. Moreover, outputs from $F2$ gives us a distribution over the source labels which itself can be useful for the target task, provided $\mathcal{N}_A^{wlat}$ is trained over a large collection of sound events. We propose two broad methods for representation learning for audios in target tasks using $\mathcal{N}_A^{wlat}$.

6.2.1 Direct Off-the-shelf Representations

In this method, $\mathcal{N}_A^{wlat}$ is treated in a ready to use mode for obtaining representations. Logmel spectrograms of audio recordings from target task are fed into $\mathcal{N}_A^{wlat}$ and segment level representations are obtained from it. These segment level representations, from F1 or F2, are then mapped to recording level representation. As specified before, we use $max()$ or $avg()$ functions to obtain recording level representations. The $max()$ operations takes the maximum in a given dimension across all segments. The $avg()$ function, on the other hand, takes the average over all segments in each dimension. Finally, the representations from F1 leads to a 1024 dimensional vectors, whereas those from F2 leads to C_s dimensional vectors.

6.2.2 Transfer and Adapt for Learning Representations

In the second method, we first adapt the network to the target task and then extract features. We expect that this adaptation will lead to more discriminative features and consequently, better suited for the target classification task. The network is adapted for the target task by fine-tuning the network parameters to the target task. We propose three methods to achieve this goal. The methods are shown in Figure 6.2. In all three methods, the parameters from B1 to B6 are transferred and are not updated during the target adaptation training. We use C_T to represent the number of classes in the target task.

Method I ($\mathcal{N}_\mathbf{T}^\mathbf{I}$)

$\mathcal{N}_T^I$ performs a direct adaptation of F1 to the target task. Here, F2 is replaced by a new convolutional layer (F_T) with C_T filters. The filters are of size 1×1. Parameters in F1 and F_T are then updated using the training set of the target task. We refer to this network as $\mathcal{N}_T^I$. Once the network has been fine-tuned, it is then used to obtain representations for the recordings in the target task. The procedure follows the same process as above. Note that, F2 layer representations will not be available using $\mathcal{N}_T^I$.

Method II ($\mathcal{N}_\mathbf{T}^\mathbf{II}$)

In $\mathcal{N}_T^{II}$, a new convolutional layer (F_T) with C_T filters is added to the network,$\mathcal{N}_A^{wlat}$, after F2 layer. This new network, $\mathcal{N}_T^{II}$, is then adapted for the target task. As shown by dashed boundaries in Fig 6.2, during the adaptive training only F1, F2 and F_T are updated. The idea is to capture target specific information by first transitioning to source label space (F2) and from there going to target label space. Representation from both F1 and F2 layers are obtained in this case.

Method III ($\mathcal{N}_\mathbf{T}^\mathbf{III}$)

In $\mathcal{N}_T^{III}$, a new fully-connected layer F_T of size C_T is added after the segment to the recording level mapping function. It implies in $\mathcal{N}_A^{wlat}$, the F_T layer is added after the mapping function layer G. Once again, only F1, F2, and F_T are updated during network adaptation training. The motivation behind this network is same as $\mathcal{N}_T^{II}$, except that it tries to learn the mapping at full recording level instead of segment level. Note that in both $\mathcal{N}_T^{II}$ and $\mathcal{N}_T^{III}$, the activation function in F2 are changed to ReLU from sigmoid in $\mathcal{N}_A^{wlat}$. Once the network has been adapted to the target task, we obtain representations as described previously.

For all three adapted networks $\mathcal{N}_T^{I}$, $\mathcal{N}_T^{II}$ and $\mathcal{N}_T^{III}$, if the target task is a multi-label problem, then the activation in the final layer is kept as sigmoid, and the loss function defined in Eq 4.9 is used. However, if the target task audios have a single label, then we can use *softmax* output with categorical cross entropy loss.

A few points are worth noting. First, the target task can have audio recordings of different length, and our proposed methods can handle such cases efficiently. The network, $\mathcal{N}_A^{wlat}$, has been designed to handle recordings of variable durations and this property is kept intact in all networks for adaptation as well. Moreover, the target task dataset can either be strongly or weakly labeled, and the proposed methods can be used to learn representations in both cases. In fact, the segment level representations can be used in MIL framework, if the target task is a weak label learning problem. Lastly, to emphasize again, the focus is on exploiting $\mathcal{N}_S$ to learn representations for audios in the target task. The fine-tuning on the target task is done to combine knowledge from both the source and target domain to obtain features which are expected to be more discriminative for the target task. Once the representations have been obtained, they can be used for the target task in any manner with a variety of other machine learning algorithms. If the target task is a classification problem, then classifiers such as Support Vector Machines (SVMs) can be easily trained on these representations.

6.3 Target Tasks

We apply the transfer learning to two problems. The source domain data and the source task in both cases, sound event recognition on Audioset. We consider two target tasks, sound event classification and acoustic scene classification. The datasets for both target tasks are relatively much smaller than the Audioset, and we expect that through transfer learning we should be able to exploit the knowledge from Audioset to improve the learning in both cases.

6.3.1 Sound Event Classification On ESC-50 dataset

The first task is again looking at the problem of sound event classification. We work on sound event classification on a well known and standard dataset called ESC-50 dataset [Piczak, 2015b]. ESC-50 dataset consists of a total of 50 sound events. The sound events in this dataset falls under 5 broad categories, namely, *Animals, Natural Soundscapes and Water Sounds, Human Non-Speech Sounds, Domestic Sounds* and *Exterior Sounds.* Sound events from each of these categories are:

- **Animals Sounds**: Dog, Pig, Cat, Rooster, Cow, Frog, Hen, Insects, Sheep, Crow
- **Natural Soundscapes and Water Sounds**: Rain, Sea waves, Crackling Fire, Crickets, Chirping birds, Water drops, Wind, Pouring Water, Toilet Flush, Thunderstorm
- **Human Non-Speech Sounds**: Crying baby, Sneezing, Clapping, Breathing, Coughing, Footsteps, Laughing, Brushing teeth, Snoring, Drinking-sipping

- **Domestic Sounds**: Door Knock, Mouse click, Keyboard typing, Door-wood creaks, Can opening, Washing machine, Vacuum cleaner, Clock alarm, Clock tick, Glass breaking
- **Exterior Sounds**: Helicopter, Chainsaw, Siren, Car horn, Engine, Train, Church bells, Airplane, Fireworks, Hand saw

The dataset contains 40 recordings for each sound event, leading to a total of 2,000 recordings. Each recording is 5 seconds long, and the overall dataset is around 2.8 hours of audio data. The recordings were obtained from Freesound.org [fre].

In transfer learning, the source and target domains are different if either the feature space between the domains are different ($\mathcal{X}_S \neq \mathcal{X}_T$) or the marginal probability distributions are different for the same feature space, that is $P(X_S) \neq P(X_T)$. The tasks are said to be different when either, $\mathcal{Y}_S \neq \mathcal{Y}_T$, that is the label spaces are different, or when the conditional distributions of the labels in the domains are different, that is $P(Y_S|X_S) \neq P(Y_T|X_T)$.

For sound event classification on the ESC-50 dataset, the label spaces are clearly different. In fact, some of the events from the target task not present in the source task. We use Logmel features to represent audio recordings in both source and target domains. However, given the significantly different sources of the two datasets Audioset (YouTube) and ESC-50 (Freesound), the marginals over the data are expected to very different. Put simply, the nature and characteristics of the audio recordings are very different between the two datasets. However, given that the two domains are related, transfer learning can be very effective in improving the learning task on the ESC-50 dataset.

6.3.2 Acoustic Scenes Classification

The second task we consider is acoustic scene classification. We use DCASE 2016 [Mesaros et al., 2016] for acoustic scenes. The dataset consists of 15 different acoustic scenes, namely, *beach, bus, cafe/restaurant, car, city center, forest path, grocery store, home, library, metro station, office, urban park, residential area, train* and *tram*.

The recordings in the dataset were manually recorded at different locations. The recordings were recorded at 44.1 KHz sampling rate and 24-bit resolution. The other details of the recording method and conditions can be found in [Mesaros et al., 2016]. The dataset consists of 78 examples of each acoustic scene. Each recording is 30 seconds long, and the overall dataset consists of around 9.75 hours of audio data.

For this scene classification task, clearly, both the domain and the task are different. However, the two tasks are still related. An acoustic scene is composed of different sound events and knowledge transfer from a large scale dataset should be able to improve the learning of scene recognition. In fact, in our experiments and results, we explicitly show how semantic relations between sounds and scenes can be established through this transfer learning process.

6.4 Experiments and Results

6.4.1 Experimental Setup Details

All audio recordings are sampled at 44.1 KHz sampling rate and represented by Logmel spectrograms. A window of 23 ms moving with an overlap of 11.5 ms is used for obtaining logmel features. 128 mel-bands are used in the feature extraction process. The source network, $\mathcal{N}_A^{wlat}$, is trained on Audioset as described in Section 4.3.

Methods	Mean Accuracy
Piczak [Piczak, 2015a]	64.5 %
Tokozume [Tokozume and Harada, 2017]	71.0 %
SoundNet [Aytar et al., 2016]	74.2 %
Arandjelovic [Arandjelovic and Zisserman, 2017]	79.3%
Human Performance [Piczak, 2015b]	81.3%
Proposed (F1)	**83.5** %

Network	**F1**		**F2**	
	$max()$	$avg()$	$max()$	$avg()$
$\mathcal{N}_{\mathbf{A}}^{\mathbf{wlat}}$	**82.8**	81.6	65.5	64.8
$\mathcal{N}_{\mathbf{T}}^{\mathbf{I}}$	**83.5**	81.3	–	–
$\mathcal{N}_{\mathbf{T}}^{\mathbf{II}}$	**83.5**	81.8	81.9	81.5
$\mathcal{N}_{\mathbf{T}}^{\mathbf{III}}$	83.3	82.6	82.6	81.9

Table 6.1: Transfer Learning: ESC-50 results
Left: ESC-50 Accuracy comparison with baselines. ***Right***: Comparison of accuracies for different representations obtained from different networks

ESC-50 comes pre-divided in 5 folds. 4 folds are used as training set, and the fifth fold is used for the test. This is done all five ways so that each fold becomes the test set and average accuracies over all five runs are reported. The average human accuracy on this dataset is 81.3% [Piczak, 2015b]. The human accuracy numbers vary from as low as 34.1% for *washing machine* sound to as high as 100% for crying babies and dog barking. In general, soundscapes and mechanical sounds are harder for humans to recognize, whereas animals and humans sounds are relatively easier to recognize.

DCASE16 also comes pre-divided into four folds. A similar procedure is used in this case as well, three folds are used for training and fourth as the test set. Once again this is done all four ways, and average numbers over all four runs are reported. The performance of human subjects on scene classification is surprisingly low, at 54.4% [Mesaros et al., 2017]. This performance is considerably lower than the baseline method using MFCCs and Gaussian Mixture Models (GMMs) [Mesaros et al., 2016]. The baseline classification accuracy is 72.5%. The accuracy on individual acoustic scenes varies from 98.6% for *Office* to 13.9% for *Park*.

Acoustic scene classification is considerably harder for humans, as is evident from the low average accuracy. Humans understand acoustic scenes in a very sophisticated manner, often relying on other cues such as vision and familiarity. In [Mesaros et al., 2017], the authors report that human subjects familiar with Finnish soundscapes (the place where examples were recorded), had a higher recognition accuracy of 60.4%. Overall, acoustic scene classification is a complex problem and knowledge transfer using a large sound event dataset can be crucial in improving acoustic scene recognition systems.

For the task adaptive training of $\mathcal{N}_A^{wlat}$, the training set of the target task is used. Learning rate during this process is fixed to 0.0002 and updates are done for 50 epochs, after which the network is used to obtain representations. Linear SVMs [Fan et al., 2008] are then trained on the representations obtained from different methods. The slack parameter C in the SVM formulation is tuned by cross-validation on the training set.

6.4.2 Results: Sound Event Classification on ESC-50

Left table in Tab. 6.1 compares mean accuracy over all 50 classes with state-of-art on the ESC-50 dataset. We outperform SoundNet by **9.3**%, and [Arandjelovic and Zisserman, 2017] by 4.3%. SoundNet is also a transfer learning system where knowledge is transferred from vision to audio. [Arandjelovic and Zisserman, 2017] is again a multi-modal framework where knowledge from both vision and audio are used to learn representations. Our method sets state-of-the-art performance on ESC-50 and outperforms even human accuracy on this task. This shows that our audio to audio transfer learning approach is able to successfully transfer knowledge from the

Hardest 10				**Easiest 10**			
Event	Human	SoundNet	Proposed (F1, $\mathcal{N}_T^{II}$)	Event	Human	SoundNet	Proposed (F1, $\mathcal{N}_T^{II}$)
Washing machine	0.342	0.40	0.53	Siren	0.926	0.82	0.88
Wind	0.458	0.50	0.55	Coughing	0.935	0.85	0.90
Crickets	0.518	0.55	0.88	Cow	0.941	0.78	0.85
Vacuum cleaner	0.577	0.62	0.75	Sheep	0.949	0.78	0.95
Crackling fire	0.634	0.78	0.90	Church bells	0.952	0.90	1.0
Helicopter	0.639	0.38	0.33	Laughing	0.973	0.60	0.78
Mouse click	0.65	0.68	0.70	Insects (flying)	0.987	0.55	0.78
Train	0.667	0.78	0.88	Crying baby	0.987	0.90	0.95
Airplane	0.679	0.45	0.62	Glass breaking	0.987	0.90	0.95
Fireworks	0.68	0.72	0.85	Dog	1.0	0.82	0.90

Table 6.2: Accuracy ($\times 100\%$) comparison for 10 easiest and 10 hardest sounds in ESC-50

source domain to the target domain, and improve the learning of the target task.

The right table in Tab. 6.1 shows the performance of different representations proposed in this work. The row shows the different networks from which the representations were obtained, and columns show the two main types of representations we obtain, each corresponding to a layer in our network design. The networks can obviously be $\mathcal{N}_A^{wlat}$ trained on Audioset or the three adapted networks, $\mathcal{N}_T^{I}$, $\mathcal{N}_T^{II}$, $\mathcal{N}_T^{III}$. The features correspond to either the F1 or the F2 layer in our network architecture, and then the segment level representations from these layers can be mapped through $max()$ or $avg()$ as described before.

Note that, even off-the-shelf F1 representations using $\mathcal{N}_A^{wlat}$ is able to achieve very high accuracy. This shows that our weakly labeled network, $\mathcal{N}_A^{wlat}$ is can actually capture knowledge which can be directly exploited for other sound related tasks.

Task adaptive training gives further improvement. Adaptation updates the network more towards the target label space leading to more discriminative features. F1 features in general work better than F2 layer features. Without adaptation, the representations from the F2 layer does not work well. However, F1 representation, which performs well even without adaptation, captures more generic characteristics of sounds from the source domain and is well suited to train an even simple linear classifier. Adaptation leads to improvement for all features and both $\mathcal{N}_T^{I}$ and $\mathcal{N}_T^{II}$ leads to the best-performing accuracy of 83.5%. As far as the segment to recording mapping is concerned, $max()$ mapping does better in all cases.

Figure 6.3 shows accuracy on each sound class using F1 features ($max()$ mapping) obtained from $\mathcal{N}_T^{II}$. Readers are requested to visit this webpage [1] for visually more pleasing figure as well as for performance numbers in other cases.

In Figure 6.3, the accuracy is 100% for some of the sound events, *Pouring Water, Church Bells.* More than 15 sound classes have 90% or higher accuracies. The lowest accuracy is obtained for *Helicopter* sounds, 33%. From the confusion matrix, it is clear that *Helicopter* sounds are confused a lot with *Engine* and *Airplane* sounds. The percent confusion with *Engine* and *Airplane* stands at 17% and 15%, respectively. *Engine* and *Airplane* sounds are very similar to *Helicopter* sounds and it is expected that differentiating them will be hard. Human accuracy for this class is around 63.9% and the confusion with *Engine* and *Airplane* stand at 12.5% and 18% respectively. Some other classes with performances in the lower end are, *Washing Machine* (53%), *Wind* (55%).

To better understand, how our transfer learning method stack up against human performance and SoundNet, we show compare accuracies for 10 easiest and 10 hardest classes in Table 6.2. The ten easiest and hardest sound classes were selected based on the human classification accuracy.

[1] `http://www.cs.cmu.edu/%7Ealnu/TLWeak.htm`

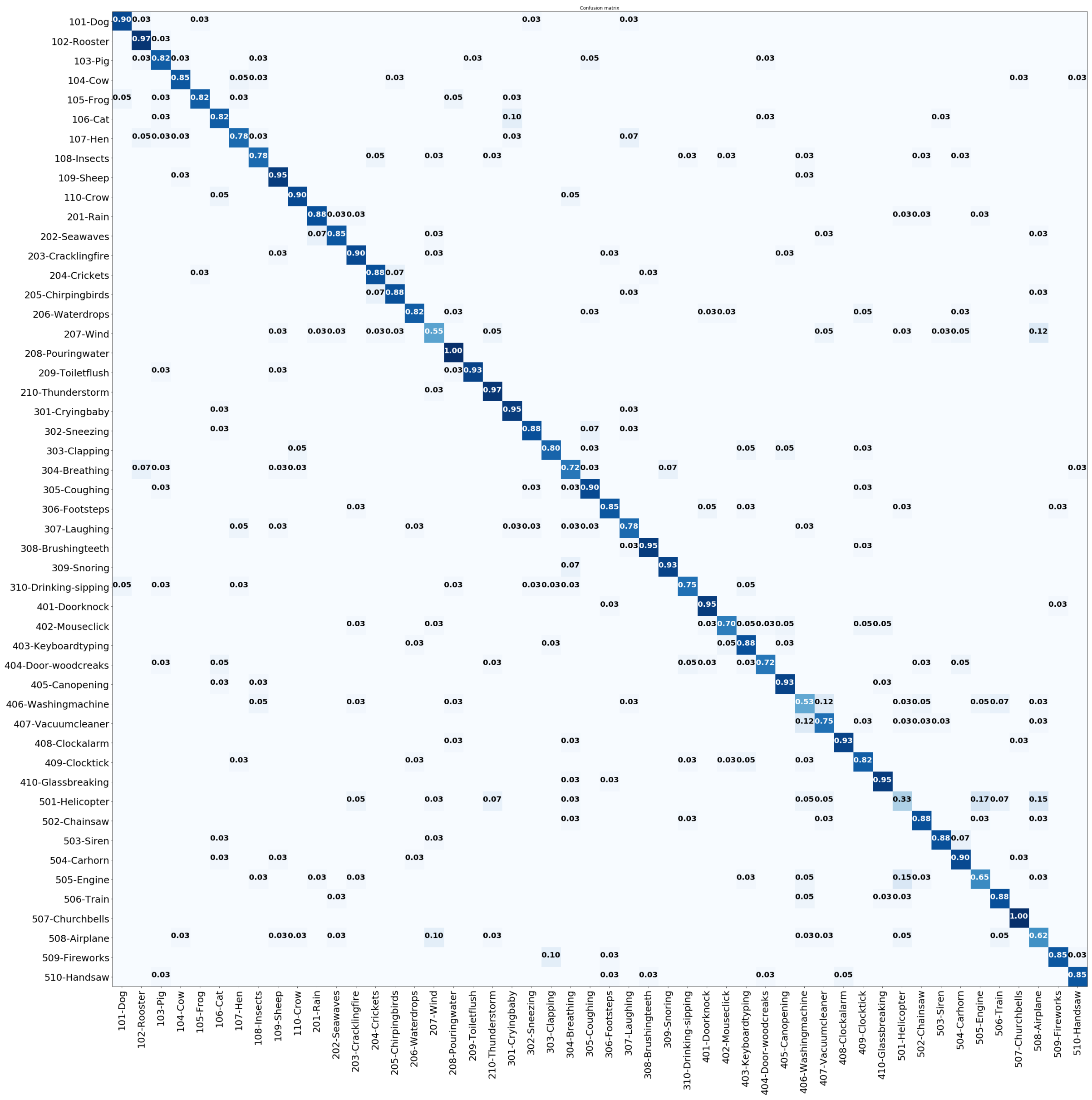


Figure 6.3: Transfer Learning: ESC-50 classwise accuracies

We can see that for sounds which are hard for humans to recognize, we can improve the accuracy by a considerable amount in several cases. For example, for *crackling fire* we improve the accuracy by 26%. Except for *Helicopter* our method is better than SoundNet in all cases. However, for sounds which are easier to recognize for humans, it is not easy to achieve accuracies which are very similar to human performance. The performance of our system is comparable to the human performance in some cases, whereas in some other cases it is considerably lower.

6.4.3 Results: Acoustic Scene Classification

Upper table in Table 6.3 compares accuracies for different acoustic scenes between the baseline and one of our proposed method. An absolute improvement of 4.1% over all 15 scenes is observed.

Scene	Baseline	$\mathcal{N}_S^{III}$ (F1, $max()$)	Scene	Baseline	$\mathcal{N}_S^{III}$ (F1, $max()$)
Beach	69.3	71.9	Library	50.4	73.6
Bus	79.6	82.4	Metro Station	94.7	80.2
Cafe	83.2	73.8	Office	98.6	85.1
Car	87.2	89.9	Park	13.9	46.9
City Center	85.5	93.3	Residential Area	77.7	63.9
Forest Path	81.0	97.4	Train	33.6	52.3
Grocery Store	65.0	84.6	Tram	85.4	84.0
Home	82.1	69.4	**Mean**	**72.5**	**76.6**

Network	**F1**		**F2**		Network	**F1**		**F2**	
	$max()$	$avg()$	$max()$	$avg()$		$max()$	$avg()$	$max()$	$avg()$
$\mathcal{N}_{\mathbf{S}}$	72.2	69.8	59.1	60.4	$\mathcal{N}_{\mathbf{T}}^{\mathbf{II}}$	75.5	73.0	73.8	73.9
$\mathcal{N}_{\mathbf{T}}^{\mathbf{I}}$	75.2	73.7	–	–	$\mathcal{N}_{\mathbf{T}}^{\mathbf{III}}$	76.6	73.7	72.5	73.3

Table 6.3: Transfer Learning: DCASE16 results
Upper: DCASE 2016 accuracy comparison with baseline ***Lower***: Accuracy comparison of different representations.

Categories	Human Performance	Proposed (Transfer Learning) $\mathcal{N}_T^{II}$, F1
Animals	86.4	86.0
Natural Soundscapes and Water	73.0	86.5
Human Non-Speech	89.5	85.0
Domestic	77.9	81.5
Exterior	79.9	78.3

Table 6.4: Transfer Learning: ESC-50 higher level categories

For certain scenes which are hard to classify such as *Park* and *Train*, absolute improvements of 33.0% and 18.7% respectively are obtained. Note that for this task, representations from task adapted networks perform much better compared to those obtained directly from $\mathcal{N}_A^{wlat}$. Since the task of acoustic scene classification is very different from the source task of sound event recognition, the adaptation of the network to the target task is more important in this case. Among the networks, the representation obtained from $\mathcal{N}_T^{III}$ gives best results, followed closely by $\mathcal{N}_T^{I}$ and $\mathcal{N}_T^{III}$. Representations from the F1 layer is again more suited for SVM classifier compared to F2 layer representations. Once again $max()$ mapping performs better compared to $avg()$.

Figure 6.4 shows the confusion matrix for our system. The lowest accuracy is obtained for *Park*, which is heavily confused with *residential area.* The other class with low accuracy is *train*, which is confused with acoustic scenes from *Bus. Car*, *City center* and *Forest Path* are acoustic scenes which have high accuracies. Readers can visit this webpage[1] for confusion matrices of other cases. Some additional analysis is provided in the next section.

6.4.4 Additional Analysis

In this section, we show some additional analysis on the two tasks to obtain some semantic understanding through the described transfer learning framework. We start by showing the

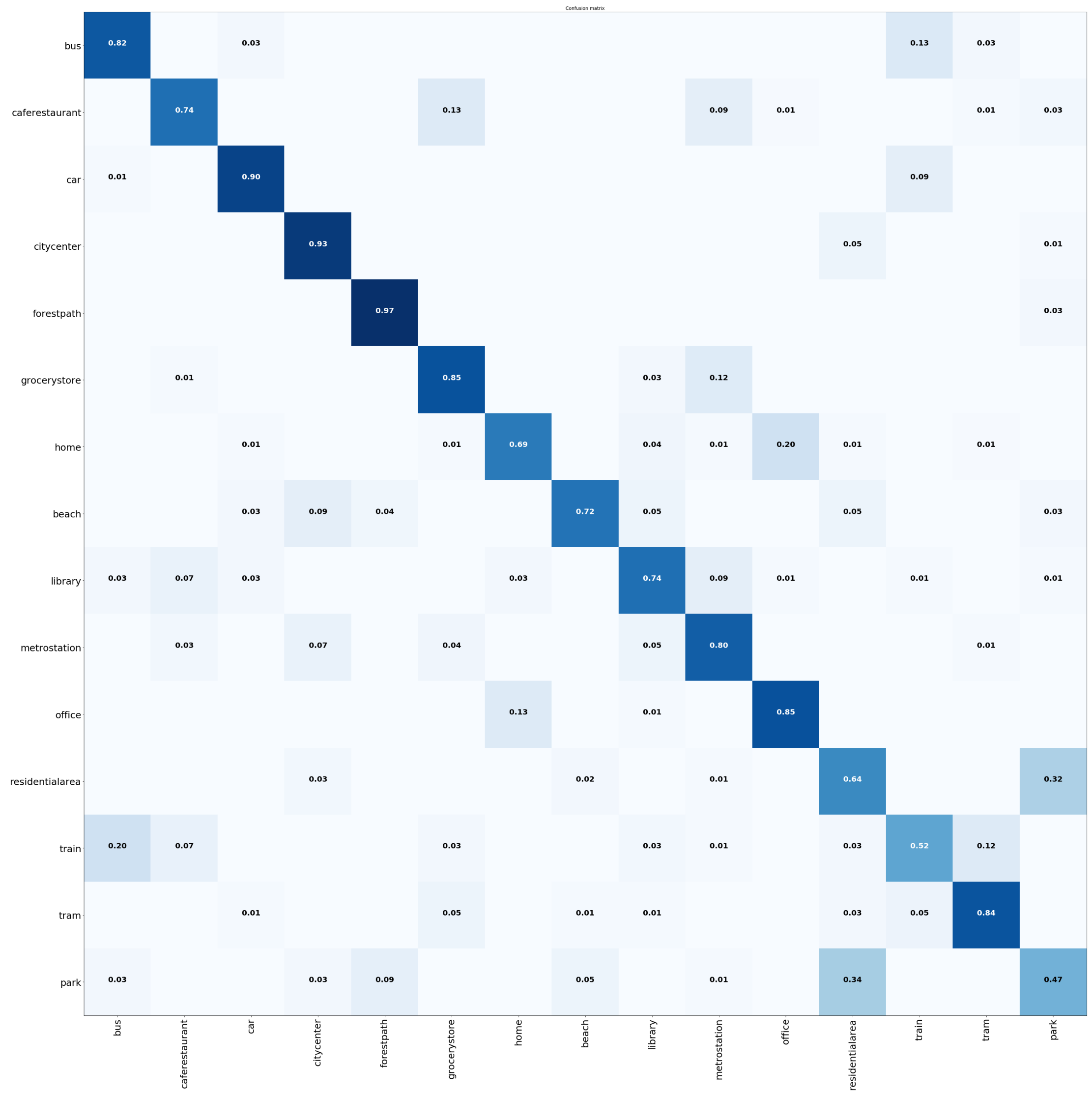


Figure 6.4: Transfer Learning: DCASE16 classwise accuracies

accuracies for the 5 broader categories in the ESC-50 dataset in Table 6.4. The accuracies are obtained by averaging the accuracies for sound events under each broad category. We see that *Animals* and *Human Non-Speech* sounds are easier to recognize and both humans and our system does well in these categories. Our transfer learning based system is able to achieve considerable improvement for *Natural Soundscapes and Water* sounds over human performance on the sounds of this class.

Visualizations

We now try to draw some semantic inferences from the proposed methods. Left panel in Fig 6.5 shows 2 dimensional t-SNE [Maaten and Hinton, 2008] embeddings for representations obtained for ESC-50. The embeddings are color coded for the 5 broad categories in the ESC-50 dataset, semantically higher level groups for sound events. One can note from the plot that these representations are capable of capturing higher level semantic information. *Vacuum Cleaner* in

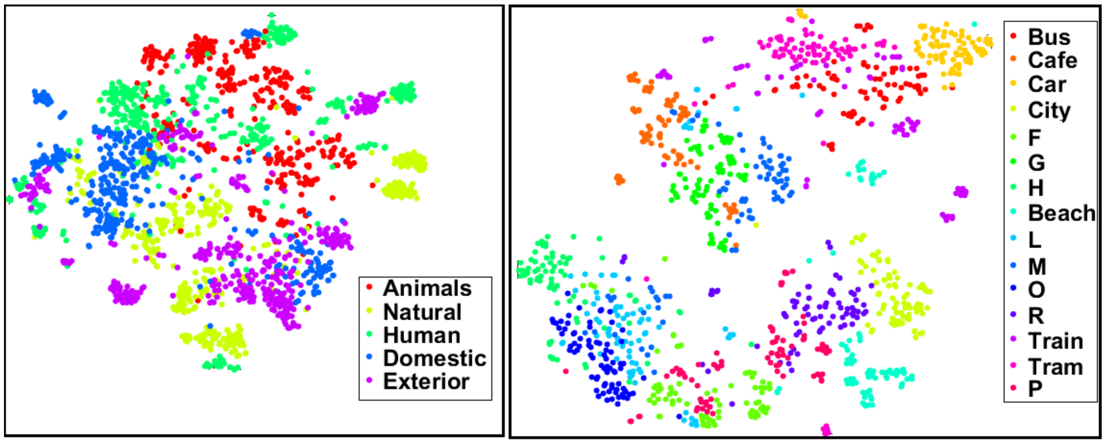


Figure 6.5: Visualizations of Learned Representations
Left: t-SNE visualizations for ESC-50. Color coded for 5 higher semantic categories ***Top Right***: t-SNE visualizations for DCASE 2016. First alphabet for some, e.g (F)orest.

Domestic closely resembles *Chainsaw*, *Engine* and *Handsaw* in *Exterior* category and its representations also lies closer to *Exterior* sounds (blue dots among purple). Similarly, visualization for 15 acoustic scenes is shown in the right panel in Fig. 6.5. These visualizations show that the representations we learn through our methods are capable of embedding the sounds in vector space.

Acoustic Scene - Sound Event Relations

Acoustic scenes are often composed of multiple sound events, and in fact, acoustic scenes can often be understood through sound events. In general, there are sound events which we expect to find in an environment or scene. For example, in *Park* we expect to find sound such as *Bird Song, Crow, Wind Noise* etc. These scenes - sound event relations can be helpful for not only acoustic scene classification task, but in general for building acoustic intelligence in machines. These semantic relations can be used for obtaining higher level semantic information which can further help in human-computer interaction and development of audio based context-aware systems. We looked at these relations previously in Chapter 2, where we tried to automatically mine scene - event relations through text. Here, we try to examine if our transfer learning strategies can help us in establishing meaningful scene - event relations for sounds through audio recordings and trained models.

Our source model has been trained on over 500 sound events. Hence, if our source model has actually learned some useful information, then we should be able to observe some meaningful relations between the acoustic scenes and sound events. We consider the network which has been adapted for the scene - classification task, $\mathcal{N}_T^{III}$.

Each neuron in F2 is essentially representing a sound event class, and the activations of these neurons can be used to understand scene-event relations. For each input of a given scene, we list the Top 5 maximally activated neurons (events) in the F2 layer. We then note the events which occurred frequently in these lists. We consider the top 10 highly frequent events. These highly active sound events for some of the scenes are shown in Table 6.5. We observe that several of these sound events are expected to occur in the corresponding acoustic scene. Hence, these scene-events relations are semantically meaningful. This shows that our network managed to

Scene	**Frequent Highly Activated Sound Events**
Cafe	Speech, Chuckle-Chortle, Snicker, Dishes, Television
City Center	Applause, Siren, Emergency Vehicle, Ambulance
Forest Path	Stream, Boat Water Vehicle, Squish, Clatter, Noise, Pour
Grocery Store	Shuffle, Singing, Speech, Music, Siren
Home	Speech, Finger Snapping, Scratch, Dishes, Baby Cry, Cutlery
Beach	Pour, Stream, Applause, Splash - Splatter, Gush
Library	Finger Snapping, Speech, Fart, Snort
Metro Station	Speech, Squish, Singing, Siren, Music
Office	Finger Snapping, Snort, Cutlery, Speech, Cutlery
Residential Area	Applause, Crow, Clatter, Siren
Park	Bird Song, Crow, Stream, Wind Noise, Stream

Table 6.5: Acoustic Scene - Sound Event Relations through transfer learning
Sound Events which are frequently among Top 5 maximally active events for a given scene.

transfer knowledge and learn relationships successfully.

6.5 Summary and Conclusions

In this chapter, we proposed methods for transfer learning in the domain of sounds. We presented methods which not only obtained state of the art performances in the event classification task but are also helpful in relating acoustic scenes to sound events. More specifically, we use a deep CNN network trained on a large scale weakly labeled dataset to transfer knowledge to target tasks. Our method does this by learning representations for audio recordings in the target task. A few prior works in transfer learning in the audio domain exist, primarily in speech and music domain. However, to the best of our knowledge, our work is the first work to show transfer learning for sound events through a large scale weakly labeled data. It is expected that sound related knowledge can be useful in other audio or video tasks. Our transfer learning can be helpful in even speech related tasks in audio or videos. For example, speaker identification and emotion recognition in consumer-generated videos. These videos are often recordings in environments where a number of other sounds are present, and the knowledge of sounds might be useful in improving the learning in these tasks. It can be useful for video concept detection or multimedia event detection in web videos. One recent work [Shah et al., 2018a] applied our method in activity recognition in short videos. We expect that this can be extended to several other tasks which can benefit from the knowledge of sounds.

Chapter 7

Evaluation on Large Scale: Limited Labeling Budget

Testing leads to failure, and failure leads to understanding.

-Burt Rutan

7.1 Introduction

The previous chapters on the automated understanding of sounds focused on the model learning aspect. First, we looked into methods to mine sound related knowledge from large text corpora, keeping the supervision required to a minimum and hence reducing the labeling effort. Then we described methods for sound events recognition using weakly labeled data, where once again the motivation was to reduce the labeling efforts. The central theme was to build learning methods for acoustic intelligence in machines through methods which reduce the labeling efforts and help scale the system. However, an equally important aspect of machine learning methods is *evaluation* of training algorithms. A learning method cannot be properly understood without a thorough evaluation. As we move towards large-scale learning, we need to evaluate on large test sets as well. This is necessary to understand the suitability of the applied machine learning algorithm in solving the concerned problem as well as to compare two different learning methods.

As pointed several times in the previous chapters, labeling data is a tedious and expensive procedure. Hence, an important focus area in the machine learning community is to develop methods which can reduce labeling efforts while maintaining or improving the system's performance. There have been concrete efforts to reduce the dependence on labeled data for training phase by developing weakly supervised, semi-supervised and unsupervised machine learning methods. However, labeling requirement in the evaluation of trained models is a problem which has not received requisite attention despite its immense significance. Irrespective of the method employed in the training phase, the *testing* phase always requires *labeled data* to compute classifier performance. Given that labeling is costly, the general tendency is to utilize most of the available labeling resources in obtaining training data. This leaves us wondering about the best strategy to *evaluate* classifier performance under limited labeling resources.

The answer to this problem is necessary as we move more and more towards big data machine learning; classifier evaluation on large datasets needs to be addressed along with classifier training. It is worth noting that this problem is completely different from cross-validation or any such method employed to measure the goodness of classifier during *training* phase. How the classifier is trained is immaterial to us here, the goal is to accurately estimate the performance of a given trained classifier on a test set with as little labeling effort as possible.

A trained classifier is almost always applied on a dataset which was never seen before and to estimate classifier performance on that dataset we require it to be labeled. This is also the case when a classifier is deployed into some real-world application where test data can be extensive and labeling even a small fraction of it might be very difficult. Consider, for example, a sound (or multimedia) event detection system deployed for event detection or content-based retrieval of videos on YouTube. We would like to get a good estimate of the performance of our trained system on the massive amount of YouTube videos. Even though we can run our system through millions of YouTube videos, we can label only a tiny fraction of them to compute the classifier performance.

Moreover, one might have to evaluate classifier as test data keeps coming in actively. Once again we may wonder if we should spend labeling resources on the new data or not. In other words, we would like to know if labeling the new data and evaluating the system on it will reveal some additional information about the classifier or not. All of these makes *testing* phase important where labeled data is needed to evaluate classifier. Yet, very little effort has been made to address the constraints posed by labeling costs during classifier evaluation phase.

One might argue that in certain cases, especially for web data we can automatically obtain labels and then evaluate the system using those labels. For example, we can try to obtain labels for YouTube videos through automated methods. However, as pointed out in the previous chapter these labels are expected to be noisy, and the estimate of the performance using these labels can be far off from the actual performance. Hence, even though in theory this approach sounds reasonable, it will require the development of methods which can automatically label web data without mistake. In [Badlani et al., 2018], we illustrate this aspect for audio event recognition and show that the evaluation of a system on a large scale can be a real challenge.

Our goal is to estimate the accuracy of a trained classifier on large test set [Kumar and Raj, 2018a]. The labeling resources are limited, meaning the maximum number of instances from the test data for which labels can be obtained is fixed and in general very small compared to the whole test set. Hence, the problem boils down to sampling instances for labeling such that the accuracy estimated on the sampled set is a close approximation of true accuracy. The simple strategy, of course, is *simple random sampling* – randomly drawing samples from the test set. This approach is, however, inefficient, and the variance of the accuracy estimated can be quite large. Hence, the fundamental question we are trying to answer is: can we do better than random sampling, where the test instances or samples to be labeled are selected from the whole test set?

The answer to the above solution lies in *Stratified Sampling*, which is a well-known concept in statistics Cochran [2007]. In stratified sampling, the idea is to divide the data into different strata and then sample a certain number of instances from each stratum. The statistical importance of this process lies in the fact that it usually leads to a reduction in the variance of the estimated variable. We will treat the problem from generic machine learning perspective and not in the specific context of sound events. We briefly discuss some related works in the next section.

7.1.1 Related Works

Some attempts have been made for unsupervised evaluation of multiple classifiers Jaffe et al. [2014], Parisi et al. [2014], Platanios et al. [2014], Donmez et al. [2010]. All of these works try to exploit outputs of multiple classifiers and use them to either rank classifiers, estimate classifier accuracies or combine them to obtain a more accurate metaclassifier. Although unsupervised evaluation sounds very appealing, these methods are feasible only if multiple classifiers are present. Moreover, assumptions such as the conditional independence of classifiers in most cases and/or knowledge of the marginal distribution of class labels in some cases need to be satisfied. In contrast, our focus is on the more general and practical case where the goal is to estimate the accuracy of a single classifier without the aid of any other classifier. The labeling resources are limited, meaning the maximum number of instances from the test data for which labels can be obtained is fixed and in general very small compared to the whole test set.

Very few works have looked into sampling techniques for classifier evaluation Bennett and Carvalho [2010],Druck and McCallum [2011],Katariya et al. [2012],Sawade et al. [2010]. Bennett and Carvalho [2010] and Druck and McCallum [2011] also used stratification for estimating classifier accuracy. Both of these works showed that stratified sampling, in general, leads to a better estimate of classifier accuracy for a fixed labeling budget. However, several important aspects are missing in these works, such as a theoretical study of the variance of the estimators, thorough investigation into stratification and allocation methods, the effect of the number of strata in stratification, and also evaluation of non-probabilistic classifiers. Other factors such as analysis of the dependence of the variance on the true accuracy are also missing. We formalize concepts related to stratified sampling based estimation and comprehensively analyze different aspects of it.

We establish variance relationships for accuracy estimators using both random sampling and stratified sampling. The variance relations not only allow us to analyze stratified sampling for accuracy estimation in theory but also allows to directly compare variances in different cases empirically, leading to a comprehensive understanding. We propose two strategies for practically implementing *Optimal* allocation in stratified sampling. We show that our proposed novel iterative method for optimal allocation offers several advantages over the non-iterative implementation of optimal allocation policy. The most important advantage is more precise estimation with lesser labeling cost. On the stratification front, we employ a panoply of stratification methods and analyze their effect on the variance of estimated accuracy. More specifically, we not only look into stratification methods well established in the statistical literature of stratified sampling but also consider clustering methods for stratification which are not directly related to stratified sampling.

Another related aspect studied here is the effect of the number of strata on the estimation of accuracy. We show the success of our proposed strategies on both probabilistic as well as non-probabilistic classifiers. The only difference between these two types of classifiers lies in the way we use classifier scores for stratification. We also empirically study the dependence of preciseness in accuracy estimation on the actual value of the true accuracy. Put simply, we look into whether stratified sampling is more effective for a highly accurate classifier or for a classifier with not so high accuracy.

In this work, we use only classifier outputs for stratification. This is not only simpler but also less restrictive compared to cases where the feature space of instances is used for stratification Katariya et al. [2012]. There are a number of cases where the feature space might be unknown due to privacy and intellectual property issues. For example, online text categorization or multimedia event detection may not give us the exact feature representations used for the inputs. These systems usually just give confidence or probability outputs of the classifier for the input. Medical data might bring in privacy issues in gaining knowledge of the feature space. Our method based only on classifier outputs is much more general and can be easily applied to any given classifier.

7.2 Accuracy Estimation

Let $\mathcal{D}$ be a dataset with N instances where i^{th} instance is represented by $\vec{x}_i$. We want to estimate the accuracy of a classifier C on dataset $\mathcal{D}$. The score output of the classifier on $\vec{x}_i$ is $C(\vec{x}_i)$ and the label predicted by C for $\vec{x}_i$ is $\hat{l}_i$. Let a_i be instance specific correctness measure such that $a_i = 1$ if $l_i = \hat{l}_i$, otherwise $a_i = 0$. Then the true accuracy, A, of the classifier over $\mathcal{D}$ can be expressed by Eq 7.1.

$$A = \frac{\sum_{i=1}^{N} a_i}{N} \tag{7.1}$$

Eq 7.1 is nothing but the population mean of variable a_i where $\mathcal{D}$ represents the whole population. To compute A, we need to know l_i for all $i = 1\ to\ N$. Our problem is to estimate the true accuracy A of C under constrained labeling resources, meaning only a small number of instances, n, can be labeled. Under these circumstances we expect to chose samples for labeling in an intelligent way such that the estimated accuracy is as precise as possible. Mathematically, we are interested in an unbiased estimator of A with minimum possible variance for a given n.

7.2.1 Simple Random Sampling Estimation

The trivial solution for the problem described in Section 7.2 is to randomly select n instances or samples and ask for labels for these instances. This process is called simple random sampling which we will refer to as random sampling at several places for convenience. Then the correctness measure a_i can be computed for these selected n instances, using which we can obtain an estimate of A. The estimate of the accuracy is the mean of a_i over the sampled set, $\hat{A}^r = \frac{\sum_{i=1}^{n} a_i}{n}$. $\hat{A}^r$ is an unbiased estimator of A and the variance of $\hat{A}^r$ is given by Eq 7.2.

$$V(\hat{A}^r) = \frac{S^2}{n},\ where\ S^2 = \frac{\sum\limits_{i=1}^{N}(a_i - A)^2}{N-1} \tag{7.2}$$

S^2 is the variance of a_i over $\mathcal{D}$. The variance formula above will include a factor $1 - \frac{n}{N}$ if sampling without replacement. For convenience we will assume sampling with replacement in our discussion and hence this term will not appear. The following lemma establishes the variance S^2 of a_i in terms of A.

Lemma 1. *S^2 for a_i is given by $S^2 = \frac{N}{N-1} \cdot A(1-A)$*

Proof. Expanding the sum in definition of S^2 in Eq 7.2

$$\begin{aligned} S^2 =& \frac{1}{N-1}\left(\sum_{i=1}^{N} a_i^2 + \sum_{i=1}^{N} A^2 - \sum_{i=1}^{N} 2Aa_i\right) \\ =& \frac{1}{N-1}(N \cdot A - N \cdot A^2) = \frac{N}{N-1} \cdot A(1-A) \end{aligned}$$

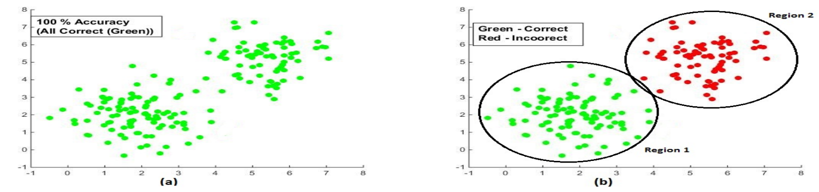


Figure 7.1: Two Cases for Illustration

The second line follows from the fact that $a_i \in \{0, 1\}$, hence, $\sum_{i=1}^{N} a_i^2 = \sum_{i=1}^{N} a_i$ and $\sum_{i=1}^{N} a_i = N \cdot A$.

Using Lemma 1 in Eq 7.2 establishes the following result for variance of $\hat{A}^r$.

Proposition 1. *The variance of random sampling based estimator of accuracy $\hat{A}^r$, is given by $V(\hat{A}^r) = \frac{N\,A(1-A)}{(N-1)\,n}$.*

Since A is unknown, we need an unbiased estimate of $V(\hat{A}^r)$ for empirical evaluation of variance. An unbiased estimate of S^2 can be obtained from a sample of size n by $s^2 = \frac{\sum_{i=1}^{n}(a_i - \hat{A}^r)^2}{n-1}$, Cochran [2007]. Clearly, a_i here corresponds to correctness measure for instances in the sampled set. Following the steps in Lemma 1, we can obtain

$$s^2 = \frac{n}{n-1} \cdot \hat{A}^r(1 - \hat{A}^r) \tag{7.3}$$

Proposition 2. *The unbiased estimate of variance of accuracy estimator $\hat{A}^r$, is given by $v(\hat{A}^r) = \frac{\hat{A}^r(1-\hat{A}^r)}{n-1}$.*

Proposition 2 follows from Eq 7.3. The estimated accuracy becomes more precise as n increases due to decrease in variance with n. The important question is, how can we achieve more precise estimation or in other words lower variance at a given n? To understand the answer to this question let us look at it a slightly different way. The question can be restated as how many instances should be labeled for a fairly good estimate of accuracy A ?

Consider Figure 7.1, where green points indicate instances for which C correctly predicts labels ($a_i = 1$). In Figure 7.1(a), the classifier is 100% accurate. In this case a single instance is sufficient to obtain the true accuracy of classifier. Now consider Figure 7.1(b), where the classifier is 100% accurate in *Set 1* and 100% incorrect in *Set 2*. Thus, labeling 1 instance from each set is sufficient to obtain true accuracy in that set and the overall accuracy is $A = \frac{1 \times N_1 + 0 \times N_2}{N}$. N_1 and N_2 are total number of points in sets 1 and 2 respectively. This leads us to the following general remark.

Remark 1. *If $\mathcal{D}$ can be divided into K "pure" sets, then true accuracy can be obtained by labeling K instances only, where 1 instance is taken from each set.*

"Pure" sets imply the classifier is either 100% *accurate* or 100% *inaccurate* in each set. In terms of the instance specific accuracy measure a_i, a "pure" set has either all $a_i = 1$ or all $a_i = 0$. This gives us the idea that if we can somehow divide the data into homogeneous sets then we can obtain a precise estimate of accuracy using very little labeling resources. The homogeneity is in terms of distribution of the values taken by a_i. The higher the homogeneity of a set the lesser the labeling resource we need for precise estimation of accuracy. Similarly, less homogeneous sets require more labeling resources. It turns out that this particular concept can be modeled in terms of a well known concept in statistics by the name of Stratified Sampling Cochran [2007].

7.3 Stratified Sampling Estimation

Let us assume that the instances have been stratified into K sets or strata. Let $\mathcal{D}_1, ..., \mathcal{D}_K$ be those strata. The stratification is such that $\mathcal{D}_1 \cup \mathcal{D}_2 \cup ... \cup \mathcal{D}_K = \mathcal{D}$ and $\mathcal{D}_j \cap \mathcal{D}_k = \emptyset, where,\ j \neq k, 1 \leq j \leq K, 1 \leq k \leq K$. All instances belong to only one stratum. The number of instances in strata $\mathcal{D}_k$ is N_k. Clearly, $\sum_{k=1}^{K} N_k = N$. The simplest form of stratified sampling is *stratified random sampling* in which samples are chosen randomly and uniformly from each stratum. If the labeling resource is fixed at n then n_k instances are randomly chosen from each stratum such that $\sum_{k=1}^{K} n_k = n$. In contrast to random sampling the estimate of accuracy by *stratified* random sampling is given by

$$\hat{A}^s = \sum_{k=1}^{K} \frac{N_k}{N} \hat{A}_k^r = \sum_{k=1}^{K} W_k \hat{A}_k^r \tag{7.4}$$

where $\hat{A}_k^r = \frac{1}{n_k} \sum_{i=1}^{n_k} a_i$ and $W_k = N_k/N$ are the estimated accuracy in k^{th} stratum and weight of k^{th} stratum respectively. The superscript r denotes that random sampling is used to select instances within each stratum. On taking expectation on both sides of Eq 7.4, it is straightforward to show that $\hat{A}^s$ is an *unbiased estimator* of A. Under the assumption that instances are sampled independently from each stratum, the variance of $\hat{A}^s$ is $V(\hat{A}^s) = \sum\limits_{k=1}^{K} W_k^2 V(\hat{A}_k^r)$. Since sampling within a stratum is random, applying Proposition 1 to each stratum leads to following result for the stratified sampling estimator.

Proposition 3. *The variance of stratified random sampling estimator of accuracy, $\hat{A}^s$, is given by*

$$V(\hat{A}^s) = \sum_{k=1}^{K} W_k^2 \frac{S_k^2}{n_k} = \sum_{k=1}^{K} W_k^2 \frac{N_k\, A_k(1 - A_k)}{(N_k - 1)\, n_k} \tag{7.5}$$

$S_k^2 = \frac{N_k\, A_k(1-A_k)}{(N_k-1)}$ is the variance of the a_i's in k^{th} stratum. A_k is the true accuracy in the k^{th} stratum and clearly, $\sum_{k=1}^{K} W_k A_k = A$.

Similarly, Proposition 2 can be applied for each stratum to obtain an unbiased estimator of $V(\hat{A}^s)$.

Proposition 4. *The unbiased estimate of variance of $\hat{A}^s$ is*

$$v(\hat{A}^s) = \sum_{k=1}^{K} W_k^2 \frac{s_k^2}{n_k} = \sum_{k=1}^{K} W_k^2 \frac{\hat{A}_k^r(1 - \hat{A}_k^r)}{(n_k - 1)} \tag{7.6}$$

The variance for stratified sampling is directly related to the two both the way samples are stratified and the allocation method applied on the stratified data. We discuss the allocation method first which deals with methods for defining n_k for each stratum. This allows a more systematic understanding of variance $V(\hat{A}^s)$ in different cases. We consider three different methods for distributing the available labeling resource n among the strata.

7.3.1 Proportional (PRO) Allocation

In proportional allocation the total labeling resource n is allocated proportional to the weight of the stratum. This implies $n_k = W_k \times n$. Substituting this value in Eq 7.5, the variance of $\hat{A}^s$

under proportional allocation, $V_{pro}(\hat{A}^s)$, is

$$V_{pro}(\hat{A}^s) = \frac{1}{n}\sum_{k=1}^{K} W_k S_k^2 = \frac{1}{n}\sum_{k=1}^{K} W_k \frac{N_k\, A_k(1-A_k)}{(N_k-1)} \tag{7.7}$$

The unbiased estimate of $V_{pro}(\hat{A}^s)$ can be similarly obtained. Once the process of stratification has been done, stratified random sampling with proportional allocation is fairly easy to implement. We compute n_k and then sample and label n_k instances from k^{th} stratum to obtain an estimate of accuracy A_k.

7.3.2 Equal (EQU) Allocation

In Equal allocation the labeling resource is allocated equally among all strata. This implies $n_k = n/K$. Equal allocation is again straightforward to use for obtaining accuracy estimate. Under equal allocation the variance of estimator $\hat{A}^s$ is

$$V_{equ}(\hat{A}^s) = \frac{K}{n}\sum_{k=1}^{K} W_k^2 S_k^2 = \frac{K}{n}\sum_{k=1}^{K} W_k^2 \frac{N_k\, A_k(1-A_k)}{(N_k-1)} \tag{7.8}$$

7.3.3 Optimal (OPT) Allocation

Optimal allocation tries to obtain the most precise estimate of accuracy using stratified sampling, for a fixed labeling resource n. The goal is to minimize the variance in the estimation process. Optimal allocation factors in both the stratum size and variance within stratum for allocating resources. In this case the labeling resource allocated to each stratum is given by

$$n_k = n\frac{W_k S_k}{\sum_{k=1}^{K} W_k S_k} \tag{7.9}$$

Using this value in Eq 7.5 the variance of $\hat{A}^s$ comes out as,

$$V_{opt}(\hat{A}^s) = \frac{\left(\sum\limits_{k=1}^{K} W_k S_k\right)^2}{n} = \frac{\left[\sum\limits_{k=1}^{K} W_k \left(\frac{N_k\, A_k(1-A_k)}{(N_k-1)}\right)^{\frac{1}{2}}\right]^2}{n} \tag{7.10}$$

Thus, a larger stratum or a stratum with higher variance of a_i or both is expected to receive more labeling resource compared to other strata. This variance based allocation is directly related to our discussion at the end of Sec 7.2.1. We remarked that a stratum which is homogeneous in terms of accuracy and hence a low variance stratum requires very few samples for precise estimation of accuracy in that stratum and vice versa. Thus, the intuitive and mathematical explanation are completely in sync with each other.

However, practical implementation of optimal allocation is not as straightforward as the previous two allocation methods. The true accuracies A_k's and hence S_k^2 are unknown implying we cannot directly obtain values of n_k. We propose two methods for practical implementation of optimal allocation policy.

In the first method, we try to obtain an initial estimate of all A_k by spending some labeling resources in each stratum. This leads us to an algorithm that we refer to as *OPT-A1*. The OPT-A1 method is shown in Algorithm 2. In the first step n_{ini} instances are chosen randomly

Algorithm 2 OPT-A1 Allocation

```
1: procedure OPT-A1($\mathcal{D}_1, ..., \mathcal{D}_k$,$n_{ini}$)
2:     Randomly Select and Label $n_{ini}$ instances from each stratum
3:     Estimate $A_k$ and then $S_k^2$ for each strata (applying Eq 7.3 for $k^{th}$ stratum)
4:     $n_{rem} = n - (K * n_{ini})$
5:     Allocate $n_{rem}$ among strata using the estimated $S_k^2$ in Eq 7.9
6:     Randomly sample again from each stratum according to above allocation
7:     Update estimates of $A_k$ and $S_k^2$ for all $k$
8: end procedure
```

Algorithm 3 OPT-A2 Allocation

```
1: procedure OPT-A2($\mathcal{D}_1, ..., \mathcal{D}_k$,$n_{ini}$,$n_{step}$)
2:     Randomly Select and Label $n_{ini}$ instances from each stratum
3:     Estimate $A_k$ and $S_k^2$ for each strata
4:     $n_{rem} = n - (K * n_{ini})$
5:     while $n_{rem} > 0$ do
6:         $n_{curr} = min(n_{step}, n_{rem})$
7:         Allocate $n_{curr}$ among strata using current estimate of $S_k^2$ in Eq 7.9
8:         Select and label new instances from each stratum according to allocation of $n_{curr}$ in
           previous step
9:         Update estimates of $A_k$ and $S_k^2$ for all $k$
10:        $n_{rem} = n_{rem} - n_{curr}$
11:    end while
12: end procedure
```

from each stratum for labeling. Then, an unbiased estimate of S_k^2 is obtained by using Eq 7.3 for k^{th} stratum. In the last step, these unbiased estimates are then used to allocate rest of the labeling resource $(n - K * n_{ini})$ according to optimal allocation policy given by Eq 7.9. Then, we sample again from each stratum according to the amount of allocated labeling resources and then update estimates of A_k.

In theory, optimal allocation gives us the minimum possible variance in accuracy estimation. However, allocation of n according to OPT-A1 depends heavily on initial estimates of S_k^2 in each stratum. If n_{ini} is small we might not able to get a good estimate of S_k^2 which might result in an allocation far from true optimal allocation policy. On the other hand, if n_{ini} is large we essentially end up spending a large proportion of the labeling resource in a uniform fashion which is same as equal allocation. This would reduce the gain in preciseness or reduction in variance we expect to achieve using optimal allocation policy. The optimal allocation in this case comes into picture for a very small portion $(n - K * n_{ini})$ of total labeling resource.

Practically, it leaves us wondering about value of parameter n_{ini}. To address this problem we propose another novel method for optimal allocation called OPT-A2. OPT-A2 is an iterative form of OPT-A1. The steps for OPT-A2 are described in Algorithm 3. In OPT-A2 n_{ini} is a small reasonable value. However, instead of allocating the remaining labeling resource in the next step we adopt an adaptive formalism. In this adaptive formalism step we allocate a fixed n_{step} labeling resource among the strata in each step. This is followed by an update in estimate of A_k and S_k^2. The process is repeated till we exhaust our labeling budget. We later show that results for OPT-A2 are not only superior compared to OPT-A1 but also removes concerns regarding the right value of n_{ini}. We show that any small reasonable values of n_{ini} and n_{step} works well.

7.3.4 Comparison of Variances

In this Section we study the variance, $V(\hat{A}^s)$ of stratified accuracy estimate $\hat{A}^s$ in different cases. The first question that needs to answered is whether stratified variance $V(\hat{A}^s)$ is always lower than random sampling variance $V(\hat{A}^r)$ for a fixed n or not. The answer depends on the sizes of strata N_k. We consider two cases; one in which all $1/N_k$ are small compared to 1 and other in which it is not.

Case 1: $1/N_k$ negligible compared to 1

This is the case we are expected to encounter in general for classifier evaluation and hence will be discussed in details. In this case, it can be easily established that, $V(\hat{A}^r) \geq V_{pro}(\hat{A}^s) \geq V_{pro}(\hat{A}^s)$ Cochran [2007]. For equal allocation no such theoretical guarantee can be established. We establish specific results below and compare variances of accuracy estimators for different cases. When needed, the assumption of $1/N_k << 1$ will be made.

First, we consider the cases of $V(\hat{A}^r)$ and $V_{pro}(\hat{A}^s)$. If $1/N_k << 1$, then so is $1/N << 1$. Hence, $N_k/(N_k - 1)$ and $N/(N - 1)$ is almost 1. Under this assumption the difference between $V(\hat{A}^r)$ and $V_{pro}(\hat{A}^s)$ is

$$V(\hat{A}^r) - V_{pro}(\hat{A}^s) = \frac{1}{n}[A(1-A) - \sum_{k=1}^{K} W_k A_k (1 - A_k)] \tag{7.11}$$

$$= \frac{1}{n}[\sum_{k=1}^{K} W_k A_k^2 - A^2] = \frac{1}{n}\sum_{k=1}^{K} W_k (A_k - A)^2 \tag{7.12}$$

The second line uses the fact that $A = \sum W_k A_k$ and $\sum W_k = 1$. Eq 7.12 implies that if the stratification is such that the accuracy of the strata are very different from each other, then the difference between $V(\hat{A}^r)$ and $V_{pro}(\hat{A}^s)$ is higher. This suggests that stratification which results in higher variance of A_k will lead to higher reduction in the variance of accuracy estimator. A special case is when A_k is same for all k. Then $A_k = A$ and in this case proportional allocation in stratified sampling will result in the same variance of estimated accuracy as simple random sampling. This implies that under this condition stratified sampling under proportional allocation is ineffective in improving the preciseness of accuracy estimation.

For stratified sampling, $V_{opt}(\hat{A}^s)$ by definition is the minimum possible variance of $\hat{A}^s$ for a fixed n. At best we can expect $V_{pro}(\hat{A}^s)$ and $V_{equ}(\hat{A}^s)$ to attain $V_{opt}(\hat{A}^s)$. Consider the difference between $V_{pro}(\hat{A}^s)$ and $V_{opt}(\hat{A}^s)$.

$$V_{pro}(\hat{A}^s) - V_{opt}(\hat{A}^s) = \frac{1}{n}\left[\sum_{k=1}^{K} W_k S_k^2 - (\sum_{k=1}^{K} W_k S_k)^2\right]$$

$$= \frac{1}{n}[\sum_{k=1}^{K} W_k S_k^2 - S_M^2] = \frac{1}{n}\sum_{k=1}^{K} W_k (S_k - S_M)^2 \tag{7.13}$$

In the second step (Eq 7.13), $S_M = \sum_{k=1}^{K} W_k S_k$ is the weighted mean of the S_k's. The second equality in Eq 7.13 uses the definition of S_M and the fact that $\sum_{k=1}^{K} W_k = 1$.

From Eq 7.13 it is straightforward to infer that $V_{pro}(\hat{A}^s)$ and $V_{opt}(\hat{A}^s)$ are equal *if and only if* $S_k = S_M$. This basically implies that if stratification of $\mathcal{D}$ is such that S_k is constant for all k then the variance of the stratified accuracy estimator under proportional and optimal allocation are equal. Thus, proportional allocation is optimal in the sense of variance.

Following the assumption of $1/N_k << 1$, $S_k = A_k(1-A_k)$. Let us assume that $S_k = S_c$ for all k, where S_c is some constant value. $S_k = S_c = A_k(1-A_k)$ implies for a given k, the value of A_k is one of the roots of the quadratic equation $y^2 - y + S_c$. If all A_k take the same root value, then from previous discussion we know $V(\hat{A}^r) = V_{pro}(\hat{A}^s)$. Constant S_k also means $V_{pro}(\hat{A}^s) = V_{opt}(\hat{A}^s)$. Hence, $V_{pro}(\hat{A}^s) = V_{opt}(\hat{A}^s) = V(\hat{A}^r)$. This leads us to the following remark.

Remark 2. *A stratification of $\mathcal{D}$ such that A_k is same for all k is the worst case stratification where the minimum possible variance of stratified estimator $\hat{A}^s$ is same as variance of random sampling accuracy estimator $\hat{A}^r$.*

Thus, even though $S_k = Constant$ will lead to simpler proportional allocation achieving minimum possible variance, it is not a very favorable situation when compared to random sampling. We might end up in the situation of Remark 2. Even if all A_k do not take same value, $S_k = S_c$ for all k implies the variance of A_k will not be very high. Hence, under this condition the minimum possible variance $V_{opt}(\hat{A}^s) = V_{pro}(\hat{A}^s)$ for stratified sampling won't be significantly smaller than $V(\hat{A}^r)$.

Now, assume $W_k S_k = S_{wc}$ for all k, where S_{wc} is a fixed constant value. If $W_k S_k$ is constant then from Eq 7.8, $V_{equ}(\hat{A}^s) = \frac{K^2 S_{wc}^2}{n}$. Also from Eq 7.10, $V_{opt}(\hat{A}^s) = \frac{K^2 S_{wc}^2}{n}$. Hence, a stratification such that $W_k S_k$ is a constant implies equal allocation is as good as optimal allocation. Hence, if it can be ensured that $W_k S_k = Constant$, then the simpler equal allocation can substitute optimal allocation.

Practical implementation of proportional and equal allocation methods are much simpler compared to optimal allocation where we need OPT-A1 or OPT-A2. In this Section apart from providing a comparison of variances in different cases, we looked into conditions under which proportional or equal allocation can be used as a substitute for optimal allocation giving same variance of estimator. For proportional allocation it did not turned out to be highly desirable because large reduction in variance compared to simple random sampling cannot be expected.

Equal allocation seems to be a better option provided the condition of constant $W_k S_k$ is satisfied. However, this condition is important and we cannot blindly use equal allocation for any stratification of $\mathcal{D}$. This is due to the fact that unlike proportional and optimal it does not come with a theoretical guarantee that worst case variance will be same as simple random sampling. In fact in certain cases it can lead to a higher variance than simple random sampling. However, our empirical evaluation suggests that equal allocation works fairly well for a variety of stratification methods. Lastly, implementation of optimal allocation is not directly possible and it is possible that the empirical variance of optimal allocation becomes more than that of random sampling even if $1/N_k << 1$ is satisfied. However, using our proposed algorithms OPTA1 and OPTA2 it happens very rarely.

Case 2: $1/N_k$ not negligible compared to 1

In general, even for moderately sized dataset we are not expected to encounter this case. Hence, for simplicity we only briefly discuss this case and show that under this condition $V_{pro}(\hat{A}^s)$ and $V_{opt}(\hat{A}^s)$ need not always be less than $V(\hat{A}^r)$. Consider a specific case of stratification when all

A_k are equal. Hence, $A_k = A$ for all k. Now, the difference between $V(\hat{A}^r)$ and $V_{pro}(\hat{A}^s)$.

$$V(\hat{A}^r) - V_{pro}(\hat{A}^s) = \frac{NA(1-A)}{n(N-1)} - \sum_k W_k \frac{N_k A_k(1-A_k)}{n(N_k-1)} \tag{7.14}$$

Using the fact that $A_k = A$

$$\begin{aligned} V(\hat{A}^r) - V_{pro}(\hat{A}^s) &= \frac{A(1-A)}{n}[\frac{N}{N-1} - \sum_k \frac{N_k}{N}\frac{N_k}{N_k-1}] \\ &= \frac{A(1-A)}{n}\left[\sum_k \frac{N_k}{N-1} - \frac{N_k^2}{N(N_k-1)}\right] \\ &= \frac{A(1-A)}{n}\left[\sum_k - \frac{N-N_k}{N(N-1)(N_k-1)}\right] \end{aligned}$$

Thus $V(\hat{A}^r) - V_{pro}(\hat{A}^s) < 0$. Hence, proportional stratified sampling gives higher variance than simple random sampling when all $A_k = A$. It is also possible to show that when S_k^2 is constant then it can lead to $V_{opt}(\hat{A}^s) = V_{pro}(\hat{A}^s) > V(\hat{A}^r)$.

7.3.5 Stratification Methods

We now consider the other aspect of stratified sampling which is construction of strata. Let us denote the variable used for stratification by z and let $f(z)$ be the density distribution of z. $z_i,\ i = 1\ to\ N$ denotes the discrete values of stratification variable for instances in dataset $\mathcal{D}$. If the classifier outputs $C(\vec{x}_i)$ are probabilistic then we use $z_i = p(\hat{l}_i/\vec{x}_i)$, that is the stratification variable is the probability of the predicted class for $\vec{x}_i$. If the classifier scores are non-probabilistic and the predicted label is given by $\hat{l}_i = sign(C(\vec{x}_i))$, we use $z_i = |C(\vec{x}_i)|$, that is the magnitude of the classifier output. This particular stratification variable has been designed keeping in mind binary classifiers like support vector machines where scores of larger magnitude generally imply a greater level of confidence in the label assigned. These two approaches can be used as a general schema for extending the definition of the stratification variable for other types of classifier as well.

The optimum stratification (in the sense of minimum variance) usually depends on the allocation policy Sethi [1963]Dalenius [1950]Dalenius and Gurney [1951]. While relationships for optimum stratification for a given allocation method exist and can be solved by complicated iterative procedures, a large body of stratification literature consists of approximate methods for optimum stratification.

We employ several known stratification methods for stratifying $\mathcal{D}$ using z. We also introduce use of clustering and simpler rule based methods which are usually not found in stratification literature. To estimate the density distribution $f(z)$ of the stratification variable using z_i's, we use Kernel Density estimation methods Friedman et al. [2001] with Guassian kernels.

cum $\sqrt{\mathbf{f}}$ (SQRT): This method proposed in Dalenius and Hodges Jr [1959] is perhaps the most popular and widely used method for stratification. The method has been designed for optimum allocation policy. The simple rule is to divide the cumulative of $\sqrt{f(z)}$ into equal intervals. The points of stratification, $z_1^s < z_2^s < .. < z_{K-1}^s$, correspond to the boundary points corresponding to these intervals. The k^{th} stratum consists of the set of instances for which z lies between z^s_{k-1} and z^s_k. z^s_0 and z^s_K can be set as max and min of z.

cum $\mathbf{f^{\frac{1}{3}}}$ (CBRT): This method is same as the SQRT except that the cube root of $f(z)$ is used in place of square root Singh [1971]. The derivation of SQRT method makes an assumption that stratification and estimation variables are same which is usually not the case. CBRT was proposed keeping in mind that stratification variable (z) is in practice different from estimation variable (a) and a regression model was assumed in deriving this method. Thomsen [1976] argues in favor of CBRT if proportional allocation is to be used.

Weighted Mean(WTMN): In this method the key idea is to to make the weighted mean of the stratification variable constant Hansen et al. [1953]. It is much simpler compared to the previous 2 methods and was proposed earlier to the previous two methods.

All of the previous methods try to approximate optimum stratification. These methods (SQRT and CBRT) work well if the stratification variable and estimation variable are highly correlated Serfling [1968]Anderson et al.. In more generic settings such as ours, no such assumption can be made for the stratification and estimated variable. Hence, we propose to introduce other techniques as well, which while not tailor-made for stratified sampling, can nevertheless serve as a way for stratification.

Clustering Methods: Clustering is one of the simplest ways to group the data $\mathcal{D}$ into different strata. We use K-means(KM) and Gaussian Mixture Models (GMM) based clustering to construct strata using z.

Simple Score Based Partitioning: The stratification variable z is obtained from classifier scores and we propose two simple partitioning methods. The first one is called EQSZ (Equal Size) in which the instances in $\mathcal{D}$ are first sorted according to the stratification variable. Starting from the top, each stratum takes away an equal number N/K of instances. It is expected that variation of z within each strata will be small. We call the other method as EQWD (Equal Width). In this case the range of z for $\mathcal{D}$ $(r = max(z) - min(z))$ is divided into sub-ranges of equal width. The points of stratification are $z^s_k = min(z) + rk/K$, $k = 1$ to K. $z^s_0 = min(z)$ is used in this case.

7.4 Experiments and Results

The proposed methods are independent of the learning problem. Since there aren't any large scale dataset for audio event detection we currently evaluate the proposed method on text classification problems.

The variance of stratified sampling depends on three important factors, Allocation Method, Stratification Method and number of strata. We perform a comprehensive analysis of all of these factors. Each allocation method is applied on all 7 stratification methods. We vary the number of strata from 2 to 10 to study the effect of K. Overall, this results in large number of experiments and we try to present the most informative results for each case in the paper.

We use three different dataset in our study. The first one, which is smallest of the three is the *News20 binary* dataset. It is the 2 class form of the text classification UCI News20 dataset Keerthi and DeCoste [2005]. It consist of a total of around 20000 instances. We use 4000 randomly selected instances for training a logistic regression classifier and the rest are used as test set $\mathcal{D}$ for which the classifier accuracy needs to be estimated. The second one is the *epsilon* dataset from the Pascal Large Scale Challenge pas [2008]. It contains 0.5 *million* instances of which we use a randomly selected $50,000$ for training a linear SVM. The remaining 0.45 million instances are used as the test set $\mathcal{D}$. The third is the two-class form of the *rcv1* text categorization dataset which is the largest of the three datasets Lewis et al. [2004]. The test set $\mathcal{D}$ consists of around 0.7 *million* instances. A logistic regression classifier is trained on the training set. We use the LIBLINEAR Fan et al. [2008] package for training all classifiers. All data are available for download from the LIBLINEAR website. Experiments on the three datasets together contain sufficient variation to study different aspects of accuracy estimation.

We will quantify our results in two ways. The first is the ratio of the variance of the stratified accuracy estimator to a random sampling estimator at a given n, VR=$V(\hat{A}^s)/V(\hat{A}^r)$. Clearly, unbiased estimates of $V(\hat{A}^r)$ and $V(\hat{A}^s)$ are used to measure VR. Ideally VR should be less than 1; the lower it is the better it is. The second measure deals with absolute error (AE) percentage in estimating accuracy. Specifically, we look at the AE vs n plot and observe the amount of labeling resource required to achieve just 1% absolute error in accuracy estimates. We focus on % reduction if any in required n to achieve 1% error when using $\hat{A}^s$ in place of $\hat{A}^r$. All experiments are repeated for 3000 runs and the variance and error terms are means over these runs. Hence, we will use MVR and MAE to refer to mean variance ratio and mean absolute error respectively.

7.4.1 Proportional Allocation

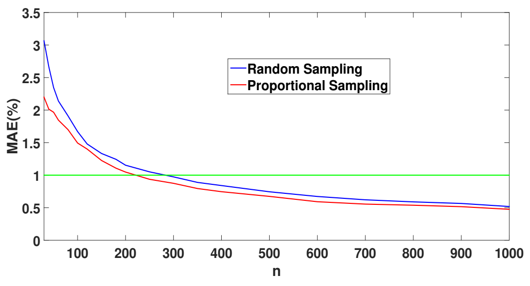
Figure 7.2a shows the MAE vs. n using EQWD stratification and $K = 10$ for the *rcv1* dataset. The number of labeled instances required to achieve a 1% error in accuracy estimation goes down from 284 in random sampling to 218. This is about **23**% reduction in labeling resources. Figure 7.2b shows the MVR values for each stratification method at different n. We can observe that EQWD is in general better compared to other methods leading to about $\mathbf{40-45}$**%** reduction in variance for some cases. WTMN is the worst showing only about 10% reduction in variances. The lower n values for which results are presented in Figure 7.2b and in subsequent figures, are in general more interesting cases. The difference between different methods are more visible and needs to be looked into carefully at lower n.

The results for the epsilon dataset are shown in Figure 7.2c and 7.2d. EQSZ stratification is used in Fig 7.2c. The reduction in labeling resources for 1% error in accuracy estimation is about 12.5%. This is can be attributed to the fact that for this dataset the maximum reduction in variance with proportional stratified sampling is in general less than 20%. EQSZ performs only marginally better than other methods such as SQRT and CBRT. Figures 7.2e and 7.2f show results for the *news20* dataset. About 16% reduction in labeling resource can be observed for KM stratification method shown in Figure 7.2e. Although on average across different n and K, KM is slightly better than other methods it does not always dominate and SQRT and CBRT work almost as well.

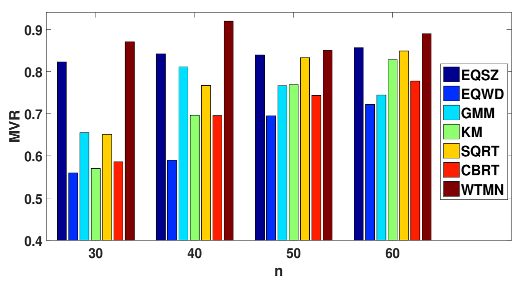
The variation of MVR with K for rcv1 and news20 is shown Figure 7.3a and 7.3b respectively. Increasing K does not necessarily leads to better results. However, the general trend is that once K is large enough major variation in MVR values cannot be expected. Hence, the parameter K is important but setting it to fixed reasonable value which will lead to good estimation of accuracy does not appear to be a hard problem. The trend is same for epsilon dataset and hence not shown here for brevity.

7.4.2 Equal Allocation

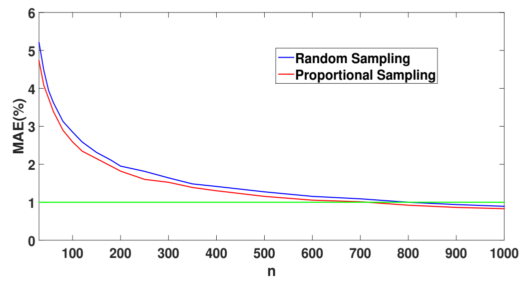
The results on *rcv1* dataset for Equal Allocation are shown is Figures 7.4a and 7.4b. Figure 7.4a uses KM based stratification. In this case n required for 1% error margin is reduced by



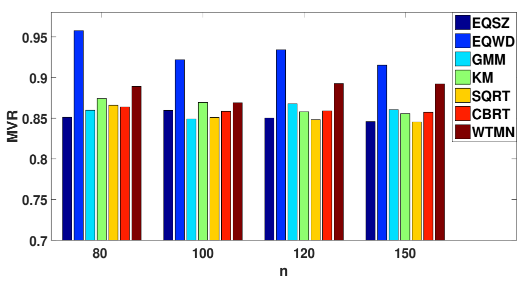
(a) rcv1, $K = 10$, EQWD



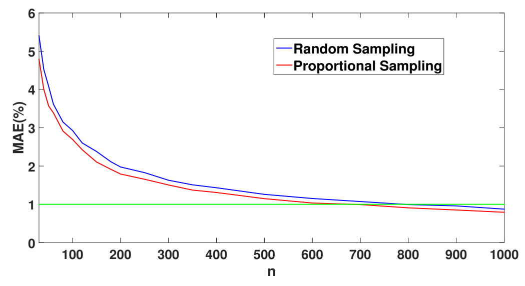
(b) rcv1, $K = 10$



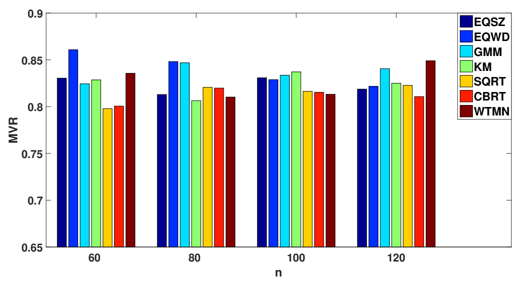
(c) epsilon, $K = 10$, EQSZ



(d) epsilon, $K = 10$



(e) news20, K=10, KM

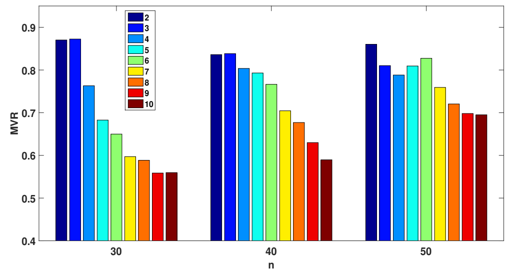


(f) news20, $K = 10$

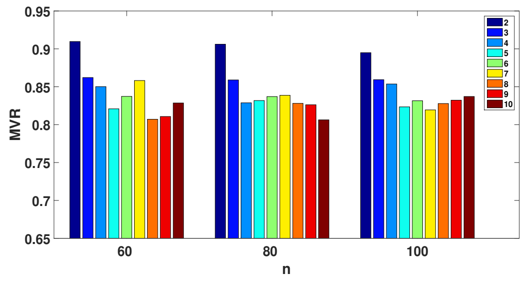
Figure 7.2: Proportional Stratified Sampling

a substantial amount which is close to **58.5**% (from 284 to 118). Fig 7.4b shows that all stratification barring EQSZ and WTMN gives similar reduction in variance which is in the range of **55** – **60**%. Thus for *rcv1* significant improvement in precision of accuracy estimates can be obtained using the Equal allocation policy.

Results on the *epsilon* dataset are shown in Figures 7.4c and 7.4d. Fig 7.4c used EQSZ for stratification, resulting in about 16% reduction in labeling resource for 1% error. However, the more important point to be noted is that barring EQSZ all other stratification methods leads to an increase in variance of accuracy estimates compared to random sampling. This illustrates that equal allocation based stratified sampling does not come with the assurance that it will lead to reduction in an estimator's variance. The results for *news20* dataset are shown in Fig 7.4e and 7.4f. In this case about 22% reduction in labeling resource can be observed and variation reduction lies in range of 18 – 23% in most cases. The variation of MVR with K for rcv1 and news20 is shown in Figure 7.5a and 7.5b. The trend is similar to what we observed as before.



(a) rcv1, EQWD, Proportional



(b) news20, KM, Proportional

Figure 7.3: MVR Variation With K for Proportional

7.4.3 Optimal Allocation

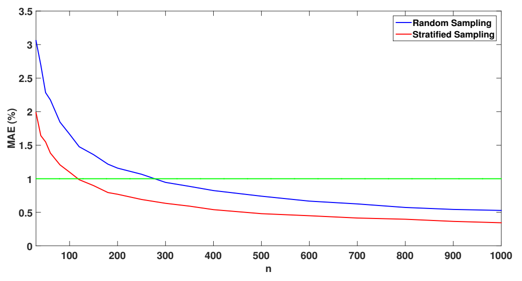
In the previous section we observed that for *rcv1* dataset Equal allocation resulted in a substantial reduction in the variance of the stratified accuracy estimator. The optimal allocation policy (OPT-A1 or OPT-A2) leads to further reduction in variance by only few percentage points ($4-6\%$ more) which does not translate into significant gain in terms of labeling resource reduction. It comes out to be slightly above 59%. However, we observe that for *epsilon* and *news20* optimal allocation actually results in substantial reduction in variance. We use $n_{ini} = 10$ for OPT-A1 algorithm. For OPT-A1 mid range K such as $K = 6$ *or* 7 are better in general, especially at lower n. K affects the number of samples ($n_{ini} * K$) used up for initial estimation of S_k. Mid range K are sufficient for obtaining good stratification and at the same time we are left with enough labeling resource which can be allocated optimally.

Figure 7.6b shows that EQSZ can results in over $\mathbf{30-35}\%$ reduction in variance compared to random sampling. In Figure 7.6a, n required for 1% error is reduced by 23% using OPT-A1 which is about 10% and 7% higher over proportional and equal allocation respectively. The worst stratification method is EQWD which corresponds to the practical problem we stated previously. Although, at higher n it does lead to reduction in variance it is still not as good as other methods for stratification. For *news20* OPT-A1 leads to reduction in n by about **27**% for 1% error which is higher than that for proportional and equal by 11% and 5% respectively. The variance is reduced by more than 35% for several cases which is substantially higher than other two allocation methods.

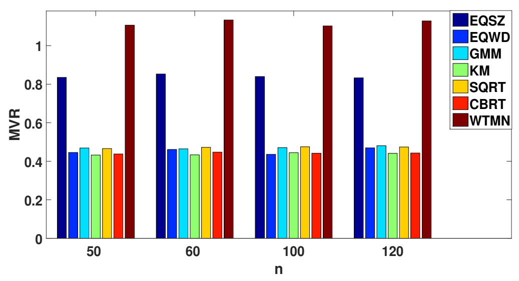
OPT-A1 vs OPT-A2

We mentioned previously that setting the right n_{ini} in OPT-A1 might present practical difficulties. This is illustrated in Figures 7.7a and 7.7b where we show MVR values for n_{ini} equal to 5,10 and 20. We first observe that for sufficiently high n, higher n_{ini} is better. This is expected as increasing n_{ini} results in better estimation of S_k and for large n we are still left with enough labeling resource which can be allocated in an optimal sense to help achieve lower variance. However, the problem occurs for lower n where we observe that MVR first reduces by increasing n_{ini} from 5 to 10 but then increases substantially when we increase it further to 20. Clearly, there is some optimal value between 5 to 20 which cannot be known a priori.

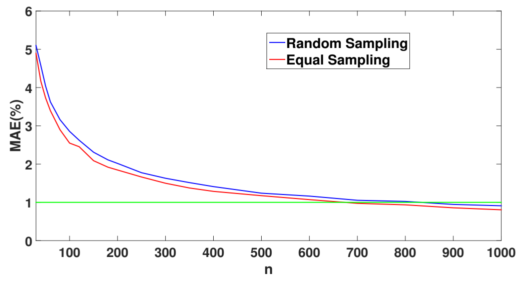
To get around the problems of OPT-A1, we proposed OPT-A2. Figures 7.8a and 7.8b shows the efficiency and benefits of OPT-A2. For both figures legend are in form $n_{ini} - n_{step}$. $n_{ini} - A1$ legends represent the corresponding MVR using OPT-A1. First, we observe that irrespective of the value of n_{ini} OPT-A2 results in reduction of MVR. In comparison to OPT-A1, OPT-A2 leads



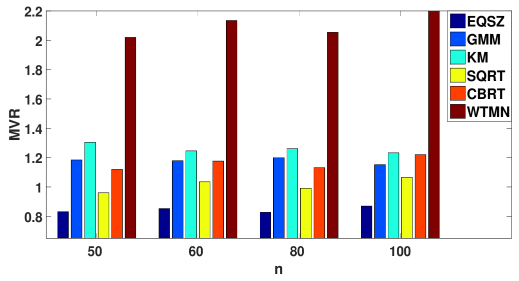
(a) rcv1, $K = 10$, KM



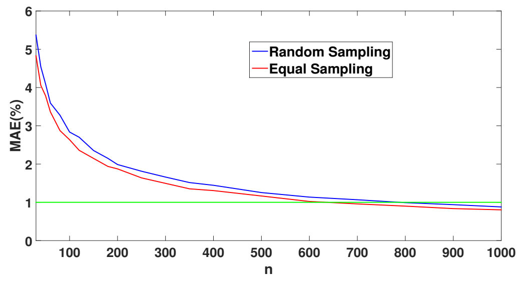
(b) rcv1, $K = 10$



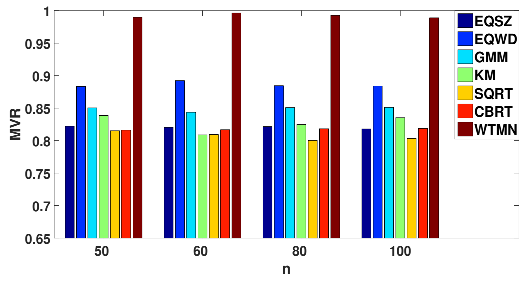
(c) epsilon, $K = 10$, EQSZ



(d) epsilon, $K = 10$



(e) news20, $K = 10$, SQRT



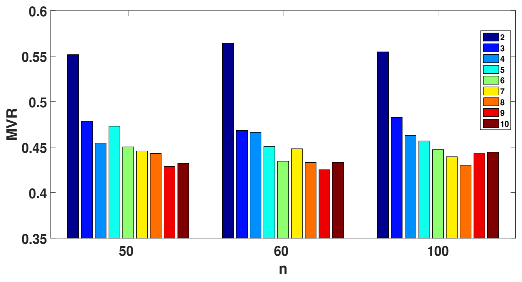
(f) news20, $K = 10$

Figure 7.4: Equal Stratified Sampling

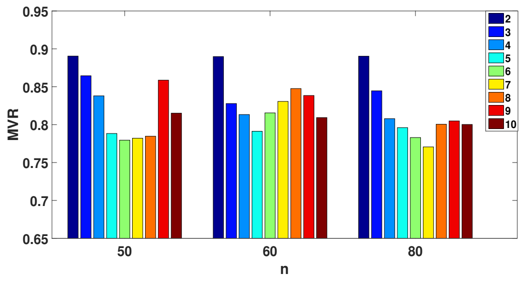
to a further reduction in variance of estimated accuracy by upto **18**% in certain cases. The range of reduction is $\mathbf{5-18}$%. This implies that for a given n, OPT-A2 will lead to a more precise estimation of true accuracy. Moreover, we observe that setting n_{ini} is no more critical; $n_{ini} = 5$ works as good as $n_{ini} = 10$. Even more convenient is the fact that n_{step} does not affect MVR in any major way which removes the role of any hyperparameter for OPT-A2. Hence, one can set n_{ini} to any small value such as 5 and any reasonable value of n_{step} such as 10 or 20 works fine. As we mentioned before for *rcv1* equal allocation OPT-A1 leads to only a small improvement in results over equal allocation. Using OPT-A2 on *rcv1* dataset leads to a further small improvements in results over *OPT-A1*, but not substantial. This again points toward existence of data specific bound.

7.4.4 Dependence on True Accuracy

It is expected that the value of true accuracy would have some effect on the MVR, which measures how well stratified sampling is doing compared to random sampling. Mainly, we would like to understand when can we expect MVR values to be low. In this section, we want to empirically

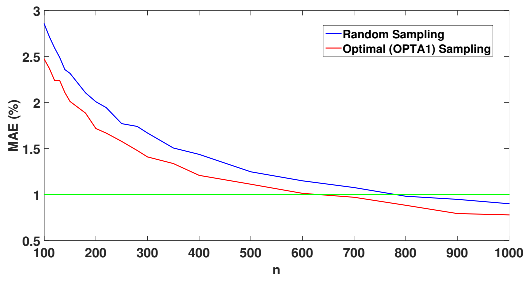


(a) rcv1,Equal

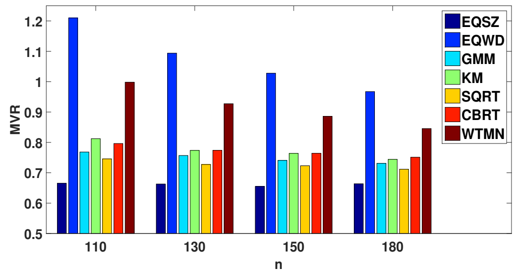


(b) news20,Equal

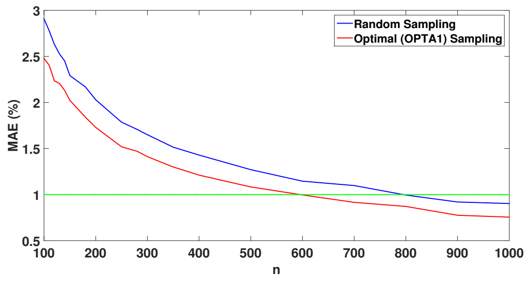
Figure 7.5: MVR Variation With K for Equal



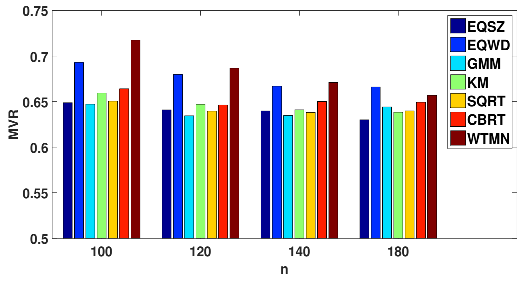
(a) epsilon,EQSZ,$K = 6$



(b) epsilon,$K = 6$



(c) news20,$K = 6$,SQRT

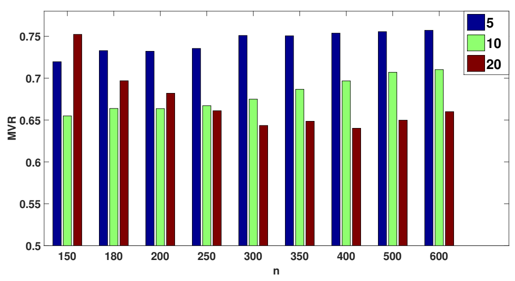


(d) news20,$K = 6$

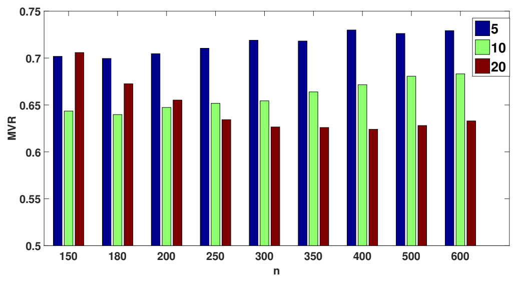
Figure 7.6: OPT-A1 Optimal Stratified Sampling

study the effect of actual value of true accuracy on the proposed accuracy estimation process. For all three datasets, the accuracy of logistic regression and linear SVM are close and hence any reasonable analysis cannot be made by comparing performance for these two types of classifiers.

We try to study this effect on the *epsilon* dataset by training 3 different classifiers (SVMs) with varying accuracies. The true accuracies of the classifiers are 88%(H), 77%(M) and 67%(L). The classifier accuracy has been reduced by reducing the amount of training data used. Obviously, the test data $\mathcal{D}$ on which these accuracies have been computed is same for all 3 classifiers. Now we try to estimate these accuracies for the 3 classifiers by sampling from $\mathcal{D}$ and we observe the MVR values for different n. Figure 7.9 show the results for three cases using OPT-A2 with $n_{ini} = 5$ and $n_{step} = 10$. We observe that MVR follows an inverse trend with classifier accuracy. Thus, the better the classifier the more effective stratified sampling is in reducing the variance of accuracy estimate. Similar trend for OPT-A1 also exist.

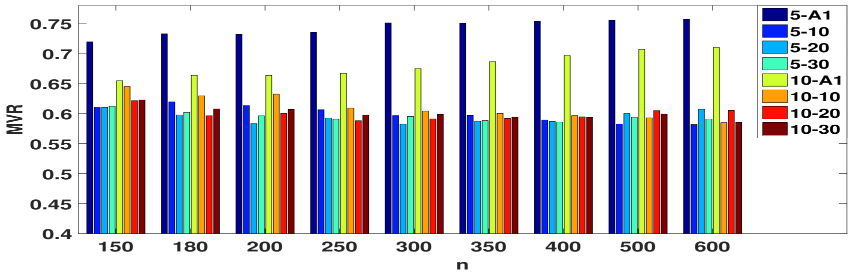


(a) epsilon, EQSZ, $K = 6$

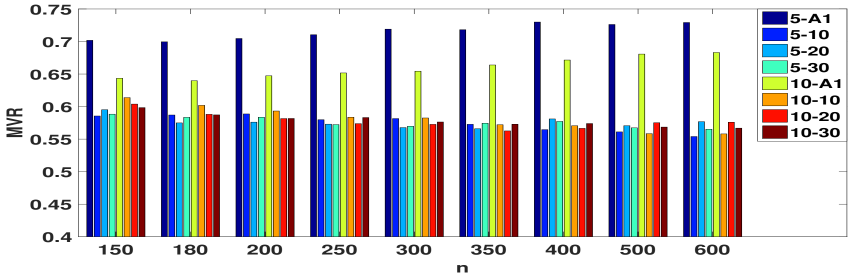


(b) news20, SQRT, $K = 6$

Figure 7.7: OPT-A1 Dependence on n_{ini}



(a) epsilon,EQSZ,$K = 6$



(b) news20,SQRT,$K = 6$

Figure 7.8: OPTA1 vs OPTA2, n_{ini} and n_{step}

7.5 Summary and Future Directions

In this chapter, we looked into efficient methods for estimating classifier performance on large test set under limited labeling budget. For the stratification methods we observed that algorithms such as SQRT and CBRT which are well established in stratified sampling literature works well for accuracy estimation as well. However, we also introduced clustering methods such as K-Means for stratification and found that these methods can also be employed for stratification and in several cases work better than all other methods. For the allocation methods clearly optimal allocation results in the lowest variance estimator. However, implementation of optimal allocation is not straightforward and is best done through OPT-A2 method.

Although, stratified sampling provides an efficient method for estimation of classifier accuracy there still remains a large body of open problems in this area. We observed that as classifier accuracy goes down the gain in preciseness using stratified sampling compared to random sampling estimator goes down. If the classifier accuracy is very low stratified estimation may be not be

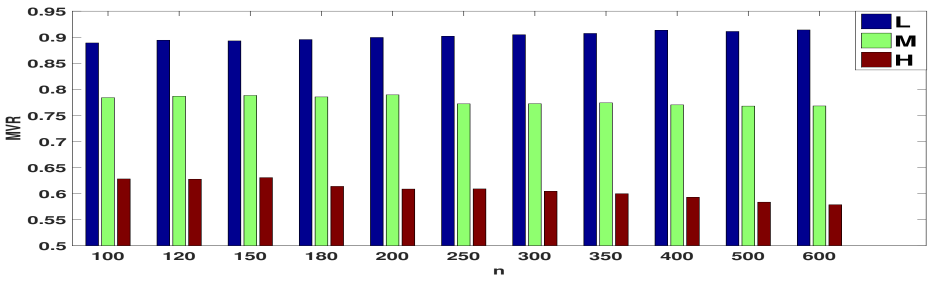


Figure 7.9: epsilon,EQSZ,$K = 6$,OPTA2

significantly better than random sampling estimator. This problem needs to be addressed. One way to address this estimation is to use the feature space of instances in the stratification process. Currently, we employed only classifier outputs for estimation. A stratification method employing both feature space and classifier output can give a relatively more homogeneous strata. This would result in lower variance of stratified estimator.

The stratified sampling estimator also needs to be extended for measuring performance measures in other practical cases. Extending it to multi-class accuracy estimation is an important task. Unlike the current case multi-class classification accuracy cannot be directly described by a binary measure. Hence, the overall approach needs to to properly adopted. Moreover, in a large number of cases we wish to measure other forms of performance metrics such as F1-score or average precision. It remains to be seen how these metrics can be precisely estimated under limited labeling budget scenario. We believe that future works will pay more attention to evaluation of systems for all machine learning tasks including sound event recognition.

Chapter 8

Some Applications

There is science and its applications, which are related to one another as the fruit is related to the tree that has borne it.

-Louis Pasteur

This chapter describes some applications oriented works done during the course of this dissertation. They deal with automated understanding of sounds, however, under specific contexts. We first show that knowledge of sounds can be used for the purposes of geotagging multimedia content. We then look into a system for query by example retrieval of sounds and then finally we will look into development of a large scale system which aims to learn about sounds in a never ending fashion.

8.1 Geotagging in Multimedia

Extracting information from multimedia recordings has received lot of attention due to the growing multimedia content on the web. A particularly interesting problem is the extraction of geo-locations or information relevant to geographical locations. This process of providing geographical identity information is usually termed as Geotagging [Luo et al., 2011] and is gaining importance due its role in several applications. It is useful not only in location based services and recommender systems [Bao et al., 2012] [Bao et al., 2015][Majid et al., 2013], but also in general cataloging, organization, search and retrieval of multimedia content on the web. Location specific information also allows a user to put his/her multimedia content into a social context, since it is human nature to associate with geographical identity of any material. A nice survey on different aspects of geotagging in multimedia can be found in [Choi et al., 2015, Luo et al., 2011].

Although, there are applications which allows users to add geographical information in their photos and videos, a larger portion of multimedia content on the web is without any geographical identity. In these cases geotags needs to be inferred from the multimedia content and the associated metadata. This problem of geotagging or location identification also features as the Placing Tasks in MediaEval [MediaEval, 2015] tasks. The goal of Placing Tasks [Choi et al., 2014] in MediaEval is to develop systems which can predict places in videos based on different modalities of multimedia such as images, audio, text etc. An important aspect of location prediction systems is the granularity at which location needs to be predicted. The Placing Task recognizes a wide range of location hierarchy, starting from neighborhoods and going upto continents. In this work we are particularly interested in obtaining city-level geographical tags which is clearly one of the most important level of location specification for any data. City-level information is widely used and well suited to location based services and recommender systems.

Most of the current works on geotagging focus on using visual/image component of multimedia and the associated text in the multimedia ([Trevisiol et al., 2013] [Luo et al., 2011][Song et al., 2012][Kelm et al., 2013] to cite a few). The audio component has largely been ignored and there is little work on predicting locations based on audio content of the multimedia. However, authors in [Choi et al., 2013] argue that there are cases where audio content might be extremely helpful in identifying location. For example, speech based cues can aid in recognizing location. Moreover, factors such as urban soundscapes and locations acoustic environment can also help in location identification. Very few works have looked into audio based location identification in multimedia recordings [Choi et al., 2015, Lei et al., 2012, Sevillano et al., 2012]. The approaches proposed in these works have been simplistic relying mainly on basic low level acoustic features such as Mel-Cepstra Coefficient (MFCC), Gammatone filter features directly for classification purposes. In other case audio-clip level features such *GMM - Supervectors* or *Bag Of Audio Words* (BoAW) histograms are first obtained and then classifiers are trained on these features [Choi et al., 2015].

We show that geotagging using *only audio* component of multimedia can be done by capturing the semantic content in the audio. Our primary assertion is that the semantic content of an audio recording in terms of different acoustic events can help in predicting locations. We argue that soundtracks of different cities are composed of a set of acoustic events. Hence, the composition of audio in terms of these acoustic events can be used to train machine learning algorithms for geotagging purposes. We start with a set of base acoustic events or sound classes and then apply methods based on matrix factorization to find the composition of soundtracks in terms of these acoustic events. Once the weights corresponding to each base sound class have been obtained, higher level features are built using these weights. Sound class specific kernels are fused to finally train Support Vector Machines for predicting location identification of the recording.

8.1.1 Audio Semantic Content based Geotagging

Audio based geotagging in multimedia can be performed by exploiting audio content in several ways. One can possibly try to use automatic speech recognition (ASR) to exploit the speech information present in audio. For example, speech might contain words or sentences which uniquely identifies a place, *I am near Eiffel Tower* clearly gives away the location as Paris, with high probability, irrespective of presence or absence of any other cues. Other details such as language used, mention of landmarks etc. in speech can also help in audio based geotagging.

In this work we take a more generic approach where we try to capture semantic content of audio through occurrence of different meaningful sound events and scenes in the recording. We argue that it should be possible to train machines to capture identity of a location by capturing the composition of audio recordings in terms of human recognizable sound events. This idea can be related to and is in fact backed by urban soundscapes works [Brown et al., 2011] [Payne et al., 2009]. Based on this idea of location identification through semantic content of audio, we try to answer two important questions. *First*, how to *mathematically* capture the composition of audio recordings and *Second*, how to use the information about semantic content of the recording for training classifiers which can predict identity of location. We provide our answers for each of these questions one by one.

Let $E = \{E_1, E_2, ..E_L\}$ be the set of base acoustic events or sound classes whose composition is to be captured in an audio recording. Let us assume that each of these sound classes can be characterized by a basis matrix M_l. For a given sound class E_l the column vectors of its basis matrix M_l essentially spans the space of sound class E_l. Mathematically, this span is in space of some acoustic feature (*e.g* MFCC) used to characterize audio recordings and over which the basis matrices have been learned. How we obtain M_l is discussed later. Any given soundtrack or audio recording is then decomposed with respect to the sound class E_l as

$$X \approx M_l W_l^T \tag{8.1}$$

where X is a $d \times n$ dimensional representation of the audio recording using acoustic features such as MFCC. For MFCCs, this implies each column of X is d dimensional mel-frequency cepstral coefficients and n is the total number of frames in the audio recording. The sound basis matrices M_l are $d \times k$ dimensional where k represents the number of basis vectors in M_l. In principle k can vary with each sound class, however, for sake of convenience we assume it is same for all E_l, for $l = 1\ to\ L$.

Equation 1 defines the relationship between the soundtrack and its composition in terms of sound classes. The weight matrix W_l captures how the sound class E_l is present in the recording. It is representative of the distribution of sound class E_l through out the duration of the recording. Hencer, obtaining W_l for each l provides us information about the structural composition of the audio in terms of sound classes in E. These W_l can then be used for differentiating locations. Thus, the first problem we need to address is to learn M_l for each E_l and then using it to compute W_l for any given recording.

8.1.2 Learning $\mathbf{M_l}$ and $\mathbf{W_l}$ using semi-NMF

Let us assume that for a given sound class E_l we have a collection of N_l audio recordings belonging to class E_l only. We parametrize each of these recordings through some acoustic features. In this work we use MFCC features augmented by delta and acceleration coefficients (denoted by MFCA) as basic acoustic features. These acoustic features are represented by $d \times n_i$ dimensional matrix $X^i_{E_l}$ for the i^{th} recording. d is dimensionality of acoustic features and each column represents acoustic features for a frame. The basic features of all recordings are collected into one single matrix $X_{E_l} = [X^i_{E_l}, ..X^N_{E_l}]$, to get a large collective sample of acoustic features for sound class E_l. Clearly, X_{E_l} has d rows and let T be the number of columns in this matrix.

To obtain the basis matrix M_l for E_l we employ matrix factorization techniques. More specifically, we use Non-Negative matrix factorization (NMF) like method proposed in [Ding et al., 2010]. [Ding et al., 2010] proposed two matrix factorization methods named *semi-NMF* and *convex-NMF* which are like NMF but do not require the matrix data to be non-negative. This is important in our case, since employing classical NMF [Lee and Seung, 2001] algorithms would require our basic acoustic feature to be non-negative. This can be highly restrictive given the challenging task at hand. Even though we employ MFCCs as acoustic features, our proposed general framework based on semi-NMF can be used with other features as well. Moreover, semi-NMF offers other interesting properties such as its *interpretation* in terms of *K-means clustering*. One of our higher level features is based on this interpretation of semi-NMF. convex-NMF did not yield desirable results and hence we do not discuss it in this paper.

semi-NMF considers factorization of a matrix, X_{E_l} as $X_{E_l} \approx M_l W^T$. For factorization number of basis vectors k in M_l is fixed to a value less than $min(d, T)$. semi-NMF does not impose any restriction on M_l, that is its element can have any sign. The weight matrix W on the other hand is restricted to be non-negative. The objective is to minimize $||X_{E_l} - M_l W^T||^2$. Assuming that

M_l and W have been initialized, M_l and W_l are updated iteratively in the following way. In each step of iteration,

$$\bullet \text{Fix } W, \text{update } M_l \; as, \; M_l = X_{E_l} W (W^T W)^{-1} \tag{8.2}$$

$$\bullet \text{Fix } M_l, \text{update } W, \; W_{rs} = W_{rs} \sqrt{\frac{(X_{E_l}^T M_l)_{rs}^{+} + [W(M_l^T M_l)^{-}]_{rs}}{(X_{E_l}^T M_l)_{rs}^{-} + [W(M_l^T M_l)^{+}]_{rs}}} \tag{8.3}$$

The process is iterated till error drops below certain tolerance. The $+$ and $-$ sign represents positive and negative parts of a matrix obtained as $Z_{rs}^{+} = (|Z_{rs}| + Z_{rs})/2$ and $Z_{rs}^{-} = (|Z_{rs}| - Z_{rs})/2$. Theoretical guarantees on convergence of semi-NMF and other interesting properties such as invariance with respect to scaling can be found in original paper. One interesting aspect of semi-NMF described by authors is its analysis in terms of K-means clustering algorithm. The objective function $||X - MW^T||^2$ can be related to K-Means objective function with M_l representing the k cluster centers. Hence, the basis matrix M_l also represents centers of clusters. We exploit this interpretation in the next phase of our approach. The initialization of M_l and W_l is done as per the procedure described in [Ding et al., 2010].

Once M_l have been learned for each E_l, we can easily obtain W_l for any given audio recording X by fixing M_l and then applying Eq 8.3 for X for several iterations. For a given X, W_l contains information about E_l in X. With K-Means interpretation of semi-NMF, the non-negative weight matrix W_l can be interpreted as containing soft assignment posteriors to each cluster for all frames in X.

8.1.3 Discriminative Learning using W_l

We treat the problem of location prediction as a retrieval problem where we want to retrieve most relevant recordings belonging to a certain location (city). Put more formally, we train binary classifiers for each location to retrieve the most relevant recordings belonging to the concerned location. Let us assume that we are concerned with a particular city C and the set $S = \{s_i, i = 1 \; to \; N\}$ is the set of available training audio recordings. The labels of the recordings are represented by $y_i \in \{-1, 1\}$ with $y_i = 1$ if s_i belongs to C, otherwise $y_i = -1$. X_i $(d \times n_i)$ denotes the MFCA representation of s_i. For each X_i weight composition matrices W_i^l are obtained with respect to all sound events E_l in E. W_i^l captures distribution of sound event E_l in X_i and we propose 2 histogram based representations to characterize this distribution.

Direct characterization of W_l as posterior

As we mentioned before semi-NMF can be interpreted in terms of K-means clustering. Based on this interpretation we proposea soft count histogram features using W_l. For a given E_l, the learned basis matrix M_l can be interpreted as matrix containing cluster centers. The weight matrix W_i^l $(n_i \times k)$ obtained for X_i using M_l can then be interpreted as posterior probabilities for each frame in X_i with respect to cluster centers in M_l. Hence, we first normalize each row of W_i^l to sum to 1, to convert them into probability space. Then, we obtain k dimensional histogram representation for X_i corresponding to M_l as

$$\vec{h}_i^l = \frac{1}{n_i} \sum_{t=1}^{n_i} \vec{w}_t \; ; \; \vec{w}_t = t^{th} \; row \; of \; W_i^l \tag{8.4}$$

This is done for all M_l and hence for each training recording we obtain a total of L, k dimensional histograms represented by $\vec{h}_i^l$.

GMM based characterization of W_l

We also propose another way of capturing distribution in W_l where we actually fit a mixture model to it. For a given sound class E_l, we first collect W_i^l for all X_i in training data. We then train a Gaussian Mixture Model $\mathcal{G}^l$ on the accumulated weight vectors. Let this GMM be $\mathcal{G}^l = \{\lambda_g, N(\vec{\mu}_g, \Sigma_g), g = 1 \ to \ G^l\}$, where λ_g^l, $\vec{\mu}_g^l$ and Σ_g^l are the mixture weight, mean and covariance parameters of the g^{th} Gaussian in $\mathcal{G}^l$. Once $\mathcal{G}^l$ has been obtained, for any W_i^l we compute probabilistic posterior assignment of weight vectors w_t in W_i^l according to Eq 8.5 ($Pr(g|\vec{w}_t)$). $\vec{w}_t$ are again the rows in W_i^l. These soft-assignments are added over all t to obtain the total mass of weight vectors belonging to the g^{th} Gaussian ($P(g)_i^l$, Eq 8.5). Normalization by n_i is done to remove the effect of the duration of recordings.

$$Pr(g|\vec{w}_t) = \frac{\lambda_g^l N(\vec{w}_t; \vec{\mu}_g^l, \Sigma_g^l)}{\sum\limits_{p=1}^{G} \lambda_p^l N(\vec{w}_t} \tag{8.5}$$

The final representation for W_i^l is $\vec{v}_i^l = [P(1)_i^l, ...P(G^l)_i^l]^T$. $\vec{v}_i^l$ is a G^l-dimensional feature representation for a given recording X_i with respect to E_l. The whole process is done for all E_l to obtain L different soft assignment histograms for a given X_i.

8.1.4 Kernel Fusion for Semantic Content based Prediction

$\vec{h}_i^l$ or $\vec{v}_i^l$ features captures acoustic events information for any X_i. We then use kernel fusion methods in Support Vector Machine (SVM) to finally train classifiers for geotagging purposes. We explain the method here in terms of $\vec{h}_i^l$, for $\vec{v}_i^l$ the steps followed are same.

For each l, we obtain separate kernels representation K_l using $\vec{h}_i^l$ for all X_i. Since exponential χ^2 kernel SVM are known to work well with histogram representations [Zhang et al., 2007] [Cao et al., 2011], we use kernels of the form $K_l(\vec{h}_i^l, \vec{h}_j^l) = exp(-D(\vec{h}_i^l, \vec{h}_j^l)/\gamma)$ where $D(\vec{h}_i^l, \vec{h}_j^l))$ is χ^2 distance between $\vec{h}_i^l$ and $\vec{h}_j^l$. γ is set as the average of all pair wise distance. Once we have all K_l, we use two simple kernel fusion methods;

- Average kernel fusion - The final kernel representation is given by, $K_S^h = \frac{1}{L}\sum_{l=1}^{L} K_l(;,;)$
- Product kernel fusion - In this case the final kernel representation is given by, $K_P^h = \frac{1}{L}\prod_{i=1}^{L} K_l(:,:)$.

Finally, K_S^h or K_P^h is used to train SVMs for prediction.

8.1.5 Experiments and Results

As stated before, our goal is to perform city - level geotagging in multimedia. Hence, we evaluate our proposed method on the dataset used in [Lei et al., 2012] which provides city level tags for flickr videos. The dataset contains contains a total of 1079 Flickr videos with 540 videos in the training set and 539 in the testing set. We work with only audio of each video and we will alternatively refer to these videos as audio recordings. The maximum duration of recordings is

90 seconds. The videos of the recording belong to 18 different cities with several cities having very few examples in training as well as testing set such as just 3 for Bankok or 5 for Beijing. We selected 10 cities for evaluation for which training as well as test set contains at least 11 examples. These 10 cities are *Berlin (B), Chicago (C), London (L), Los Angeles (LA), Paris (P), Rio (R), San Francisco (SF), Seoul(SE), Sydney (SY)* and *Tokyo (T).* As stated before the basic acoustic feature used are MFCC features augmented by delta and acceleration coefficients. 20 dimensional MFCCs are extracted for each audio recording over a window of 30 ms with 50% overlap. Hence, basic acoustic features for audio recordings are 60 dimensional and referred to as MFCA features.

We compare our proposed method with two methods, one based on GMM based bag of audio words (BoAW) and other based on GMM-supervectors. These are clip level feature representation built over MFCA acoustic features for each recording. The first step in this method is to train a background GMM $\mathcal{G}^{bs}$ with G^b components over MFCA features where each Gaussian represents an audio word. Then for each audio recording clip level histogram features are obtained using the GMM posteriors for each frame in the clip. The computation is similar to Eq 8.5; except that the process is done over MFCA features. These clip level representation are soft count bag of audio words representation. GMM-supervectors are obtained by adapating means of background GMM $\mathcal{G}^{bs}$ for a given using *maximum a posteriori* (MAP) adaptation [Campbell et al., 2006]. We will use $\vec{b}$ to denote these G^b dimensional bag of audio words features and $\vec{s}$ to denote the $G^b \times 60$ dimensional GMM - supervectors. Exponential χ^2 kernel SVMs are used with $\vec{b}$ features and linear SVMs are used with GMM - supervectors features. Exponential χ^2 kernels are usually represented as $K(x, y) = \exp^{-\gamma D(x,y)}$, where $D(x, y)$ is χ^2 distance between vetors x and y. Both of these kernels are known to work best for the corresponding features. All parameters such as γ and the slack parameter C in SVMs are selected by cross validation over the training set.

For our proposed method we need a set of sound classes E. Studies on *Urban soundscapes* have tried to categorize the urban acoustic environments [Brown et al., 2011] [Payne et al., 2009] [Schafer, 1993]. [Salamon et al., 2014] came up with a refined taxonomy of *urban sounds* and also created a dataset, *UrbanSounds8k*, for urban sound events. This dataset contains 8732 audio recordings spread over 10 different sound events from urban sound taxonomy. These sound events are car horn, children playing, dog barking, air conditioner noise, drilling, engine idling, gun shot, jackhammer, siren and street music. We use these 10 sound classes as our set E and then obtain the basis matrices M_l for each E_l using the examples of these sound events provided in the UrbanSounds8k dataset.

The number of basis vectors for all M_l is same and fixed to either 20 or 40. We present results for both cases. Finally, in the classifier training stage; SVMs are trained using the fused kernel K_S^h (or K_P^h, or K_S^v, or K_P^v) as described in Section 8.1.4. Here the slack parameter C in SVM formulation is set by performing 5 fold cross validation over the training set.

We formulate the geotagging problem as retrieval problem where the goal is to retrieve most relevant audios for a city. We use well known Average Precision (AP) as metric to measure performance for each city and Mean Average Precision (MAP) over all cities as the overall metric. Due to space constraints we are not able to show AP results in every case and will only present overall metric MAP.

Table 8.1 shows MAP results for BoAW and Supervector based methods (top 3 rows) and our proposed method (bottom 3 rows) using $\vec{h}^l$ features described in Section 8.1.3. For baseline method we experimented with 4 different component size G^b for GMM $\mathcal{G}^{bs}$. k represents the number of basis vectors in each M_l. K_S^h represents the average kernel fusion and K_P^h product

Table 8.1: MAP for different cases ($\vec{b}$, $\vec{s}$ and $\vec{h}^l$)

$G^b \rightarrow$	32	64	128	256
MAP ($\vec{b}$) $\rightarrow$	0.362	0.429	0.461	**0.478**
MAP ($\vec{s}$) $\rightarrow$	0.446	**0.491**	0.471	0.437

Kernel $\rightarrow$	Avg Ker. (K^h_S)		Prod. Ker (K^h_P)	
$k \rightarrow$	20	40	20	40
MAP $\rightarrow$	0.454	0.500	**0.520**	**0.563**

kernel fusion. First, we observe that our proposed method outperforms these state of art methods by a significant margin. For BoAW, $G^b = 256$ gives highest MAP of 0.478 but MAP saturates with increasing G^b and hence, any significant improvement in MAP by further increasing G^b is not expected. For supervectors $G^b = 64$ gives best result and MAP decreases on further increasing G^b. Our proposed method with $k = 40$ and product kernel fusion gives $\mathbf{0.563}$ MAP, an absolute improvement of $\mathbf{8.5}$% and $\mathbf{7.2}$% when compared to BoAW and supervectors based methods respectively. MAP in other cases for our proposed method are also in general better than best MAP using state of art methods. We also note that for $\vec{h}^l$ features, product kernel fusion of different sound class kernels performs better than average kernel fusion. Also, for $\vec{h}^l$, $k = 40$ is better than $k = 20$.

Table 8.2 shows results for our $\vec{v}^l$ features in Section 8.1.3 which uses GMM based characterization of composition matrices W_l. We experimented with 4 different values of GMM component size G^l. Once again we observe that overall this framework works gives superior performance. Once again MAP of $\mathbf{0.527}$ with $\vec{v}^l$ is over 3.6% higher in absolute terms when comapred to best MAP with supervectors.

This shows that the composition matrices W_l are actually capturing semantic information from the audio and these semantic information when combined helps in location identification. If we compare $\vec{v}^l$ and $\vec{h}^l$ methods then overall $\vec{h}^l$ seems to give better results. This is worth noting since it suggests that W_l on its own are extremely meaningful and sufficient. Another interesting observation is that for $\vec{v}^l$ average kernel fusion is better than product kernel fusion.

Figure 8.1 shows city wise results for all 4 methods methods. For each method the shown AP correspond to the case which results in best MAP for that method. This implies GMM component size in both BoAW and $\vec{v}^l$ is 256 that is $G^l = G^b = 256$; for $\vec{h}^l$ $k = 40$ and product kernel fusion; for $\vec{v}^l$ $k = 20$ and average kernel fusion. For supervector based method $G^b = 64$. For convenience, city names have been denoted by indices used in the beginning paragraph of this section. Figure 8.1 also shows MAP values in the extreme right. One can observe from Figure 8.1 that cities such as *Rio (R), San Francisco (SF), Seoul (SE)* are much easier to identify and all methods give over 0.60 AP. On the other hand *Sydney (SY)* is a much harder to geotag comapred to other cities. Once again our proposed method outperforms BoAW and supervector based methods for all cities except for Berlin (B).
We presented methods for geotagging in multimedia using its audio content. We showed that the semantic content of the audio captured in terms of different sound events which occur in the environment, can be used for location identification purposes. It is expected that larger the number of sound classes in E, the more distinguishing elements we can expect to obtain and the better it is for geotagging. Hence, it is desirable that any framework working under this idea should be scalable in terms of number of sounds in E. In our proposed framework the process of learning basis matrices M_l are independent of each other and can be easily parallelized. Similarly,

Table 8.2: MAP for different cases for $\vec{v}^l$

$G^l \downarrow$ — $k \rightarrow$	Avg Ker. (K_S^v)		Prod. Ker (K_P^v)	
	20	40	20	40
32	0.454	0.427	0.448	0.417
64	0.482	0.466	0.432	0.424
128	**0.510**	0.465	0.466	0.427
256	**0.527**	0.455	0.471	0.441

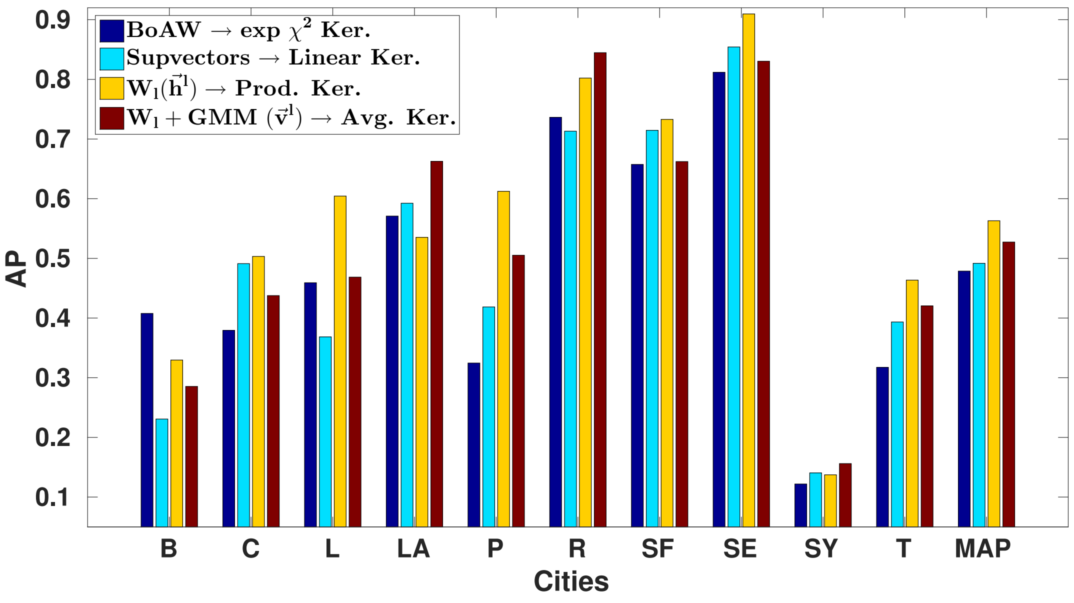


Figure 8.1: Average Precision for Cities (MAP in right extreme)

obtaining composition weight matrices W_i^l can also be computed in parallel for each E_l and so do the features $\vec{h}_i^l$ (or $\vec{v}_i^l$) and kernel matrices. Hence, our proposed is completely scalable in terms of number sound events in the set E. If required, one can also easily add any new sound class to an existing system if required. Moreover, our proposed framework can be applied on any acoustic feature.

Even with 10 sound events from urban sound taxonomy we obtained reasonably good performance. Our framework outperformed supervector and bag of audio word based methods by a considerable margin. Currently, we used simple kernel fusion methods to combine event specific kernels. One can potentially use established methods such as multiple kernel learning at this step. This might lead to further improvement in results. One can also look into other methods for obtaining basis matrices for sound events. A more comprehensive analysis on a larger dataset with larger number of cities can through more light on the effectiveness of the method presented here. However, this work does give sufficient evidence to show that sounds can be useful in geotagging as well.

8.2 Query by Example Retrieval

Humans have an inherent ability to distinguish and recognize different sounds. Moreover, we are also able to relate and match similar sounds. In fact, we have the capability to detect and relate sound events or "acoustic objects" which we have never encountered before, based on how that phenomenon stands out against the background [Kumar et al., 2014]. This ability plays a crucial role in our interactions with the surroundings and it is also expected that machines have this ability to relate the two audio recordings based on their semantic content. We address the problem of query by example retrieval for audio: given an input audio recording we intend to retrieve audio recordings which are semantically related to it.

Semantic similarity matching and retrieval based on it has received much attention for video and images [Wang et al., 2016], [Ong et al., 2017] and [Qi et al., 2016]. However, in the broad field of machine learning for audio, semantic similarity matching and retrieval based on audio has received limited attention [Wold et al., 1996]. A major focus has been music information retrieval [Casey et al., 2008], [Lew et al., 2006], [Foote, 1997] and semantic retrieval of audio using text queries [Chechik et al., 2008], [Patil and Nemade, 2016]. Our focus here is on non-music and non-speech content, sounds which we hear everyday in our daily life, since they play an important role in defining the overall semantic content of an audio recording. Note that the problem of semantic content matching is a bit different from the problem of audio event detection and classification. Here detection and classification of sound events is not the primary goal, instead the idea is to capture the semantic content of an audio. One method which has been explored considerably for audios is the idea of fingerprinting.

Audio fingerprinting is an acoustic approach that provides the ability to derive a compact representation which can be efficiently matched against other audio clips to compare their similarity or dissimilarity [Ellis, 2009]. Audio fingerprinting has various applications like Broadcast Monitoring[Haitsma and Kalker, 2002], Audio/Song Detection[Wang et al., 2003], Filtering Technology for File Sharing[Sherlock et al., 1994] and Automatic Music Library organization[Cano et al., 2005].

We focus on developing an efficient content-based retrieval system that can retrieve audio clips which contain similar sounds as the query audio clip. One can potentially think of applying the conventional fingerprinting approach [Wang et al., 2003] for matching to find recordings with similar audio content. However, fingerprinting is useful only in finding *exact match*. It has been used for finding multiple videos of the same event [Cotton and Ellis, 2010]. In [Ogle and Ellis, 2007] it is used to find multiple occurrences of a sound event in a recording. But it cannot solve the problem of retrieving all semantically similar files together. In fact even for finding repetitions of the same sound, it does not work well if the sound event is unstructured [Ogle and Ellis, 2007]. The reason is that fingerprinting tries to capture local features specific to an audio recording. It does not try to capture the broader features which might represent a semantically meaningful class. For searching similar recordings based on the content, we need audio retrievals belonging to the correct audio class and not just exact matches as in conventional fingerprinting. Hence, we need representations which can encode class specific information. We try to achieve this by using a Siamese Neural Network. Although the siamese network has been previously explored for representations and content understanding in images and video [Ong et al., 2017, Wang and Gupta, 2015], to the best of our knowledge this is the first work employing it in the context of sound events.

Siamese neural networks incorporate methods that excel at detecting similar instances but fail to offer robust solutions that may be applied to other types of problems like classification. In this paper, we present a novel approach that uses a Siamese network to automatically acquire features which enable the model to distinguish between clips containing distinct audio events and encodes a given audio into a vector fingerprint. We show that the output feature vector has an inherent property to capture semantic similarity between audio containing same events. Although the cost of the learning algorithm itself may be considerable, this compressed representation is powerful as we are able to not only learn them without imposing strong priors like in [Wang et al., 2003], but also to retrieve semantically similar clips by using this feature space.

8.2.1 Siamese Network for Encoding

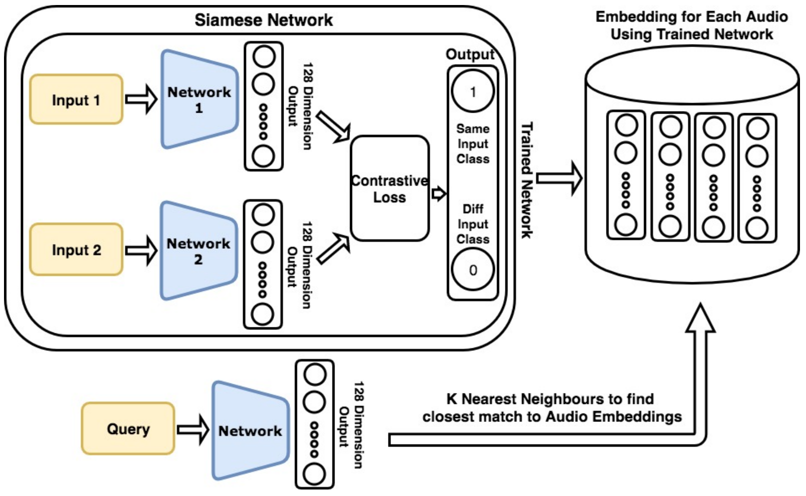


Figure 8.2: Siamese Network For Encoding

We give a neural network based approach to obtain representations or embeddings for audio recordings such that semantically similar audios have similar embeddings. We learn these semantic representations through Siamese Neural Networks. Fig 8.2 shows the framework. A Siamese neural network actually consists of two twin networks. The Siamese network takes in two different inputs, one applied to each network, and is trained to learn the similarity between these inputs. If the inputs are similar then it should predict 1 otherwise 0. The network is used to learn representations for the audio recording, as shown in the figure. An audio query is also embedded through the same network and its embedding is matched with embeddings of recordings in the database to rank them in decreasing order of similarity. This ranking can be done through any distance or similarity measure. Ee use cosine similarity and euclidean distance. Based on the ranked list one can return the *top K* most similar audios.

The Siamese neural network is a class of neural network architectures that contains two or more identical sub-networks, meaning that all sub-networks have the same configuration with the same parameters. Siamese networks have previously been used in tasks involving similarity or identifying relationships between two or more comparable things. Muller et al.[Mueller and Thyagarajan, 2016] used a Siamese network for paraphrase scoring by giving a score to a pair of input sentences. Bromley et al.[Bromley et al., 1994] used a Siamese network for the task of signature verification. In the domain of audio, it has been incorporated for content-based matching in music [Raffel and Ellis, 2015] and in speech to model speaker related information [Chen and Salman, 2011, Zeghidour et al., 2016].

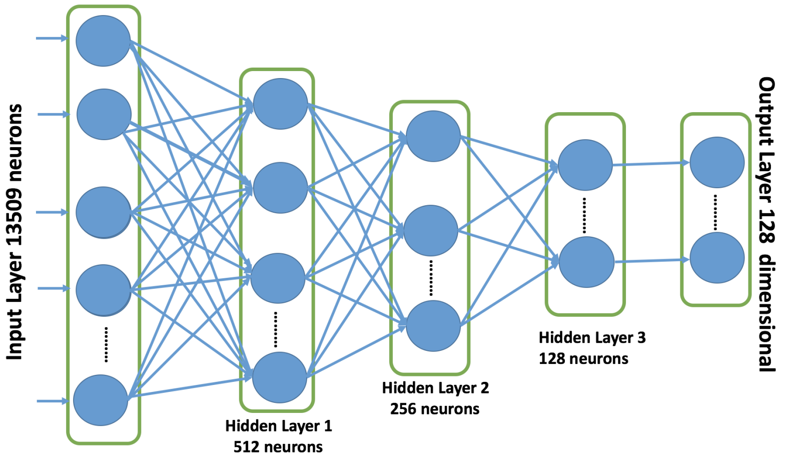


Figure 8.3: Architecture of the Subnetworks in the Siamese Network

Siamese networks offers several advantages. All subnetworks have similar weights which leads to fewer training parameters thus requiring less training data and a lesser tendency to over fit. Moreover, the outputs of each of the subnetworks are representation vectors with the same semantics and this makes them much easier to compare with one other. These characteristics makes them well suited for our task.

Contrastive loss function, shown in Eq 8.6, is used to train the network [Hadsell et al., 2006].

$$L(W, Y, X_1, X_2) = (Y)\frac{1}{2}(D_w)^2 + (1 - Y)\frac{1}{2}\{max(0, m - D_w)\}^2 \tag{8.6}$$

In Eq 8.6, $Y = 1$ if the inputs X_1 and X_2 to the networks are similar, otherwise $Y = 0$. The distance between X_1 and X_2 is defined as the euclidean distance between the mapping from the network function G_W, $D_W(X_1, X_2) = \| G_w(X_1) - G_w(X_2) \|$. $m > 0$ is the margin. The idea behind this margin is that the dissimilar points contribute to the training loss only if the distance between them, D_W, is within the radius defined by margin value m. For the pairs of similar inputs we always want reduce the distance between them.

The network tries to learn the parameters W such that inputs which are expected to be similar are pulled together through G_W and those which are different are pushed apart.

The architecture of the individual sub-networks in the Siamese network is shown in Fig 8.3. The final layer of 128 neurons is also the output layer. Each sub-network is a feed-forward multi layer perceptron (MLP) network. The input to the network are log-frequency spectrograms of audio recordings. The frames in Logspec are concatenated to create one long extended vector. The dimensionality of the inputs are 13509 (See 8.2.4 for details). The network consists of a total of 3 layers after the input layer. The first layer consists of 512 neurons, the second layer 256 neurons and the last layer has 128 neurons. The last layer also serves as the output layer. The activation function in all layers is ReLU $(max(0, x))$. A dropout of 0.3 is applied between all three layers during training. We will refer to the network as $\mathcal{N}_R$

8.2.2 Representations and Retrieval

All audio clips in the audio database are represented through the 128 dimensional output from the network $\mathcal{N}_R$. When a query audio clip is given, we first obtain its 128 dimensional representation using $\mathcal{N}_R$. This representation is then matched to representations of all audios in the database using a similarity or distance function. The clips in database are ranked according to the similarity measure and then the top K clips are returned. In other ways, one can think of it as obtaining K nearest neighbors in the database. Note that all operations are done on fixed length audios of 2 seconds, details are provided in further sections.

All audio clips in the database are represented by their 128 dimensional representations. At the time of testing, we obtain the representation for the query audio clip using the network and compute its similarity with representations of audios in the database. For computing similarity between two representations we use either euclidean distance or cosine similarity.

8.2.3 Dataset and Experimental Setup

We consider the list of sound events from 3 databases, ESC-50[Piczak, 2015a], US8K[Salamon et al., 2014] and TUT 2016 [Mesaros et al., 2016]. A total of 76 sound events are considered. It includes wide range of sound events, animal sounds such as *Dog Barking and Crow*, non-speech human sounds such as *Clapping and Coughing*, exterior sounds such as *Siren, Engine, Airplane* to urban soundscape sounds such as *Street Music, Jackhammer* etc.

We work with audio recordings from YouTube to perform query by example retrieval. For each of the 76 classes, we obtain 100 recordings from YouTube. To obtain relevant results we use <SOUND_NAME sound >(e.g <car horn sound >) as search query. From the returned list we select 100 recordings based on their length and relevance for the query. Very short (<2 seconds) and very long (>10 min) recordings are not considered.

The dataset is divided in the ratio of 70-10-20. 70 percent of the data per class is used for training and the remaining 30 percent data is split 1:2 between validation and testing. Thus, we take 70 samples per class for training, 10 for validation and the remaining 20 for testing, given that we roughly have 100 files per audio class. Overall, we have around 5,000 audio files for training, 760 files for validation and around 1500 files for testing. For our experiments, we operate on 2 second clips from each of these recordings. Hence, our actual database for experiments are fixed length 2 second audio clips in training as well as validation and test sets.

8.2.4 Siamese Network Training

The inputs to the Siamese network must be pairs of audio clips. We assign label 1 to the pairs of clips from the same class and label 0 to the pairs from different classes. We consider two training sets, balanced and unbalanced. The network trained on balanced set will be referred to as $\mathcal{N}_R^B$ and that trained on unbalanced as $\mathcal{N}_R^U$. In the balanced case, to create pairs with positive label ($Y = 1$), we consider all possible pairs belonging to the same audio class. For pairs with negative label ($Y = 0$), a clip belonging to a sound class is randomly paired with a clip from any other sound class. Hence, we end up with equal number of positive and negative label pairs. In the unbalanced case the positive label pairs are obtained in the same way. But for the negative label, we pair a clip belonging to a sound class with all clips not belonging to that sound. Thus, we have a non equal distribution of positive and negative labels.

We used the log spectrogram features, taking 1024 point FFT over a window size of 64ms and an overlap of 32ms per window. Both the axis were converted to the log scale and 79 bins were chosen for the frequency axis whereas 171 quantization bins were chosen for the time scale. We then concatenate these $79 \times 171 = 13509$ features and use as an input to the Siamese Network.

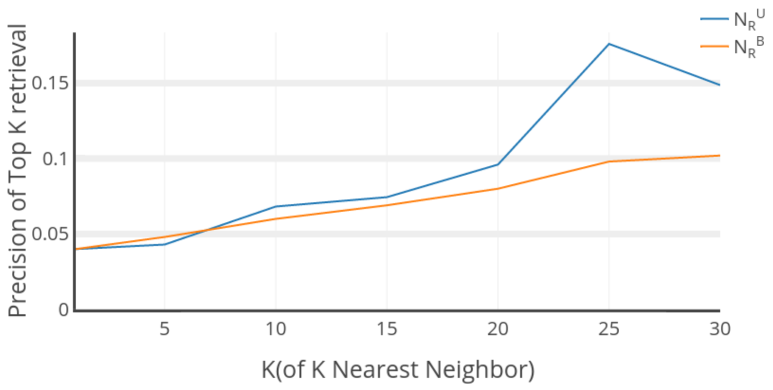


Figure 8.4: Variation of MP^K with different K, from K = 1 to K = 30

All parameters were tuned using the validation set. We train each model to 200 epochs and optimize on the training and validation losses.

8.2.5 Evaluation and results

For any given query audio, we obtain a ranked list of audio clips present in the database which contain similar audio events present in the query clip. We then compute 3 metrics for evaluation which are defined below:

Average Precision

The average precision for a query is defined as mean of precisions at all positive hits.

$$AP = \frac{1}{m_j} \sum_{i=1}^{m_j} Precision_i \tag{8.7}$$

$Precision_i$ measures the fraction of correct items among first i recommendations. This precision is measured at every positive hit in the ranked list. m_j refers to the number of positive items in the database for the query. Average precision is simply the mean of these precision values. We will be reporting the mean of average precision (MAP) over all queries.

Precision at 1

This metric measures the precision at the first positive hit only. The idea is to understand where does the first positive item lie in the ranked list. Again the mean of Precision at 1 (MP^1) over all queries are reported.

Precision of Top K retrieval

This metric measures the quality of retrieved items in the top K items in the ranked list. For each query, we calculate the number of correct class instances in the top K files and then divide that by K to get the precision of the correct class amongst the top K retrieved files and take an average across all queries. Multiplying this score by K tells us the average number of correct

Measures	$\mathcal{N}_R^B$	$\mathcal{N}_R^U$
MAP	0.0241	0.0342
MP^1	0.314	0.436
$MP^{K=25}$	0.099	0.177

Measures	$\mathcal{N}_R^B$	$\mathcal{N}_R^U$
MAP	0.0186	0.0133
MP^1	0.132	0.333
$MP^{K=25}$	0.105	0.133

Table 8.3: Left: Performance using euclidean distance, Right: Performance using cosine similarity

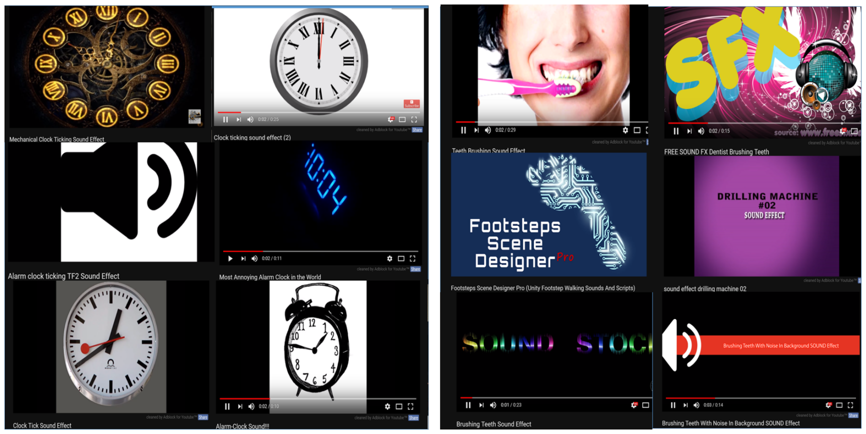

Mechanical Clock Ticking Sound Effect
Clock ticking sound effect (2)
Alarm clock ticking TF2 Sound Effect
Most Annoying Alarm Clock in the World
Clock Tick Sound Effect
Alarm-Clock Sound!!!
SFX
Teeth Brushing Sound Effect
FREE SOUND FX Dentist Brushing Teeth
Footsteps Scene Designer Pro
Footsteps Scene Designer Pro (Unity Footstep Walking Sounds And Scripts)
DRILLING MACHINE #02 SOUND EFFECT
sound effect drilling machine 02
Brushing Teeth With Noise In Background SOUND Effect
Brushing Teeth Sound Effect
Brushing Teeth With Noise In Background SOUND Effect

Figure 8.5: Examples of content-based retrieval. Left: Clock Tick, Right: Brushing Teeth

class matches in the top K of the retrieved list. This measure tells us about the precision of the correct class in the top K retrieved list. Once again the mean of this metric over all queries is reported (MP^K)

The variation of MP^K with K is shown in Figure 8.4. We observe that this metric is maximum around K=25 and hence we report the best possible performance from now on for K=25.

8.2.6 Results and Discussion

We first show performance with respect to queries. From table 8.3, we observe that the Euclidean distance performance exceeds the Cosine similarity performance in the Mean Average Precision measure. This may be due to the fact during siamese network training, euclidean distance is used to measure the closeness between two points. Hence, the learned representations are inherently designed to work better with euclidean distance.

We note that the overall MAP of the system is similar to what has been traditionally observed throughout audio retrieval work [Buckley and Voorhees, 2004]. MP^1 value of around 0.3 (for $\mathcal{N}_R^B$) indicates that the first positive hit on an average is achieved at rank 3. However, for a given specific query it can be much better. We note that the MP^1 values are fairly high, implying that the first positive hit can be easily obtained using the audio embeddings generated using Siamese Network. Also, note that the network $\mathcal{N}_R^U$ performs much better compared to $\mathcal{N}_R^B$. $\mathcal{N}_R^U$ is trained using a larger set of pairs of dissimilar audios and hence it is able learn more discriminitive representations.

The most important metric for understanding performance of a retrieval system is MP^K. The values for MP^K multiplied by K gives us the average number of correct class instances in the top K for a query. A low value of this measure means that a low number of correct class instances are obtained in the top K retrieved files. We are able to obtain fairly reasonable value of MP^K.

Fig8.5 gives visualization of a retrieval example. It shows two examples of queries and their top 6 retrieved similar files. We observe that for the class 'clock tick sound', the retrieval is from classes 'clock tick sound' and 'clock alarm sound', which are both nearly similar audio events. For the class 'brushing teeth sound', the system performs well as their are no other similar audio classes in the database and hence it retrieves 5 out of the 6 files correctly. Overall, it illustrates that our system is capable of delivering content based retrieval of audio recordings.
We described a method for query by example retrieval of audio. Our focus was on audio recordings containing sounds. Our results indicate that the siamese network representations can be used to encode recordings for query by example retrieval. Retrieving audio recordings based on their similarity is an important research topic and recently other works have also been proposed in this area [Jansen et al., 2017].

8.3 Never Ending Learning of Sounds

The learning process in humans is a continuous process. We keep observing and interacting with our surroundings and learning new things about it. Ideally, a machine with intelligence is expected to do the same. It should continuously obtain data and gather information from it. In short, the learning process should be continuous. This idea of continuous learning in a never-ending fashion have been proposed in the past [Carlson et al., 2010, Chen et al., 2013a, Mitchell et al., 2018]. [Carlson et al., 2010] is a never-ending language learning system which is designed to perform two task continuously, (a) *reading* task - which mines the textual data on the web to enhance the knowledge base (b) *learning* task which aims to improve the reading, ensuring that the next time the reading task is done more accurately. This system called Never Ending Language Learner (NELL) has a broad scope. It aims to learn a variety of knowledge by continuously mining textual information from the web.

Closer to what we aim here, [Chen et al., 2013a] presented a never-ending learning system for extracting visual knowledge. This system aimed to understand the semantic content of images on the web and then use this understanding in creating a visual knowledge base. A similar system is desirable for sounds as well. A system which can create a knowledge base about sounds by continuously learning from the web data. We call this Never Ending Learning of Sounds (NELS).

The development of a full-fledged NELS system is beyond the scope of this dissertation. However, an outline of a very early form of such a system is presented here. The goal of such a system is to mine sound related knowledge and examples continuously. The fundamentals of the acoustic intelligence presented in this thesis clearly have a role to play in NELS. As outlined in Figure 1.1, acoustic intelligence in machines requires at least three essential components (a) extracting sound related knowledge by from text (b) recognizing and detecting a large number of sound events and (c) a learning system which links these two systems. A system like NELS is expected to do all of these same 24/7, continuously trying to discover new sound concepts, attempting to build a computational model for it by mining audio on the web and then finally trying to infer some higher level semantic knowledge and commonsense relations about sounds. We addressed the individual components in this dissertation, and the next steps would be to tie up the components together. Moreover, we will also have to incorporate the *never ending* part where the processes are continuously running and updating the knowledge base.

A very early form of the system can be seen on `http://nels.cs.cmu.edu` [Elizalde et al., 2018]. In the current form, the system primarily consists of a front end (`http://nels.cs.cmu.`

`edu`) which shows information about the current state of the system. It shows the vocabulary of sound events which the system can currently detect and examples of videos from YouTube where the system has detected the sound events. We also provide users with several ways to query the system. Users can search for a sound through a text query, and the system returns a set of videos from YouTube where it has detected the presence of the queried sound event. Users can also directly upload an audio recording, and the system can annotate the recording with sound events. The same can be done for any YouTube video as well by providing the URL link of that video to the system. The back end of the system primarily consists of crawling the YouTube for sounds and then learning from the crawled videos. The recognition system has been seeded by detectors built using various datasets such as Audioset, Urbansounds, and ESC-50. In the next phase of NELS, the goal would be to add semi-supervised approaches which will allow the system to learn from unlabeled audio data on the web. Moreover, the system also needs to continuously improve itself as far as detection of sound events is concerned. To this end, human feedback might be important to ascertain that the system is actually improving its performance [Badlani et al., 2018]. The next phase would also involve linking the natural language understanding from text with the detection process in audio/multimedia recordings. The two processes can not only help each other but can also be useful in establishing commonsense relationships for sounds.

Chapter 9

Conclusions and Future Works

What we call the beginning is often the end
And to make an end is to make a beginning.
The end is where we start from.

-T.S. Eliot

Even though weakly supervised learning has played a significant role in scaling audio event detection, one can expect that semi-supervised learning (SSL) can take it further, SSL is designed to exploit both labeled and unlabeled data [Belkin et al., 2006]. We begin the concluding chapter of this dissertation by taking a quick peek at semi-supervised learning for sound events.

Semi-supervised learning is a well-studied problem. The primary idea in semi-supervised classification is that if the instances of a class form a coherent group, then it might be possible to use a large number of unlabeled instances along with a labeled collection of instances to obtain a better decision function Zhu and Goldberg [2009]. Since unlabeled data can be obtained on a very large scale, SSL provides an effective way of obtaining robust models without any additional labeling costs. It has been successfully applied in image classification and segmentation problems, natural language processing Fergus et al. [2009], Liang [2005], Zhu and Goldberg [2009]. For audio, it has mostly been explored in the context of speech recognition Yu et al. [2010]. There have been a few works in the context of sounds as well, such as [Elizalde et al., 2017, Han et al., 2016, Zhang and Schuller, 2012]. We describe one of the most straightforward approaches for self-supervised learning, called *Self Training* for sound event detection [Zhu and Goldberg, 2009].

9.1 Self Training for AED

Self Training is the simplest form of semi-supervised learning. The idea behind self-training is that an algorithm with non-random performance can self-teach and improve its performance. The process is fairly simple. First, a model is trained using the available labeled data as training pool. This trained model is then used to predict on the unlabeled instances. Then a few instances from the unlabeled data along with their predicted label are added to the training pool. The model is then retrained and the process is repeated. The process of Self-Training is shown in Figure 9.1.

The simplicity of self-training makes it an ideal choice for any learning algorithm. One crucial step in self-training process is the instance selection step. The easiest and the most common way to select instances is by choosing those for which the current model predicts labels with

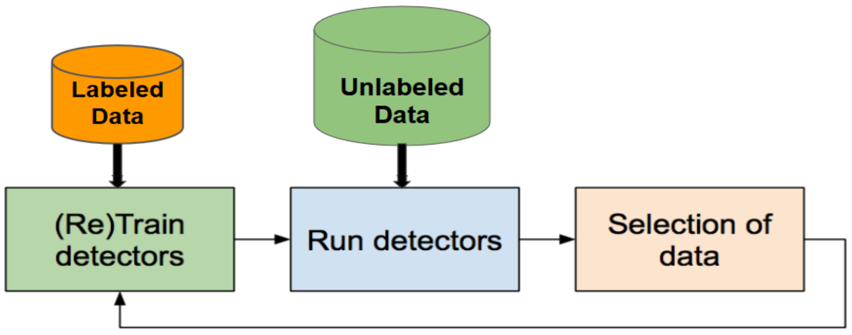


Figure 9.1: Self Training for SSL

high confidence. This is based on the intuition that the predicted labels with high confidences instances are most likely to be correct and hence the predicted label is essentially the ground truth label. However, this can lead to self-bias as the algorithm will repeatedly select instances on which it is already doing well. Here, we introduce another method of instance selection. This instance selection method is based on the idea of *clarity index* Huang et al. [2008] which has previously been used in active learning.

9.1.1 Clarity Index Based Instance Selection

The idea of clarity index originates from active learning where the goal is to select instances for expert labeling. However, we explored it for other problems as well such as classifier fusion and instance ranking Kumar and Raj [2015a,b]. To understand clarity index, let us assume that the training data is $\mathcal{D} = \{(x_1, y_1), (x_2, y_2) ..., (x_n, y_n)\}$. A classifier is trained on this training data and let f be the decision function. Now consider an unlabeled test point x^u. Clarity index is defined in terms of two loss terms namely *Relevance Loss* and *Irrelevance Loss*. The Relevance Loss (RL) and the Irrelevance Loss (IL) are defined as

$$RL(x^u, f) = \frac{1}{|\mathcal{D}_0|} \sum_{x_i \in \mathcal{D}_0} I(f(x_i) - f(x^u)) \tag{9.1}$$

$$IL(x^u, f) = \frac{1}{|\mathcal{D}_1|} \sum_{x_i \in \mathcal{D}_1} I(f(x^u) - f(x_i)) \tag{9.2}$$

$\mathcal{D}_0$ is the set of negative instances in training data and $\mathcal{D}_1$ is the set of positive instances in $\mathcal{D}$. $|\,|$ represents the number of elements in the set. The relevance loss is expected to be low if x^u is relevant (positive) and irrelevance loss is expected to be low if x^u is irrelevant (negative instance). The difference of the two losses $CI = IL - RL$ is expected to be high (close to 1) for positive instances and low (close to -1) for negative instance. Overall, the clarity index helps us rank unlabeled segments to choose instances for retraining.

Higher CI implies that X^u is more likely to be positive. An unlabeled point with very high CI would have outscored a large number of training points and hence is expected to be positive. Similarly, lower CI implies the instance is most likely negative. Based on this idea, we select and add the instances to the training pool. Clarity Index based selection of instances factors in the performance of the classifier on the training set itself. Hence, the clarity index based ranking of unlabeled instances is expected to be more robust compared to that based on classifier output.

Table 9.1: Results (AP) for Self-Training

Events	Baseline	Prob-Sel	CI-Sel
Air Conditioner	0.393	0.413	0.410
Car Horn	0.524	0.520	0.525
Children Playing	0.538	0.534	0.540
Dog Bark	0.762	0.760	0.760
Drilling	0.567	0.581	0.572
Engine Idling	0.538	0.561	0.548
Gunshot	0.678	0.685	0.693
Jackhammer	0.602	0.582	0.620
Siren	0.722	0.727	0.729
Street Music	0.460	0.467	0.460
MAP	**0.578**	**0.583**	**0.589**

9.1.2 Experiments and Results

We use Urbansounds8k Salamon et al. [2014] as the source of labeled data. It contains $8,732$ audio samples of up to 4 seconds in duration. The samples include urban sounds from 10 classes, namely *air conditioner, car horn, children playing, dog bark, drilling, engine idling, gunshot, jackhammer, siren and street music.* The data in this dataset comes pre-divided into ten folds. 9 folds are used for training, and the remaining fold is used for testing.

Unlabeled audio data is downloaded from Youtube. A total of 1000 videos are downloaded. To ensure that unlabeled data contains some audio belonging to relevant event classes, the sound event name is used as the search query on Youtube. However, no labeling is actually done and this data is treated as unlabeled recordings. Top 100 videos belonging to each event class is downloaded. All audio data is segmented into 3.5 second segments which is the average duration of events in the Urbansounds8k dataset. The total number of audio segments turns out to be around 34000.

$\vec{F}$ features described previously (Section 3.6.1) are used to represent audio segments. A binary classifier is trained for each class. Linear kernel SVM is used as the classifier. The selection of instances for retraining is done using two approaches. First, based on the confidence (probabilities) output of the classifier. The second using the clarity index described in the previous section.

Table 9.1 shows the results using both selection methods after a few iterations of selection and retraining. In Table 9.1 Prob-Sel represents classifier confidence (probability) based selection and CI-Sel represents clarity index based selection. Average Precision (AP) for each event has been used as the performance metric. For both methods, we can observe an improvement in MAP over the baseline classifier. With CI based selection we are able to obtain around 2% relative improvement in mean average precision. We expect that a larger dataset and a more exhaustive set of experiments will reveal more about how self-training is suited for audio event detection.
The current sets of experiments are only a small illustration of supervised learning. A wide variety of semi-supervised learning methods have been proposed over the years, and it remains to be seen how automated understanding of sounds can be improved using semi-supervised methods. Existing methods such as graph-based SSL, semi-supervised SVMs, EM-based SSL, Co-Training or Multi-view learning Zhu and Goldberg [2009], to name a few, can be adopted for sound event

detection as well.

For audio event detection scalability of any SSL method becomes very important. Audio event detection usually involves segmenting long audio recordings into short duration (1 second or so) segments. The total number of segments or instances in the learning process can quickly scale up to a large number even for a modest amount of training data, say 100 hours of audio. Under these circumstances, it is important to consider the scalability of the employed SSL method. Hence, investigating the suitability of different SSL methods for audio event detection by considering both performance and scalability is an essential first step for applications of SSL in AED.

In the last few years, deep learning has been the dominant learning framework for a large number of machine intelligence tasks, however, its success has primarily been driven through labeled data. Efforts are ongoing to use unlabeled data in the deep learning methods. For example, it was observed that greedy layer-wise pre-training of deep models is an effective way of utilizing unlabeled data in deep learning models Erhan et al. [2010]. citeweston2012deep proposed another way of exploiting unlabeled data for deep neural networks. They describe different ways of adding a regularizer using the unlabeled data. More recently ladder networks have been proposed for semi-supervised deep learning Rasmus et al. [2015]. Once again, development of such methods for acoustic intelligence remains to be seen.

9.2 Conclusions and Future Works

This dissertation was aimed at developing the idea of acoustic intelligence in machines. For the purposes of development of methods for acoustic intelligence in machines, we proposed that acoustic intelligence should consist of two important components. One involved creating a knowledge of sounds including methods to mine sound names and discovering commonsense relations about sounds. The other is about detecting and recognizing sound events on large scale. Ideally, the two parts are expected to be linked and aid each other. In this dissertation, we looked at methods for each of these individual components.

We developed methods for discovering sound names and relations by mining a large text corpus. In the process, we introduced the idea of more generic *Audible or Sonic phrases*, which are textual phrases which carry a notion of audibility in them. These audible phrases can be sound names, or they can be longer sentences giving the idea of an acoustic phenomenon. We described methods for natural language understanding of sounds; however, a lot still needs to be done. At the end of chapter 2, we pointed out several immediate tasks which need to be explored within this problem.

A significant portion of this dissertation has been devoted to sound event detection, primarily addressing the problem of scalability. We introduced the idea of training audio event detects using weakly labeled data, thereby throwing light on the problem of lack of labeled data. Weakly labeled learning for sounds has shown to be the primary way to scale audio event detection, although, a lot more still needs to be done. We showed that the label noise could adversely affect the learning for weakly labeled data and future methods should explore ways to address this problem. Learning from *webly labeled* recordings needs to be more carefully addressed. By *webly labeled*, we imply weakly labeled data obtained from the web, where the weak labels have been automatically obtained instead of using a manual labeler. The label noise is these situations are expected to be high, and learning methods need to address it appropriately. We also looked into methods to learn from weakly and strongly labeled data under a unified framework. It would be interesting to explore more into this idea, as it has the potential to address the problems of weakly labeled data (especially webly labeled).

Methods under the WEASL framework are similar to semi-supervised learning methods. However, the advantage of weakly labeled data over entirely unlabeled is considerably high. Weak labels still provide supervision, and deep learning methods are easier to employ, compared to completely unlabeled data. As we mentioned before, semi-supervised learning for sounds when done successfully can have a high impact in scaling systems. From a learning perspective, an interesting avenue worth exploring is semi-supervised learning with weakly labeled data. In this case, only weakly labeled data is available along with unlabeled data.

A major focus of the future works on acoustic intelligence should be on the coupling or integration of language and sound event detection. For example, natural language understanding of sounds can be used for collecting weakly labeled data. They can be employed to filter the noise in labels for webly labeled data. In fact, a multi-modal understanding of sounds is required. Knowledge from textual or even other modes such as vision should be used incorporated for sound event recognition and vice versa. Acoustic intelligence in machines has a long way to go. We believe that the progress will be much faster in the coming years and this dissertation will play an important role in exciting new methods yet to come.

Bibliography

Multimedia event detection. `www.nist.gov/itl/iad/mig/med11.cfm`.

Freesound website. `https://freesound.org/`.

Pascal large scale learning challenge, 2008. URL `http://largescale.ml.tu-berlin.de`.

Leila Abdoune and Mohamed Fezari. A sound database for health smart home. In *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*, pages 1–5. IEEE, 2014.

R. B. Adams and Janata P. A comparison of neural circuits underlying auditory and visual object categorization. *Neuroimage*, 16:361–377, 2002.

Claude Alain, Stephen R Arnott, et al. Selectively attending to auditory objects. *Front. Biosci*, 5:D202–D212, 2000.

Peter W Alberti. The anatomy and physiology of the ear and hearing. *Occupational exposure to noise: Evaluation, prevention, and control*, pages 53–62, 2001.

Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. What is an object? In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 73–80. IEEE, 2010.

Ehsan Amid, Annamaria Mesaros, Kalle J Palomaki, Jorma Laaksonen, and Mikko Kurimo. Unsupervised feature extraction for multimedia event detection and ranking using audio content. In *IEEE ICASSP*, pages 5939–5943, 2014.

Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595*, 2015.

Dallas W Anderson, Leslie Kish, and Richard G Cornell. Implications of optimum and approximately optimum stratification.

Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. *Advances in neural information processing systems*, 15:561–568, 2002.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

Relja Arandjelovic and Andrew Zisserman. Look, listen and learn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 609–617. IEEE, 2017.

Relja Arandjelović and Andrew Zisserman. Objects that sound. *arXiv preprint arXiv:1712.06651*, 2017.

Barry Arons. A review of the cocktail party effect. *Journal of the American Voice I/O Society*, 12(7):35–50, 1992.

K Ashraf, B Elizalde, F Iandola, M Moskewicz, J Bernd, G Friedland, and K Keutzer. Audio-based multimedia event detection with DNNs and sparse sampling. In *Proc. of the 5th ACM International Conference on Multimedia Retrieval*, 2015a.

Khalid Ashraf, Benjamin Elizalde, Forrest Iandola, Matthew Moskewicz, Julia Bernd, Gerald Friedland, and Kurt Keutzer. Audio-based multimedia event detection with dnns and sparse sampling. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, ICMR '15, pages 611–614. ACM, 2015b.

Pradeep K Atrey, Namunu C Maddage, and Mohan S Kankanhalli. Audio based event detection for multimedia surveillance. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5, pages V–V. IEEE, 2006.

Mohamed Ayari, Jonathan Delhumeau, Matthijs Douze, Hervé Jégou, Danila Potapov, Jérôme Revaud, Cordelia Schmid, Jiangbo Yuan, et al. INRIA@ TRECVID'2011: Copy detection & multimedia event detection. In *TRECVID*, 2011.

Yusuf Aytar, Carl Vondrick, and Antonio Torralba. Soundnet: Learning sound representations from unlabeled video. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 892–900. 2016.

Yusuf Aytar, Carl Vondrick, and Antonio Torralba. See, hear, and read: Deep aligned representations. *arXiv preprint arXiv:1706.00932*, 2017.

Rohan Badlani, Ankit Shah, Benjamin Elizalde, Anurag Kumar, and Bhiksha Raj. Framework for evaluation of sound event detection in web videos. *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*, 2018.

Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting on association for computational linguistics*, pages 26–33. Association for Computational Linguistics, 2001.

Jie Bao, Yu Zheng, and Mohamed F Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 199–208. ACM, 2012.

Jie Bao, Yu Zheng, David Wilkie, and Mohamed Mokbel. Recommendations in location-based social networks: a survey. *GeoInformatica*, 19(3):525–565, 2015.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247, 2014.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

Kristin Bennett, Ayhan Demiriz, et al. Semi-supervised support vector machines. *Advances in Neural Information processing systems*, pages 368–374, 1999.

Paul N Bennett and Vitor R Carvalho. Online stratified sampling: evaluating classifiers at web-scale. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1581–1584. ACM, 2010.

Frédéric Bimbot and et al. A tutorial on text-independent speaker verification. *EURASIP journal on applied signal processing*, 2004:430–451, 2004.

A. S. Bregman. *Auditory Scene Analysis.* MIT Press, 1990.

Forrest Briggs, Balaji Lakshminarayanan, Lawrence Neal, Xiaoli Z Fern, Raviv Raich, Sarah JK Hadley, Adam S Hadley, and Matthew G Betts. Acoustic classification of multiple simultaneous bird species: A multi-instance multi-label approach. *The Journal of the Acoustical Society of America*, 131(6):4640–4650, 2012.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994.

AL Brown, Jian Kang, and Truls Gjestland. Towards standardization in soundscape preference assessment. *Applied Acoustics*, 72(6):387–392, 2011.

Chris Buckley and Ellen M Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 25–32. ACM, 2004.

Susanne Burger, Qin Jin, Peter F Schulam, and Florian Metze. Noisemes: Manual annotation of environmental noise in audio streams. 2012.

Emre Cakir, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Polyphonic sound event detection using multi label deep neural networks. In *2015 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE, 2015.

William M Campbell, Douglas E Sturim, and Douglas A Reynolds. Support vector machines using gmm supervectors for speaker verification. *IEEE signal processing letters*, 13(5):308–311, 2006.

Pedro Cano, Markus Koppenberger, and Nicolas Wack. Content-based music audio recommendation. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 211–212. ACM, 2005.

Liangliang Cao, Shih-Fu Chang, Noel Codella, Courtenay Cotton, Dan Ellis, Leiguang Gong, Matthew Hill, Gang Hua, John Kender, Michele Merler, et al. Ibm research and columbia university trecvid-2011 multimedia event detection (med) system. 2011.

José Carles, Fernando Bernáldez, and José de Lucio. Audio-visual interactions and soundscape preferences. *Landscape research*, 17(2):52–56, 1992.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3, 2010.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96 (4):668–696, April 2008.

Sourish Chaudhuri, Mark Harvilla, and Bhiksha Raj. Unsupervised learning of acoustic unit descriptors for audio content representation and classification. In *INTERSPEECH*, 2011.

Gal Chechik, Eugene Ie, Martin Rehn, Samy Bengio, and Dick Lyon. Large-scale content-based audio retrieval from text queries. In *Proceedings of the 1st ACM international conference on*

Multimedia information retrieval, pages 105–112. ACM, 2008.

Michael Cheffena. Fall detection using smartphone audio features. *IEEE journal of biomedical and health informatics*, 20(4):1073–1080, 2016.

Ke Chen and Ahmad Salman. Extracting speaker-specific information with a regularized siamese deep network. In *Advances in Neural Information Processing Systems*, pages 298–306, 2011.

Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Neil: Extracting visual knowledge from web data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1409–1416, 2013a.

Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. NEIL: Extracting Visual Knowledge from Web Data. In *International Conference on Computer Vision (ICCV)*, 2013b. `http://www.neil-kb.com/`.

Yixin Chen, Jinbo Bi, and James Ze Wang. Miles: Multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12): 1931–1947, 2006.

Pak-Ming Cheung and James T Kwok. A regularization framework for multiple-instance learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 193–200. ACM, 2006.

Michel Chion. Guide to sound objects. pierre schaeffer and musical research. *Trans. John Dack and Christine North), http://www. ears. dmu. ac. uk*, 1983.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

J Choi, B Thomee, G Friedland, L Cao, K Ni, D Borth, B Elizalde, L Gottlieb, C Carrano, R Pearce, et al. The placing task: A large-scale geo-estimation challenge for social-media videos and images. In *Proceedings of the 3rd ACM Multimedia Workshop on Geotagging and Its Applications in Multimedia*, pages 27–31. ACM, 2014.

Jaeyoung Choi, Howard Lei, Venkatesan Ekambaram, Pascal Kelm, Luke Gottlieb, Thomas Sikora, Kannan Ramchandran, and Gerald Friedland. Human vs machine: establishing a human baseline for multimodal location estimation. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 867–876. ACM, 2013.

Jaeyoung Choi, Gerald Friedland, et al. *Multimodal Location Estimation of Videos and Images.* Springer, 2015.

Szu-Yu Chou, Jyh-Shing Roger Jang, and Yi-Hsuan Yang. Learning to recognize transient sound events using attentional supervision. In *IJCAI*, pages 3336–3342, 2018.

Selina Chu, Shrikanth Narayanan, and C-C Jay Kuo. Environmental sound recognition with time–frequency audio features. *IEEE Transactions on Audio, Speech, and Language Processing*, 17 (6):1142–1158, 2009.

Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Weakly supervised object localization with multi-fold multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 39(1):189–203, 2017.

Chloé Clavel, Thibaut Ehrette, and Gaël Richard. Events detection for an audio-based surveillance system. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 1306–1309. IEEE, 2005a.

Chloé Clavel, Thibaut Ehrette, and Gaël Richard. Events detection for an audio-based surveillance system. In *2005 IEEE International Conference on Multimedia and Expo*, pages 1306–1309. IEEE, 2005b.

William G Cochran. *Sampling techniques*. John Wiley & Sons, 2007.

Courtenay V Cotton and Daniel PW Ellis. Audio fingerprinting to identify multiple videos of an event. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 2386–2389. IEEE, 2010.

Courtenay V Cotton and Daniel PW Ellis. Spectral vs. spectro-temporal features for acoustic event detection. In *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 69–72. IEEE, 2011.

Eduardo Coutinho, Jun Deng, and Bjorn Schuller. Transfer learning emotion manifestation across music and speech. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 3592–3598. IEEE, 2014.

Michael Coyle, Desmond B Keenan, P Alexander Derchak, Marvin Sackner, Frank Wilhelm, Keith Gilroy, Emerance M Gummels, Dana Michael Inman, Paul Kennedy, Mark Mitchnick, et al. Systems and methods for respiratory event detection, September 11 2007. US Patent 7,267,652.

Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Co-clustering based classification for out-of-domain documents. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 210–219. ACM, 2007.

Tore Dalenius. The problem of optimum stratification. *Scandinavian Actuarial Journal*, 1950 (3-4):203–213, 1950.

Tore Dalenius and Margaret Gurney. The problem of optimum stratification. ii. *Scandinavian Actuarial Journal*, 1951(1-2):133–148, 1951.

Tore Dalenius and Joseph L Hodges Jr. Minimum variance stratification. *Journal of the American Statistical Association*, 54(285):88–101, 1959.

Marie-Catherine De Marneffe and Christopher D Manning. Stanford typed dependencies manual. Technical report, 2008.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

Jun Deng, Zixing Zhang, Erik Marchi, and Bjorn Schuller. Sparse autoencoder-based feature transfer learning for speech emotion recognition. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 511–516. IEEE, 2013.

Daniel C Dennett. The role of language in intelligence. 1994.

Arnaud Dessein, Arshia Cont, and Guillaume Lemaitre. Real-time detection of overlapping sound events with non-negative matrix factorization. In *Matrix Information Geometry*, pages 341–371. Springer, 2013.

T G Dietterich, R H Lathrop, and T Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1):31–71, 1997.

Chris Ding, Tao Li, and Michael I Jordan. Convex and semi-nonnegative matrix factorizations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):45–55, 2010.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

Merlin Donald. Précis of origins of the modern mind: Three stages in the evolution of culture and cognition. *Behavioral and brain sciences*, 16(4):737–748, 1993.

Pinar Donmez, Guy Lebanon, and Krishnakumar Balasubramanian. Unsupervised supervised learning i: Estimating classification and regression errors without labels. *The Journal of Machine Learning Research*, 11:1323–1351, 2010.

Daniel R Dooly, Qi Zhang, Sally A Goldman, and Robert A Amar. Multiple instance learning of real valued data. *The Journal of Machine Learning Research*, 3:651–678, 2003.

Gregory Druck and Andrew McCallum. Toward interactive training and evaluation. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 947–956. ACM, 2011.

Lixin Duan, Dong Xu, and Shih-Fu Chang. Exploiting web images for event recognition in consumer videos: A multiple source domain adaptation approach. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1338–1345. IEEE, 2012a.

Lixin Duan, Dong Xu, Ivor Wai-Hung Tsang, and Jiebo Luo. Visual event recognition in videos by learning from web data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9):1667–1680, 2012b.

B. J. Dyson and C. Alain. Representation of concurrent acoustic objects in primary auditory cortex. *Journal of the Acoustic Society of America*, 115(1):280–288, 2004.

Benjamin Elizalde, Anurag Kumar, Ankit Shah, Rohan Badlani, Emmanuel Vincent, Bhiksha Raj, and Ian Lane. Experimentation on the dcase challenge 2016: Task 1acoustic scene classification and task 3sound event detection in real life audio. *IEEE AASP Challenge: Detection and Classification of Acoustic Scenes and Events*, 2016.

Benjamin Elizalde, Ankit Shah, Siddharth Dalmia, Min Hun Lee, Rohan Badlani, Anurag Kumar, Bhiksha Raj, and Ian Lane. An approach for self-training audio event detectors using web data. In *Signal Processing Conference (EUSIPCO), 2017 25th European*, pages 1863–1867. IEEE, 2017.

Benjamin Elizalde, Rohan Badlani, Ankit Shah, Anurag Kumar, and Bhiksha Raj. Nels-never-ending learner of sounds. *arXiv preprint arXiv:1801.05544*, 2018.

D. Ellis. Robust landmark-based audio fingerprinting. 09 2009.

Dan Ellis, Tuomas Virtanen, Mark D Plumbley, and Bhiksha Raj. Future perspective. In *Computational Analysis of Sound Scenes and Events*, pages 401–415. Springer, 2018.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

Miquel Espi, Masakiyo Fujimoto, Daisuke Saito, Nobutaka Ono, and Shigeki Sagayama. A tandem connectionist model using combination of multi-scale spectro-temporal features for acoustic event detection. In *ICASSP*, pages 4293–4296, 2012.

Miquel Espi, Masakiyo Fujimoto, Keisuke Kinoshita, and Tomohiro Nakatani. Exploiting spectro-temporal locality in deep learning based acoustic event detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015(1):26, 2015.

Rong-En Fan, K Chang, C Hsieh, X Wang, and C Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 2008.

Tom Fawcett. Roc graphs: Notes and practical considerations for researchers. *Machine learning*, 31(1):1–38, 2004.

Bernhard Feiten and Stefan Günzel. Automatic indexing of a sound database using self-organizing neural nets. *Computer Music Journal*, 18(3):53–65, 1994.

Jacob Feldman. What is a visual object. *Trends in Cognitive Science*, 5:887–892, 2004.

Zheyun Feng, Songhe Feng, Rong Jin, and Anil K Jain. Image tag completion by noisy matrix recovery. In *European Conference on Computer Vision*, pages 424–438. Springer, 2014.

Rob Fergus, Yair Weiss, and Antonio Torralba. Semi-supervised learning in gigantic image collections. In *Advances in neural information processing systems*, pages 522–530, 2009.

Anthony Fleury, Norbert Noury, Michel Vacher, Hubert Glasson, and J-F Seri. Sound and speech detection and classification in a health smart home. In *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pages 4644–4647. IEEE, 2008.

Jonathan T Foote. Content-based retrieval of music and audio. In *Multimedia Storage and Archiving Systems II*, volume 3229, pages 138–148. International Society for Optics and Photonics, 1997.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

Kurt M Fristrup and William A Watkins. Marine animal sound classification. Technical report, Woods Hole Oceanographic Institution, 1993.

J Gauvain and C Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *Speech and audio processing, IEEE Trans. on*, 1994.

William W Gaver. What in the world do we hear?: An ecological approach to auditory event perception. *Ecological psychology*, 5(1):1–29, 1993.

Jort F Gemmeke, Lode Vuegen, Peter Karsmakers, Bart Vanrumste, et al. An exemplar-based nmf approach to audio event detection. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4. IEEE, 2013.

Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 776–780. IEEE, 2017.

Oguzhan Gencoglu, Tuomas Virtanen, and Heikki Huttunen. Recognition of acoustic events using deep neural networks. In *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*, pages 506–510. IEEE, 2014.

Michael Grant and Stephen Boyd. Cvx: Matlab software for disciplined convex programming.

Richard L Gregory and Oliver Louis Zangwill. *The Oxford companion to the mind.* Oxford University Press, 1987.

Timothdy D. Griffith and Jason D. Warren. What is an auditory object. *Nature Reviews Neuroscience*, 7:2:252–256, 2003.

S Gunasekaran and K Revathy. Content-based classification and retrieval of wild animal sounds using feature selection algorithm. In *machine learning and computing (ICMLC), 2010 second international conference on*, pages 272–275. IEEE, 2010.

Guodong Guo and Stan Z Li. Content-based audio classification and retrieval by support vector machines. *IEEE transactions on Neural Networks*, 14(1):209–215, 2003.

Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, Xian Wu, and Zhong Su. Domain adaptation with latent semantic association for named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 281–289. Association for Computational Linguistics, 2009.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE, 2006.

Jaap Haitsma and Ton Kalker. A highly robust audio fingerprinting system. In *Ismir*, volume 2002, pages 107–115, 2002.

Wenjing Han, Eduardo Coutinho, Huabin Ruan, Haifeng Li, Björn Schuller, Xiaojie Yu, and Xuan Zhu. Semi-supervised active learning for sound classification in hybrid learning environments. *PloS one*, 11(9):e0162075, 2016.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

Morris H Hansen, William N Hurwitz, and William G Madow. Sample survey methods and theory. 1953.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. Task-oriented learning of word embeddings for semantic relation classification. *CoNLL 2015*, page 268, 2015.

Tomoki Hayashi, Shinji Watanabe, Tomoki Toda, Takaaki Hori, Jonathan Le Roux, and Kazuya Takeda. Bidirectional lstm-hmm hybrid system for polyphonic sound event detection. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, pages 35–39, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Laurie M Heller, Benjamin Skerritt, and Emily Ammerman. Perceptually important acoustic features of environmental sounds. *The Journal of the Acoustical Society of America*, 125(4): 2724–2724, 2009.

Hynek Hermansky, Daniel PW Ellis, and Sangita Sharma. Tandem connectionist feature extraction for conventional hmm systems. In *icassp*, pages 1635–1638. IEEE, 2000.

Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing

Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 131–135. IEEE, 2017.

Souta Hidaka and Masakazu Ide. Sound can suppress visual perception. *Scientific reports*, 5: 10483, 2015.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Judy Hoffman, Deepak Pathak, Trevor Darrell, and Kate Saenko. Detector discovery in the wild: Joint multiple instance and representation learning. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 2883–2891, 2015.

Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7304–7308. IEEE, 2013.

Qiang Huang and Stephen Cox. Hierarchical language modeling for audio events detection in a sports game. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 2286–2289. IEEE, 2010.

Thomas S Huang, Charlie K Dagli, Shyamsundar Rajaram, Edward Y Chang, Michael I Mandel, Graham E Poliner, and Daniel PW Ellis. Active learning for interactive multimedia retrieval. *Proceedings of the IEEE*, 96(4):648–667, 2008.

F. T. Husain, M. A. Tagamets, S. J. Fromm, A. R. Braun, and B. Horwitz. Relating neuronal dynamics for auditory object processing to neuroimaging activity: a computational modeling and an fmri study. *Neuroimage*, 21:1701–1720, 2004.

Ozan Irsoy, Olcay Taner Yildiz, and Ethem Alpaydin. Design and analysis of classifier learning experiments in bioinformatics: survey and case studies. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(6):1663–1675, 2012.

Ariel Jaffe, Boaz Nadler, and Yuval Kluger. Estimating the accuracies of multiple classifiers without labeled data. *arXiv preprint arXiv:1407.7644*, 2014.

Aren Jansen, Manoj Plakal, Ratheet Pandya, Daniel PW Ellis, Shawn Hershey, Jiayang Liu, R Channing Moore, and Rif A Saurous. Unsupervised learning of semantic audio representations. *arXiv preprint arXiv:1711.02209*, 2017.

Maxime Janvier, Xavier Alameda-Pineda, Laurent Girinz, and Radu Horaud. Sound-event recognition with a companion humanoid. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 104–111. IEEE, 2012.

Yangqing Jia and Changshui Zhang. Instance-level semisupervised multiple instance learning. In *AAAI*, pages 640–645, 2008.

Yu-Gang Jiang, Xiaohong Zeng, Guangnan Ye, Subhabrata Bhattacharya, Dan Ellis, Mubarak Shah, and Shih-Fu Chang. Columbia-UCF TRECVID2010 multimedia event detection: Combining multiple modalities, contextual concepts, and temporal matching. In *NIST TRECVid Workshop*, volume 2, page 6, 2010.

Qin Jin, Peter Schulam, Shourabh Rawat, Susanne Burger, Duo Ding, and Florian Metze. Event-based video retrieval using audio. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

Biing-Hwang Juang and Lawrence R Rabiner. Automatic speech recognition–a brief history of the technology development. *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara*, 1:67, 2005.

I. Kant. *Critique of pure reason*. Macmillan, 2003.

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

Namit Katariya, Amrit Iyer, and Sunita Sarawagi. Active evaluation of classifiers on large datasets. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 329–338. IEEE, 2012.

S Sathiya Keerthi and Dennis DeCoste. A modified finite newton method for fast solution of large scale linear svms. In *Journal of Machine Learning Research*, pages 341–361, 2005.

Pascal Kelm, Sebastian Schmiedeke, Jaeyoung Choi, Gerald Friedland, Venkatesan Nallampatti Ekambaram, Kannan Ramchandran, and Thomas Sikora. A novel fusion method for integrating multiple modalities and knowledge for multimodal location estimation. In *Proceedings of the 2nd ACM international workshop on Geotagging and its applications in multimedia*, pages 7–12. ACM, 2013.

Muhammad Salman Khan, Miao Yu, Pengming Feng, Liang Wang, and Jonathon Chambers. An unsupervised acoustic fall detection system using source separation for sound interference suppression. *Signal processing*, 110:199–210, 2015.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Serkan Kiranyaz, Ahmad Farooq Qureshi, and Moncef Gabbouj. A generic audio classification and segmentation approach for multimedia indexing and retrieval. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(3):1062–1081, 2006.

Tatsuya Komatsu, Takahiro Toizumi, Reishi Kondo, and Yuzo Senda. Acoustic event detection method using semi-supervised non-negative matrix factorization with a mixture of local dictionaries. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, pages 45–49, 2016.

Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? *arXiv preprint arXiv:1805.08974*, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

M. Kubovy and D. Van Valkenburg. Auditory and visual objects. *Cognition*, 80:97–126, 2001.

A Kumar, P Dighe, R Singh, S Chaudhuri, and B Raj. Audio event detection from acoustic unit occurrence patterns. In *IEEE ICASSP*, pages 489–492, 2012.

A Kumar, R M Hegde, R Singh, and B Raj. Event detection in short duration audio using gaussian mixture model and random forest classifier. In *21st European Signal Processing Conference 2013 (EUSIPCO 2013)*, 2013a.

Anurag Kumar and Bhiksha Raj. A novel ranking method for multiple classifier systems. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1931–1935. IEEE, 2015a.

Anurag Kumar and Bhiksha Raj. Unsupervised fusion weight learning in multiple classifier systems. *arXiv preprint arXiv:1502.01823*, 2015b.

Anurag Kumar and Bhiksha Raj. Weakly supervised scalable audio content analysis. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2016a.

Anurag Kumar and Bhiksha Raj. Audio event detection using weakly labeled data. In *24th ACM International Conference on Multimedia*. ACM Multimedia, 2016b.

Anurag Kumar and Bhiksha Raj. Features and kernels for audio event recognition. *arXiv preprint arXiv:1607.05765*, 2016c.

Anurag Kumar and Bhiksha Raj. Audio event and scene recognition: A unified approach using strongly and weakly labeled data. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 3475–3482. IEEE, 2017a.

Anurag Kumar and Bhiksha Raj. Deep cnn framework for audio event recognition using weakly labeled web data. In *Machine Learning for Audio, 2017 NIPS Workshop on*. NIPS, 2017b.

Anurag Kumar and Bhiksha Raj. Classifier risk estimation under limited labeling resources. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 3–15. Springer, 2018a.

Anurag Kumar and Bhiksha Raj. Learning to recognize sound events using webly labeled recordings *submitted in. IEEE transactions on Neural Networks and Learning Systems*, 2018b.

Anurag Kumar, Rajesh M Hegde, Rita Singh, and Bhiksha Raj. Event detection in short duration audio using gaussian mixture model and random forest classifier. In *21st European Signal Processing Conference 2013 (EUSIPCO 2013)*, 2013b.

Anurag Kumar, Rita Singh, and Bhiksha Raj. Detecting sound objects in audio recordings. In *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*, pages 905–909. IEEE, 2014.

Anurag Kumar, Benjamin Elizalde, and Bhiksha Raj. Audio content based geotagging in multimedia. *Interspeech*, 2017a.

Anurag Kumar, Bhiksha Raj, and Ndapandula Nakashole. Discovering sound concepts and acoustic relations in text. *submitted IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017b.

Anurag Kumar, M. Khadkevich, and C. Fugen. Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes. In *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE, 2018.

Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 951–958. IEEE, 2009.

Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In

Advances in neural information processing systems, pages 556–562, 2001.

Shane Legg, Marcus Hutter, et al. A collection of definitions of intelligence. *Frontiers in Artificial Intelligence and applications*, 157:17, 2007.

Howard Lei, Jaeyoung Choi, and Gerald Friedland. Multimodal city-verification on flickr videos using acoustic and textual features. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2273–2276. IEEE, 2012.

Guillaume Lemaitre and Laurie M Heller. Evidence for a basic level in a taxonomy of everyday action sounds. *Experimental Brain Research*, pages 1–12, 2013.

Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2:1–19, 2006.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

Stan Z Li. Face recognition based on nearest linear combinations. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 839–844. IEEE, 1998.

Stan Z Li. Content-based audio classification and retrieval using the nearest feature line method. *IEEE Transactions on Speech and Audio Processing*, 8(5):619–625, 2000.

Yan Li, Junge Zhang, Kaiqi Huang, and Jianguo Zhang. Mixed supervised object detection with robust objectness transfer. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

Percy Liang. *Semi-supervised learning for natural language*. PhD thesis, Massachusetts Institute of Technology, 2005.

Thomas Lidy and Alexander Schindler. Cqt-based convolutional neural networks for audio scene classification. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, volume 90, pages 1032–1048. DCASE2016 Challenge, 2016.

Hyungjun Lim, Myung Jong Kim, and Hoirin Kim. Cross-acoustic transfer learning for sound event classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 2504–2508. IEEE, 2016.

Joseph J Lim, Ruslan R Salakhutdinov, and Antonio Torralba. Transfer learning by borrowing examples for multiclass object detection. In *Advances in neural information processing systems*, pages 118–126, 2011.

Dima Litvak, Yaniv Zigel, and Israel Gannot. Fall detection of elderly through floor vibrations and sound. In *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pages 4632–4635. IEEE, 2008.

Zhu Liu, Jincheng Huang, Yao Wang, and Tsuhan Chen. Audio feature extraction and analysis for scene classification. In *Multimedia Signal Processing, 1997., IEEE First Workshop on*, pages 343–348. IEEE, 1997.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

Mingsheng Long, Jianmin Wang, Guiguang Ding, Dou Shen, and Qiang Yang. Transfer learning with graph co-regularization. *IEEE Trans. Knowl. Data Eng.*, 26(7):1805–1818, 2014.

Lie Lu, Hong-Jiang Zhang, and Stan Z Li. Content-based audio classification and segmentation by using support vector machines. *Multimedia systems*, 8(6):482–492, 2003.

Xugang Lu, Yu Tsao, Shodai Matsuda, and Chiori Hori. Sparse representation based on a bag of spectral exemplars for acoustic event detection. In *IEEE ICASSP*, pages 6255–6259, 2014.

Zhiwu Lu, Zhenyong Fu, Tao Xiang, Peng Han, Liwei Wang, and Xin Gao. Learning from weak and noisy labels for semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(3):486–500, 2017.

Jiebo Luo, Dhiraj Joshi, Jie Yu, and Andrew Gallagher. Geotagging in multimedia and computer visiona survey. *Multimedia Tools and Applications*, 51(1):187–211, 2011.

Gary Lupyan. Linguistically modulated perception and cognition: the label-feedback hypothesis. *Frontiers in psychology*, 3:54, 2012.

Emma Lynch, Lisa Angeloni, Kurt Fristrup, Damon Joyce, and George Wittemyer. The use of on-animal acoustical recording devices for studying animal behavior. *Ecology and evolution*, 3 (7):2030–2037, 2013.

Richard F Lyon. Machine hearing: An emerging field [exploratory dsp]. *IEEE signal processing magazine*, 27(5):131–139, 2010.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Abdul Majid, Ling Chen, Gencai Chen, Hamid Turab Mirza, Ibrar Hussain, and John Woodward. A context-aware personalized travel recommendation system based on geotagged social media data mining. *International Journal of Geographical Information Science*, 27(4):662–684, 2013.

Michael I Mandel and Daniel PW Ellis. Multiple-instance learning for music information retrieval. In *ISMIR 2008: Proceedings of the 9th International Conference of Music Information Retrieval*, pages 577–582. Drexel University, 2008.

Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. *Advances in neural information processing systems*, pages 570–576, 1998.

Alvin Martin, George Doddington, Terri Kamm, Mark Ordowski, and Mark Przybocki. The det curve in assessment of detection task performance. Technical report, National Inst of Standards and Technology Gaithersburg MD, 1997.

Janvier Maxime, Xavier Alameda-Pineda, Laurent Girin, and Radu Horaud. Sound representation and classification benchmark for domestic robots. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6285–6292. IEEE, 2014.

Stephen McAdams. Recognition of auditory sound sources and events. *Thinking in sound: the cognitive psychology of human audition*, 1993.

Brian McFee, Justin Salamon, and Juan Pablo Bello. Adaptive pooling operators for weakly labeled sound event detection. *arXiv preprint arXiv:1804.10070*, 2018.

Ian McLoughlin, Haomin Zhang, Zhipeng Xie, Yan Song, and Wei Xiao. Robust sound event classification using deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):540–552, 2015.

MediaEval. `http://www.multimediaeval.org/`, 2015.

Prem Melville, Wojciech Gryc, and Richard D Lawrence. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1275–1284. ACM, 2009.

Annamaria Mesaros, Toni Heittola, Antti Eronen, and Tuomas Virtanen. Acoustic event detection in real life recordings. In *18th European Signal Processing Conference*, pages 1267–1271, 2010.

Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Tut database for acoustic scene classification and sound event detection. In *24th European Signal Processing Conference*, volume 2016, 2016.

Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Assessment of human and machine performance in acoustic scene classification: Dcase 2016 case study. In *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2017 IEEE Workshop on*, pages 319–323. IEEE, 2017.

T Mikolov and J Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. *Commun. ACM*, 61:103–115, 2018.

Dalibor Mitrovic, Matthias Zeppelzauer, and Christian Breiteneder. Discrimination and retrieval of animal sounds. In *Multi-Media Modelling Conference Proceedings, 2006 12th International*, pages 5–pp. IEEE, 2006.

Brian CJ Moore. *An introduction to the psychology of hearing*. Brill, 2012.

Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, pages 2786–2792, 2016.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. Discovering semantic relations from the web and organizing them with PATTY. *SIGMOD Record*, 42(2):29–34, 2013.

Stavros Ntalampiras. A transfer learning framework for predicting the emotional content of generalized sound events. *The Journal of the Acoustical Society of America*, 141(3):1694–1701, 2017.

James P Ogle and Daniel PW Ellis. Fingerprinting to identify repeated sound events in long-duration personal audio recordings. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 1, pages I–233. IEEE, 2007.

Eng-Jon Ong, Syed Husain, and Miroslaw Bober. Siamese network of deep fisher-vector descriptors for image retrieval. 02 2017.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760. ACM, 2010.

S Pancoast and M Akbacak. Bag-of-audio-words approach for multimedia event classification. In *Interspeech*, 2012.

Fabio Parisi, Francesco Strino, Boaz Nadler, and Yuval Kluger. Ranking and combining multiple predictors without labeled data. *Proceedings of the National Academy of Sciences*, 111(4): 1253–1258, 2014.

N. M. Patil and M. U. Nemade. Content-based audio classification and retrieval: A novel approach. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pages 599–606, 2016.

R.D. Patterson, M.H. Allerhand, and C. Giguere. Time-domain modeling of peripheral auditory processing: a modular architecture and a software platform. *Journal of the acoustic society of america*, 98:1890–1894, 1995.

SR Payne, WJ Davies, and MD Adams. Research into the practical and policy applications of soundscape concepts and techniques in urban areas. 2009.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.

Huy Phan, Marco Maass, Radoslaw Mazur, and Alfred Mertins. Early event detection in audio streams. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2015a.

Huy Phan, Marco Maaß, Radoslaw Mazur, and Alfred Mertins. Random regression forests for acoustic event detection and classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23:20–31, 2015b.

Huy Phan, Lars Hertel, Marco Maass, and Alfred Mertins. Robust audio event recognition with 1-max pooling convolutional neural networks. *arXiv preprint arXiv:1604.06338*, 2016.

Karol J Piczak. Environmental sound classification with convolutional neural networks. In *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*, pages 1–6. IEEE, 2015a.

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

Aggelos Pikrakis, Theodoros Giannakopoulos, and Sergios Theodoridis. Gunshot detection in audio streams from movies by means of dynamic programming and bayesian networks. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 21–24. IEEE, 2008.

Christopher J Plack. *The sense of hearing*. Routledge, 2018.

Emmanouil Antonios Platanios, Avrim Blum, and Tom Mitchell. Estimating accuracy from unlabeled data. 2014.

Mihail Popescu, Yun Li, Marjorie Skubic, and Marilyn Rantz. An acoustic fall detector system that uses sound height information to reduce the false alarm rate. In *Engineering in Medicine* *and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pages 4628–4631. IEEE, 2008.

Yonggang Qi, Yi-Zhe Song, Honggang Zhang, and Jun Liu. Sketch-based image retrieval via siamese convolutional neural network. pages 2460–2464, 09 2016.

Asma Rabaoui, Manuel Davy, Stéphane Rossignol, and Noureddine Ellouze. Using one-class svms and wavelets for audio surveillance. *IEEE Transactions on information forensics and security*, 3(4):763–775, 2008.

Colin Raffel and Daniel PW Ellis. Large-scale content-based matching of midi and audio files. In *ISMIR*, pages 234–240, 2015.

Manon Raimbault and Daniele Dubois. Urban soundscapes: Experiences and knowledge. *Cities*, 22(5):339–350, 2005.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554, 2015.

Shourabh Rawat, Peter F Schulam, Susanne Burger, Duo Ding, Yipei Wang, and Florian Metze. Robust audio-codebooks for large-scale event detection in consumer videos. 2013.

Jianfeng Ren, Xudong Jiang, Junsong Yuan, and Nadia Magnenat-Thalmann. Sound-event classification using robust texture features for robot hearing. *IEEE Trans. Multimedia*, 19(3): 447–458, 2017.

Bruno H Repp. The sound of two hands clapping: An exploratory study. *The Journal of the Acoustical Society of America*, 81(4):1100–1109, 1987.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3, 1988.

B. Russell. *A history of western philosophy.* Simon and Schuster, 1945.

Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach.* Malaysia; Pearson Education Limited,, 2016.

Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1041–1044. ACM, 2014.

J Sánchez, F Perronnin, T Mensink, and J Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 2013.

Christoph Sawade, Niels Landwehr, Steffen Bickel, and Tobias Scheffer. Active risk estimation. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 951–958, 2010.

R Murray Schafer. *The soundscape: Our sonic environment and the tuning of the world.* Inner Traditions/Bear & Co, 1993.

W Andrew Schloss. On the automatic transcription of percussive music–from acoustic signal to high-level analysis. 1986.

Arda Senocak, Tae-Hyun Oh, Junsik Kim, Ming-Hsuan Yang, and In So Kweon. Learning to localize sound source in visual scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4358–4366, 2018.

RJ Serfling. Approximately optimal stratification. *Journal of the American Statistical Association*, 63(324):1298–1309, 1968.

VK Sethi. A note on optimum stratification of populations for estimating the population means. *Australian Journal of Statistics*, 5(1):20–33, 1963.

Xavier Sevillano, Xavier Valero, and Francesc Alías. Audio and video cues for geo-tagging online videos in the absence of metadata. In *Content-Based Multimedia Indexing (CBMI), 2012 10th International Workshop on*, pages 1–6. IEEE, 2012.

Ankit Shah, Harini Kesavamoorthy, Poorva Rane, Pramati Kalwad, Alexander Hauptmann, and Florian Metze. Activity recognition on a large scale in short videos-moments in time dataset. *arXiv preprint arXiv:1809.00241*, 2018a.

Ankit Shah, Anurag Kumar, Alexander G Hauptmann, and Bhiksha Raj. A closer look at weak label learning for audio events. *arXiv preprint arXiv:1804.09288*, 2018b.

Shihab Shamma. On the role of space and time in auditory processing. *Trends in Cognitive Science*, 5:340–348, 2001.

Ling Shao, Fan Zhu, and Xuelong Li. Transfer learning for visual categorization: A survey. *IEEE transactions on neural networks and learning systems*, 26(5):1019–1034, 2015.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

Barry G Sherlock, DM Monro, and K Millard. Fingerprint enhancement by directional fourier filtering. *IEE Proceedings-Vision, Image and Signal Processing*, 141(2):87–94, 1994.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Ravindra Singh. Approximately optimum stratification on the auxiliary variable. *Journal of the American Statistical Association*, 66(336):829–833, 1971.

Alex J Smola and Bernhard Schölkopf. *Learning with kernels*. Citeseer, 1998.

Alexander J Smola, SVN Vishwanathan, and Thomas Hofmann. Kernel methods for missing variables. In *AISTATS*. Citeseer, 2005.

Y Song and C Zhang. Content-based information fusion for semi-supervised music genre classification. *Multimedia, IEEE Transactions on*, pages 145–152, 2008.

Yi-Cheng Song, Yong-Dong Zhang, Juan Cao, Tian Xia, Wu Liu, and Jin-Tao Li. Web video geolocation by geotagged social resources. *Multimedia, IEEE Transactions on*, 14(2):456–470, 2012.

D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M.D. Plumbley. Detection and classification of acoustic scenes and events. *Multimedia, IEEE Transactions on*, PP, 2015.

Dan Stowell. Computational bioacoustic scene analysis. In *Computational Analysis of Sound Scenes and Events*, pages 303–333. Springer, 2018.

Dan Stowell and Mark D Plumbley. Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning. *PeerJ*, 2014.

Dan Stowell, Mike Wood, Yannis Stylianou, and Hervé Glotin. Bird detection in audio: a survey and a challenge. *arXiv preprint arXiv:1608.03417*, 2016.

Ting-Wei Su, Jen-Yu Liu, and Yi-Hsuan Yang. Weakly-supervised audio event detection using event-specific gaussian filters and fully convolutional networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 791–795, 2017.

Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016.

Naoya Takahashi, Michael Gygli, Beat Pfister, and Luc Van Gool. Deep convolutional neural networks and data augmentation for acoustic event detection. *arXiv preprint arXiv:1604.07160*, 2016.

Jinhui Tang, Shuicheng Yan, Richang Hong, Guo-Jun Qi, and Tat-Seng Chua. Inferring semantic concepts from community-contributed images and noisy tags. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 223–232. ACM, 2009.

Ib Thomsen. A comparison of approximately optimal stratification given proportional allocation with other methods of stratification and allocation. *Metrika*, 23(1):15–25, 1976.

Yuji Tokozume and Tatsuya Harada. Learning environmental sounds with end-to-end convolutional neural network. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 2721–2725. IEEE, 2017.

Tatiana Tommasi, Francesco Orabona, and Barbara Caputo. Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In *Proceedings of IEEE Computer Vision and Pattern Recognition Conference*, 2010.

A. Treisman. Properties, parts and objects. *Handbook of perception and human performance, Cognitive processes and performance*, 2:35–1:35–70, 1986.

Michele Trevisiol, Hervé Jégou, Jonathan Delhumeau, and Guillaume Gravier. Retrieving geo-location of videos with a divide & conquer hierarchical multimodal approach. In *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*, pages 1–8. ACM, 2013.

Shao-Yen Tseng, Juncheng Li, Yun Wang, Joseph Szurley, Florian Metze, and Samarjit Das. Multiple instance deep learning for weakly supervised audio event detection. *arXiv preprint arXiv:1712.09673*, 2017.

G Valenzise, L Gerosa, M Tagliasacchi, F Antonacci, and A Sarti. Scream and gunshot detection and localization for audio-surveillance systems. In *Advanced Video and Signal Based Surveillance, IEEE Conference on*, pages 21–26, 2007.

Aäron Van Den Oord, Sander Dieleman, and Benjamin Schrauwen. Transfer learning by supervised pre-training for audio-based music classification. In *Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, 2014.

Jan C Van Gemert, Cor J Veenman, Arnold WM Smeulders, and Jan-Mark Geusebroek. Visual word ambiguity. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(7): 1271–1283, 2010.

A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008.

Tuomas Virtanen, Mark D Plumbley, and Dan Ellis. *Computational analysis of sound scenes and events*. Springer, 2018.

Avery Wang et al. An industrial strength audio search algorithm. In *Ismir*, volume 2003, pages 7–13. Washington, DC, 2003.

Chong Wang, Weiqiang Ren, Kaiqi Huang, and Tieniu Tan. Weakly supervised object localization with latent category learning. In *European Conference on Computer Vision*, pages 431–445. Springer, 2014a.

Fangzhou Wang, Sourish Chaudhuri, Daniel Ellis, and Nathan Reale. Automatic smoothed captioning of non-speech sounds from audio, September 28 2017. US Patent App. 15/245,152.

Feng Wang, Zhanhu Sun, Yu-Gang Jiang, and Chong-Wah Ngo. Video event detection using motion relativity and feature selection. *Multimedia, IEEE Transactions on*, 16(5):1303–1315, 2014b.

J Wang and J Zucker. Solving the multiple-instance problem: A lazy learning approach. In *Proc. of 7th International Conference on Machine Learning*, 2000.

Liwei Wang, Yin Li, and Svetlana Lazebnik. Learning deep structure-preserving image-text embeddings. pages 5005–5013, 06 2016.

Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2015.

Yun Wang and Florian Metze. A transfer learning based feature extractor for polyphonic sound event detection using connectionist temporal classification. 2017.

Yun Wang, Juncheng Li, and Florian Metze. Comparing the max and noisy-or pooling functions in multiple instance learning for weakly supervised sequence learning tasks. *arXiv preprint arXiv:1804.01146*, 2018.

William H Warren and Robert R Verbrugge. Auditory perception of breaking and bouncing events: a case study in ecological acoustics. *Journal of Experimental Psychology: Human perception and performance*, 10(5):704, 1984.

X Wei, J Wu, and Z Zhou. Scalable multi-instance learning. In *Data Mining (ICDM), 2014 IEEE Intl. Conf. on*. IEEE, 2014.

Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, 2016.

Paul J Werbos. Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization*, pages 762–770. Springer, 1982.

Erling Wold, Thom Blum, Douglas Keislar, and James Wheaten. Content-based classification, search, and retrieval of audio. *IEEE multimedia*, 3(3):27–36, 1996.

Jia Wu, Shirui Pan, Xingquan Zhu, Chengqi Zhang, and S Yu Philip. Multiple structure-view learning for graph classification. *IEEE transactions on neural networks and learning systems*, pages 1–16, 2017.

Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. Achieving human parity in conversational speech recognition. *arXiv* *preprint arXiv:1610.05256*, 2016.

Ziyou Xiong, Regunathan Radhakrishnan, Ajay Divakaran, and Thomas S Huang. Audio events detection based highlights extraction from baseball, golf and soccer games in a unified framework. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 5, pages V–632. IEEE, 2003a.

Ziyou Xiong, Regunathan Radhakrishnan, Ajay Divakaran, and Thomas S Huang. Audio events detection based highlights extraction from baseball, golf and soccer games in a unified framework. In *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 3, pages III–401. IEEE, 2003b.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2(3):5, 2015a.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1785–1794, 2015b.

Yong Xu, Qiuqiang Kong, Qiang Huang, Wenwu Wang, and Mark D Plumbley. Attention and localization based on a deep convolutional recurrent model for weakly supervised audio tagging. *arXiv preprint arXiv:1703.06052*, 2017.

Zhe Xu, Shaoli Huang, Ya Zhang, and Dacheng Tao. Augmenting strong supervision using web data for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision*, pages 2524–2532, 2015c.

Jun Yang, Rong Yan, and Alexander G Hauptmann. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 188–197. ACM, 2007.

Guangnan Ye, I Jhuo, Dong Liu, Yu-Gang Jiang, DT Lee, Shih-Fu Chang, et al. Joint audio-visual bi-modal codewords for video event detection. In *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*, page 39. ACM, 2012.

Guangnan Ye, Yitong Li, Hongliang Xu, Dong Liu, and Shih-Fu Chang. Eventnet: A large scale structured concept library for complex event detection in video. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 471–480. ACM, 2015.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

Dong Yu, Balakrishnan Varadarajan, Li Deng, and Alex Acero. Active learning and semi-supervised learning for speech recognition: A unified framework using the global entropy reduction maximization criterion. *Computer Speech & Language*, 24(3):433–444, 2010.

Neil Zeghidour, Gabriel Synnaeve, Nicolas Usunier, and Emmanuel Dupoux. Joint learning of speaker and phonetic similarities with siamese networks. In *INTERSPEECH*, pages 1295–1299, 2016.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

Han Zhang, Tao Xu, Mohamed Elhoseiny, Xiaolei Huang, Shaoting Zhang, Ahmed Elgammal, and Dimitris Metaxas. Spda-cnn: Unifying semantic part detection and abstraction for fine-grained recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1143–1152, 2016.

Haomin Zhang, Ian McLoughlin, and Yan Song. Robust sound event recognition using convolutional neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 559–563. IEEE, 2015.

Jianguo Zhang, Marcin Marszałek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision*, 73(2):213–238, 2007.

Zixing Zhang and Björn Schuller. Semi-supervised learning helps in sound event classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 333–336. IEEE, 2012.

Zhi-Hua Zhou and Jun-Ming Xu. On the relation between multi-instance learning and semi-supervised learning. In *Proceedings of the 24th international conference on Machine learning*, pages 1167–1174. ACM, 2007.

Zhi-Hua Zhou and Min-Ling Zhang. Neural networks for multi-instance learning. In *Proceedings of the International Conference on Intelligent Information Technology, Beijing, China*, pages 455–459, 2002.

Zhi-Hua Zhou, Yu-Yin Sun, and Yu-Feng Li. Multi-instance learning by treating instances as non-iid samples. In *Proceedings of the 26th annual international conference on machine learning*, pages 1249–1256. ACM, 2009.

Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3:1–130, 2009.

Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003.

Xiaodan Zhuang, Jing Huang, Gerasimos Potamianos, and Mark Hasegawa-Johnson. Acoustic fall detection using gaussian mixture models and gmm supervectors. 2009.

Xiaodan Zhuang, Xi Zhou, Mark A Hasegawa-Johnson, and Thomas S Huang. Real-world acoustic event detection. *Pattern Recognition Letters*, 31(12):1543–1551, 2010.

Eberhard Zwicker and Hugo Fastl. *Psychoacoustics: Facts and models*, volume 22. Springer Science & Business Media, 2013.