NORTHWESTERN UNIVERSITY

Sound Event Annotation and Detection with Less Human Effort

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

Bongjun Kim

EVANSTON, ILLINOIS

June 2020

© Copyright by Bongjun Kim 2020

All Rights Reserved

ABSTRACT

Sound Event Annotation and Detection with Less Human Effort

Bongjun Kim

Sound is one of the most important mediums to understand the environment around us. Identifying a sound event in prerecorded audio (such as a police siren, a dog bark, or a creaking door in soundscapes) leads to a better understanding of the context where the sound events occurred. To do so, we record a sound scene, search for sound events of interest in the recording, determine their time positions (i.e., start and end), and give them meaningful text labels.

A typical way for a human to find and annotate a sound event of interest in unlabeled audio recordings is simply to listen to the audio until one hears it and finds accurate onset and offset of the event. This sound event annotation process is very labor-intensive. My research goal is to reduce the human effort required for sound event detection and annotation.

In this dissertation, I present methods to speed up the sound event annotation process. My specific goals are divided into two, in terms of what the annotated data is used for. First, sound event annotation is essential to quantify the contents of a recorded acoustic scene for a direct analysis. For this purpose of sound event annotation, I focus on building a system that helps a user to find sound events of interest and annotate them as quickly as possible.

Secondly, sound event annotation is also one of the essential steps to provide the training data needed for building an AI machine that automatically identifies sound events (e.g., sound-based surveillance systems). My focus for this situation is to help human annotators spend less time labeling training data, but still build a high-performance machine learning model with less annotation effort.

To achieve these goals, I present a human-in-the-loop system for sound event annotation, *I-SED* that lets users find sound events of interest roughly twice as fast as manually labeling the target sounds. Then, I present methods that can solve the problem where query-by-example search of I-SED could fail if the initially selected region (i.e., a query) contains multiple sound events. The solution is a new way of improving query-by-example audio search using user's vocal imitations (i.e., Imitating what they do or do *not* want in a query recording) which would help a user to find target sound events quickly. Finally, I present a new type of audio labeling, called *point labeling*, which makes it easier for human annotators to provide ground truth labels to train a machine learning sound event detection system. Point labels provide more information than weak labels, but are still faster to collect than strong labels. I show that a model trained on point-labeled data is comparable to one trained on the typical type of labeled data, strongly labeled data that is harder to collect.

This dissertation will be a valuable resource for researchers and practitioners who are looking for new annotation methods under a limited budget. I expect that it will facilitate the process of sound scene understanding of humans as well as AI systems.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

1.1. Background and Motivation

Sound is one of the most important mediums for us to understand the environment around us. Identifying a sound event (such as a police siren, a dog bark, or a creaking door) leads to a better understanding of the context where the sound events occurred.

To label sound events, people record a sound scene, search for sound events of interest in the recording, determine their time positions (i.e., start and end), and give them meaningful text labels. This task is called *Sound event annotation*. Figure 1.1 shows an example of annotated sound events in a recording. Sound event annotation can be applied to many kinds of audio, for many different purposes. Examples include speech diarization [**85**], labeling music recordings by predominant instrument [**28**], labeling nature recordings with the species of animals heard in the recording [**61, 100**], and identifying gunshots in city recordings [**104**].

Sound event annotation is essential to quantify the sound scene in a recording. Statistics on annotated sound events are important information to understand an environment and make a decision for many different tasks. For example, ecologists might want to know when and how often a bird was signing in an environmental recording to understand bird migration patterns around the environment. Language pathologists might be interested in how often and how long their patients are exposed to a certain type of noise sound during a day. To obtain such information, they need to record audio in a particular environment, search for sound events of their interest in the recording and label them.

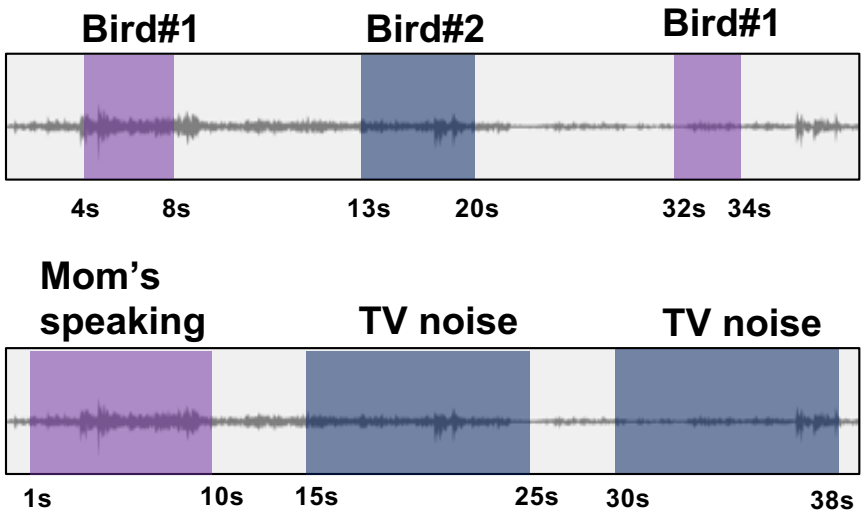


Figure 1.1. Examples of sound event annotation. Each sound event of interest is labeled with a text tag and its temporal location within a recording.

Sound identification and annotation of prerecorded audio are also essential for making searchable audio content in many existing online multimedia (or audio) repositories that contain a large amount of audio content. This is because it allows us to leverage all the work done for text-based search and apply it to multimedia search. Example public repositories of audio include the online music sharing service SoundCloud which contains over 200 million tracks[1], and the online audio sample library FreeSound which contains more than 400,000 audio files[2].

The standard way of searching for audio in these online repositories is text-based search. People search for audio with descriptive text metadata (text keywords associated with an audio file or a sub-portion of the audio file). However, text-based search is not possible when there is no tag provided for a relevant portion of the audio content. Machine-assisted searching for sound events in a lengthy recording (e.g. a 24-hour long recording of a natural

[1]https://blog.soundcloud.com/2019/02/13/celebrating-the-200-millionth-track-uploaded-to-soundcloud/
[2]https://blog.freesound.org/?p=942

Figure 1.2. Typical steps of a supervised machine learning process. Labeling data is an essential step for training supervised machine learning models. The performance of a trained model is affected by the quality and quantity of labeled data.

scene) is currently practical only when time-coded labels have been added to the recording. When this is not the case, then the audio must be annotated first.

Sound event annotation is also one of the essential steps for building an AI machine that automatically identifies sound events (e.g., sound-based surveillance systems). A typical approach to building such a system is supervised machine learning. The system learns a function that maps audio data to a known sound event label so it can predict a label of a new sound event. Therefore, in order to build such a system using supervised machine learning, data labeling has to be performed before model training, as shown in Figure 1.2.

Recently, building systems for automatic sound object identification has obtained much attention from both industry and the research community. They have put significant efforts into sound event annotation to foster the development of the AI systems. Every year since 2016, the Detection and Classification of Acoustic Scenes and Events (DCASE) community[3] has released annotated datasets for DCASE challenges [**72**] where researchers can develop and evaluate computational scene and event analysis methods with the provided public datasets. Google also has released AudioSet [**29**] containing a collection of 2 million 10-second sound clips drawn from YouTube videos with their labels of 632 sound event classes. Sound of New York City (SONYC) [**5**] is a noise monitoring project where large-scale audio data collection and labeling have been performed.

[3]http://dcase.community/

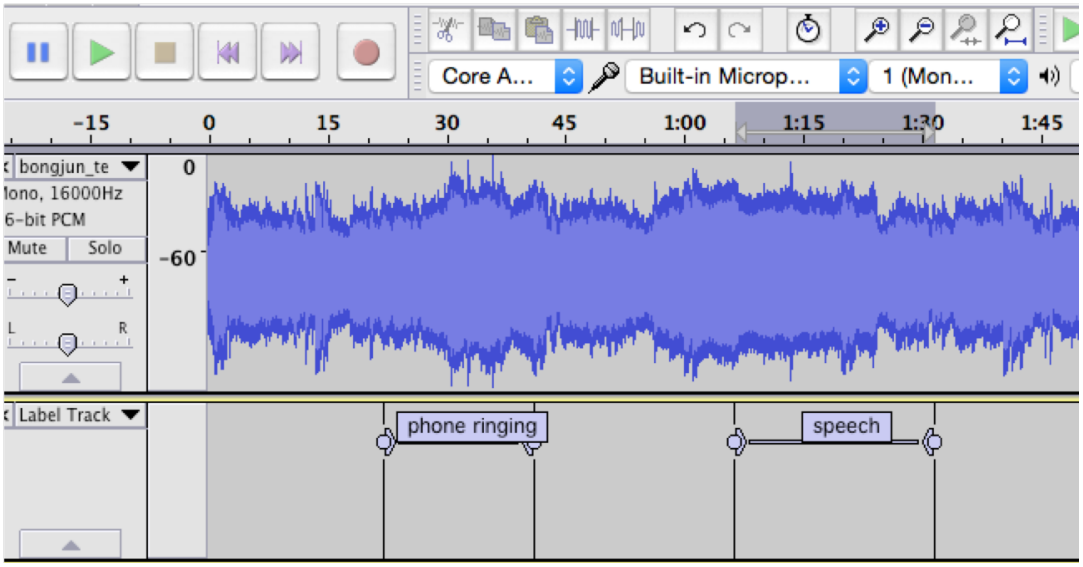


Figure 1.3. An example of sound event annotation tools: Audacity [**64**]. A user can load a label track in addition to an audio track and add a text tag with its temporal locations (onset and offset of a sound event).

A typical way to annotate sound events of interest in a recording is to simply listen to the audio until one hears them and determine the start and stop times of the sound events (i.e. onset and offset of an event). For this task, one can use a visual interface where a user can play an audio recording and add labels of sound events. Figure 1.3 shows an example of sound event annotation using Audacity, which is an open-source digital audio editor. It provides a visual annotation environment where a user can select a sub-section of an audio track and add text-tags to the selected regions.

Labeling sound events of interest in a lengthy audio recording or in a large database of unlabeled audio files is a very time-consuming task. For example, one might need to listen to several seconds of a sound event to figure out it is the sound of a car passing by. This is typically much slower than identifying objects in images (e.g., circle the cat in this image). Images are viewable all at once and it is easy to scan one's eyes in any direction and at any speed. However, to successfully identify a sound object, one listens start-to-finish at a

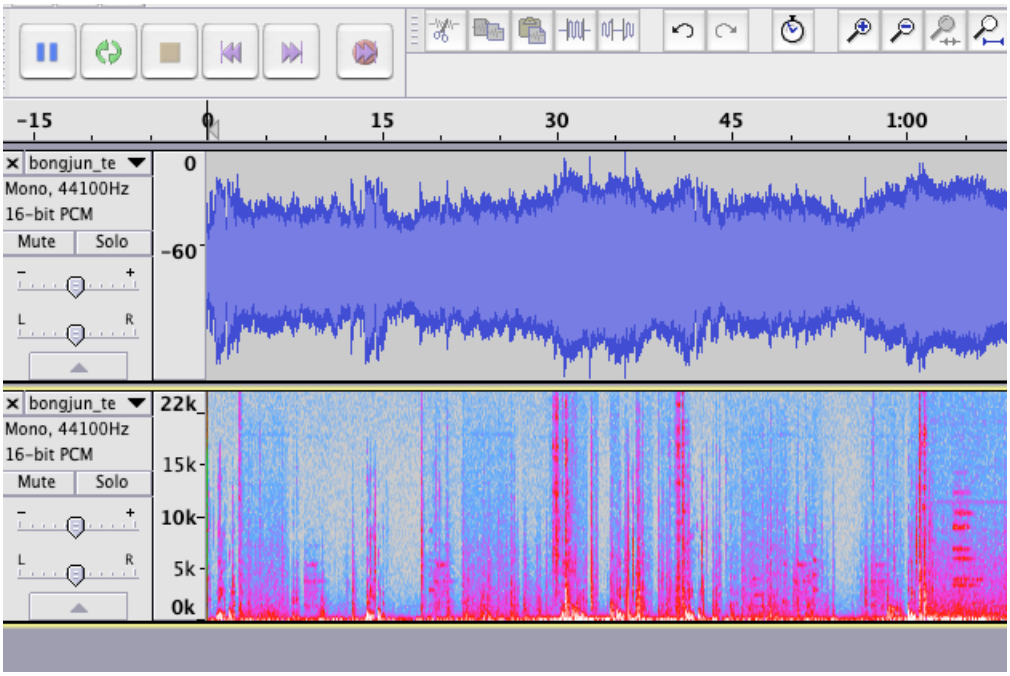


Figure 1.4. Audio visualization in Audacity: a waveform (the first track) and a magnitude spectrogram (the second track). A waveform represents amplitude of audio signal over time. A magnitude spectrogram represents how energies of different frequencies of a sound change over time. Its horizontal dimension is time, the vertical dimension is frequency. Frequency can be thought of as a pitch of a sound. The color of each time-frequency point indicates the strength of the energy at the time-frequency bin. Red means high energy (loud sound), blue means low energy (quiet sound).

fixed rate. Moreover, marking the exact start and stop times (i.e., onset and offset) of a sound event might require one to listen to certain regions of the recording multiple times [**49**]. While many audio annotation tools support a visual representation of audio such as a waveform and a magnitude spectrogram (see Figure 1.4), it is almost impossible to correctly identify sound events only by looking at those visual representations of audio. Therefore, listening to audio is an essential step to fully identify sound events, which makes sound event annotation labor-intensive.

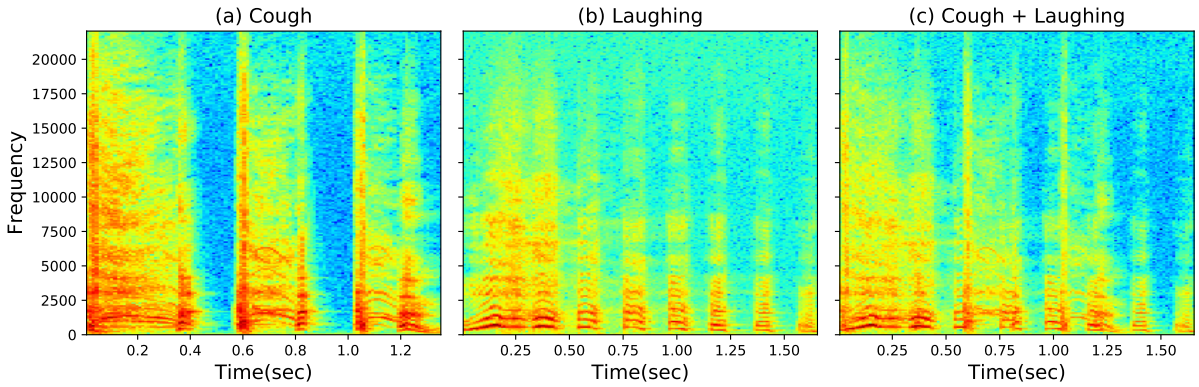


Figure 1.5. Magnitude spectrograms of humans couphing (a), laughing (b), and a mixture of the two.

Identifying sound events and finding their correct time-boundaries become more difficult when multiple sound events are fully overlapped (e.g., a cough is concurrent with television noise and the sound of a blender). Figure 1.5 shows spectrograms of two overlapping sound events (coughing and laughing). This is not occlusion. This is a case of simultaneous overlapped sounds. A visual analog would be observing something through a reflective glass window where two overlapping images occur, the one reflected in the window and the one visible through the window. Overlapped sounds are far more common than isolated sounds in the real world and marking the start and stop of sounds that are overlapped may impose more cognitive load on a human annotator.

This dissertation aims to reduce the human effort required for sound event annotation. To do so, I set up two different goals depending on what the annotated data is used for. The purpose of sound event annotation can be either to directly use the annotation to solve a problem or to build machine learning models. One might label audio to directly quantify sound events of interest in a recording and use the information for a direct analysis. In this case, the goal is to collect accurate human-labeled data quickly. Alternatively, one might annotate audio as a precursor to training a machine learning model

for automatic sound recognition. In this situation, the goal is to build a high-performance machine learning model with less human annotation effort.

1.2. Problem Statement

As mentioned in the previous section, a typical way for a human to annotate a sound event in an audio recording is simply to listen to the audio until one hears it. Done this way, human effort for annotation would linearly increase with the amount of audio one needs to listen. In this section, I address this problem in more detail by examining two different scenarios depending on what the annotated data is used for: 1) quantifying sound scenes for a direct analysis which requires very accurate labels, 2) building a machine learning model on the annotated audio data.

1.2.1. Problems with collecting ground-truth labels for direct analysis

Imagine you are a language pathologist and trying to analyze the relationship between children's language development and their listening environment. You collect days of audio files recorded from a wearable microphone installed on a patient. You listen to the audio tracks and found an interesting class of sound events (e.g. their mother speaking to them) which might affect children's language development. It occurs repeatedly in their days of recordings. Now you are interested in how often and when the sound event occurred. This requires finding all the temporal locations of this class of sound events within the recordings.

Searching through lengthy audio files manually is very time consuming and a natural thought is to automate the process. One can build a machine to automatically identify various sound events and let the machine perform the annotation task. However, the typical approach to building automatic sound recognition systems uses supervised machine learning

algorithms that still require annotated training audio data. Examples include neural networks [**81, 38**], Gaussian Mixture Models (GMM) [**109**], decision trees [**60**] and Support Vector Machines (SVM) [**84**]. While, as mentioned in the previous section, there are public datasets available (e.g., DCASE dataset, AudioSet), these datasets do not cover every arbitrary class of sound events (e.g., a particular children's mother speaking to them). If the existing repositories do not contain labeled sound events of current interest, we cannot build a machine to identify the sound events. This problem remains the same even when one uses pre-trained models to annotate audio or automatic annotation tools because the models or tools have been built to detect a finite set of pre-defined sound classes. It might be difficult for one to collect enough training examples of the sound class that you just found interesting.

Moreover, automatic annotation is not the best solution when you need to make sure all the collected sound events are labeled with correct class names and accurate onset and offset (i.e., human expert-level accuracy of labeling). There is still a gap between human and machine abilities to identify sound events. For example, the top-ranked audio tagging system in the recent DCASE challenge task5 [4] produces predicted labels that diverge from the human-generated labels by an F-score of 0.26 on the DCASE challenge dataset containing 8 different classes.

The scenario and problems described above show us that there are situations where manual audio search is required. However, even though manual sound event annotation by human experts leads to more accurate results than using automatic detection systems, hand-labeling events in recordings is prohibitively labor-intensive. For example, to monitor how long a patient was exposed to a certain type of a sound event during a day, one needs to listen to 24 hours of audio. Typically, this would take more than 24 hours to label. Although

[4]http://dcase.community/challenge2019/task-urban-sound-tagging

one could listen to a recording at higher playback speed (e.g., 2x or 3x), it might make it harder to find accurate time boundaries of an event which often requires repeated listening to a small-time region in a recording [**49**]. **In this dissertation, I present new methods to speed up human's searching for a set of sound events of interest in an audio recording when there are too few labeled examples (e.g., one) of the sound class to train a state-of-the-art machine audio labeling system.**

1.2.2. Problems with building automatic sound event detection systems

Imagine you are working on building an automatic noise monitor device for environmental sounds, as is being done in New York University's Sound of New York City (SONYC) project [**5**]. You try to analyze what kind of noises occur by day and by night. You have a list of noise sources which could happen near your place and a set of recordings collected from YouTube videos and microphones installed in front of your house. Now you want to build a machine learning model to automatically detect sound events of the pre-defined classes. The typical approach to training automatic sound event detection systems is supervised machine learning which requires annotated training data. So you need to label the collected audio data to use them as training dataset.

For a supervised machine learning model to be maximally effective at detecting sound events with their onset/offset times within a recording, they need to be trained on audio data with time-coded labels that indicate the start and stop times of sound events (*strongly labeled data*). Training examples without time-information of sound events will generate noisy training signal, thus they will prevent models from learning accurate mappings between sound events and the ground-truth labels. The problem is that manually annotating each sound's onset and offset within a recording is a very time-consuming task. It often requires

repeated listening and adjusting of label time boundaries on a visual interface [**49**]. Moreover, building high-performance machine learning models usually requires lots of training data (e.g. tens of thousands of labeled audio files). This imposes a large burden on human annotators. **In this dissertation, I present a new type of sound event labeling that requires less human annotation cost as well as a new way of training supervised machine learning models with the new labels.**

1.3. Summary of Contribution

In this section, I summarize the main contributions of my dissertation. Chapter 2 and 3 contain methods to speed up sound search for sound even annotation. Chapter 4 introduces a new type of labeling *point-labeling* that requires much less annotation effort than collecting strong labels, and present methods to train a sound event detection model using point-labeled audio data.

1.3.1. Contribution in Chapter 2

I present a new human-in-the-loop sound search method to speed up human annotation of a recording. It leverages machine learning's ability to learn from human-provided examples not to replace a human annotator, but to speed up human annotator. The human-in-the-loop search method helps a user look at promising regions first in a recording, enabling the user to find a set of sound events of interest very quickly not listening to the entire recording. To evaluate the method, I built a human-in-the-loop interface for sound event annotation, called Interactive Sound Event Detector (I-SED) where the annotation is performed by a collaboration between a user and a machine. The user study shows that I-SED helps users label target sound events twice as fast as labeling them manually.

I-SED is the first general-purpose sound labeling interface where an interactive machine learning approach is applied to sound event annotation. This method can be used in any situation where ground-truth sound event labels are required. I also present an in-depth study about how to evaluate the human-in-the-loop interface. I believe the study can be a useful reference to one who needs to evaluate the human-in-the-loop system where both machine's performance and human's ability can affect the system's overall performance.

My works in Chapter 2 have been published in the Transaction on Interactive Intelligent System (TiiS) [**49**] and presented at ACM conference on Interactive User Interface (IUI) [**48**].

1.3.2. Contribution in Chapter 3

I present a new method for a user to improve Query-By-Example audio retrieval which would potentially speed up the interactive annotation process presented in Chapter 2. The method solves the situation where a machine's searching for promising regions in the audio is confused by overlapping sound in a query which leads to poor retrieval results during early stages of the interactive annotation.

The method utilizes users' vocal imitation. Users can improve the search results simply by imitating the sound events they do or do not want. It is a new approach for a user to provide an audio search system with additional audio examples as positive and negative feedback. The interaction is useful especially when prerecorded examples of each isolated sound event in a query are not available. It is often hard to find a recording that sounds the same as an isolated sound event in a query containing overlapping sound. To implement the interaction, I present ways of using an existing deep neural network model for generating audio embeddings to create a similarity measure for Query-By-Vocal imitation (QBV) search.

To evaluate the effectiveness of vocal imitation in content-based audio search, I created *Vocal Imitation Set*, a new crowd-sourced vocal imitation dataset. Vocal Imitation Set is the first dataset of vocal imitations that uses a widely-used ontology, AudioSet ontology [**29**] and it has more than double the number of imitations available in existing vocal imitation datasets [**17**].

My works in Chapter 3 have been presented at IEEE conference on Acoustics, Speech and Signal Processing (ICASSP) [**50**] and the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE) [**47**].

1.3.3. Contribution in Chapter 4

I present a new type of sound event labeling, *Point labeling* that can be obtained with less human effort as well as a new method to build a sound event detection system on point-labeled audio data. Point labels contain names of sound events at a single time point per sound event instance in a recording. A human annotator can indicate a sound occurred (e.g., by clicking a mouse button or hitting a key) anywhere within the area of the sound event. Therefore, point-labeling is much faster than *strong-labeling* (i.e., labeling with accurate time-boundaries of a sound event) and better than *weak-labeling* (i.e., labeling without any time information). I also present a method to automatically make point labels competitive with strong labels by bootstrapping from weak labels. It helps to build a SED system with point-labeled data that is comparable to one built on strongly labeled data.

I evaluate the efficacy of point labels for building a sound event detection system using the proposed training method. The experiment results show that a model trained on point labeled audio data is comparable to a model trained on data labeled with correct

time-boundaries of sound events (i.e., strongly labeled data). Therefore, I believe that the proposed methods will allow us to build a high-performance machine annotation system with much less human labeling cost.

My works in Chapter 4 have been presented at IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA) [**51**].

Supplementary materials for each chapter such as audio examples, demo videos, or codes for experiments are available at `https://github.com/bongjun/thesis`.

1.4. Broader Impact

The proposed methods are applicable in any field that needs to search and label sounds. Sound designers might want to search for sound effects of interest in a database or a long audio recording quickly. Ecologists need tools for labeling bird calls and singing in lengthy recordings. Language pathologists are also interested in audio searching tools to find sound events that would affect children's language development in an audio file where their everyday life is recorded.

The proposed work would also help people working on citizen science for sound object labeling such as the SONYC project [**5**] where crowd-workers are asked to annotate sound events in a recording to be used to build noise monitoring systems. The research in this dissertation on the new sound event labeling (i.e. point-labeling) and the proposed model training method will make it possible to generate enough labeled data for a statistical machine learner to be trained in cases where it is currently prohibitive to label enough data by hand. This will eventually increase the range of sound-objects that could be automatically identified by AI systems.

In addition to audio annotation, the proposed methods can be applied to video data. Researchers who study animal behavior might need to annotate video data quickly to build a system for automatically computing quantitative measures of animal behavior [**43**]. It is also very time-consuming to label video scenes to train a machine learning model for computer vision tasks such as visual event detection [**115, 74**]. The proposed model training method on point-labeled data will reduce such annotation efforts.

Finally, the proposed point-labeling is a new audio labeling method that has not been addressed in any of sound event detection literature. In this dissertation, I provide the evidence that models trained on point-label data outperform models on weakly-labeled data. Therefore, I expect that my works on point labeling will open new opportunities to researchers who have been working on an audio detection model with weak supervision.

CHAPTER 2

A human-in-the-loop system for sound event annotation

2.1. Introduction

In this chapter, I address sound event labeling tasks that fall in a middle ground: there is too much audio to be practically labeled by hand, yet there are too few training examples to train an accurate statistical model. I want to develop an efficient way to achieve human-level labeling accuracy with much less human effort than is typical for manual audio annotation. I note that my primary goal in this chapter is to speed the labeling task at hand, rather than to train a generalized machine learning model for later use on different data.

To achieve the goal, I apply a human-in-the-loop approach to sound event detection and annotation. The idea is to engage users in an interactive process [**102**] to collaboratively label the audio with the machine. Human-in-the-loop machine learning is a technique that has received attention recently as one approach to resolving limits of fully automated systems. It has been applied in many areas, such as image retrieval and identification systems [**102, 12, 110**], image foreground extraction [**90**], image object labeling systems [**92**], biomedical image recognition [**116**], natural language processing [**97**], Network Alarm Triage [**2**], interactive visualization for machine learning [**101**], musical performance [**26, 25**], and audio source separation [**78, 11, 24**].

In this chapter, I present a new human-in-the-loop sound search method to speed up sound event annotation in a recording. The system directs the user's attention to the most promising regions of audio for labeling. The user labels these regions and gives the system

feedback by labeling and adjusting region boundaries. The system learns from this feedback and updates future recommendations for high-interest regions.

To assess the effectiveness of the proposed method for sound event annotation task, I build an audio annotation tool called *Interactive Sound Event Detector (I-SED)* and perform a human subject study with potential users of the tool. In the experiments, I evaluate how much the proposed tool speeds up sound event annotation tasks. The results show that my approach helped the experimental participants label target sound events twice as fast as labeling them manually. I also present a method to quantitatively assess the system's retrieval performance and the interaction overhead separately, as these are two key factors that determine the performance of a human-in-the-loop system. The analysis shows that an ideal interface that does not have interaction overhead at all could speed labeling by as much as a factor of four.

The contributions of my works in this chapter are the following:

- A new human-in-the-loop sound search method which greatly reduces sound event annotation time
- The first general-purpose sound annotation interface where interactive machine learning is applied
- Qualitative and quantitative evaluation of the proposed interface.
- Evaluating the interactive overhead caused by the human-machine collaboration.

2.2. Related works

2.2.1. A human-in-the-loop system for multimedia retrieval and annotation

A common approach to labeling large amounts of multimedia data is through crowd-sourcing [**15, 106, 108**] where a small amount of data to label is assigned to a crowd-worker with

web-based annotation interfaces. Even though crowd-sourcing annotation is a great way to collect large scale labeled audio data quickly, it is not appropriate in a situation where the audio data should be annotated very accurately by domain experts or must not be distributed in public, such as audio recordings of patients for clinical purposes or costumer's private conversation with AI agent (e.g., Apple's Siri or Amazon's Alexa). In this chapter of my dissertation, I focus on the case where a very accurate annotation is required and crowd-sourcing is not an appropriate solution.

Interactive learning frameworks that depend on users' relevance feedback have been actively researched in image retrieval. *CueFlik* [**27**] is a web image search application that allows users to create and adjust rules for concepts (e.g. portraits of people) by providing the machine with positive and negative examples. The user feedback iteratively updates the rules to obtain more accurate image search results. Their interactive approach is aimed at training the best classifier to retrieve images relevant to a query. My goal is to completely label the audio easily and quickly.

In general, interactive image annotation/retrieval systems provide a user interface where a user can look at sets of images and give the system feedback by clicking the images [**1**] or selecting sub-regions of the images [**12, 105**]. The interface design for my sound detection tool focuses on directing the user's attention to promising sub-sections (i.e. the machine's recommendations) of a long audio track for labeling.

2.2.2. Existing sound annotation tools

Several audio editing applications such as *Audacity* [**64**] and *Sonic Visualizer* [**14**] provide an annotation environment where a user manually selects a sub-section of an audio track and labels it. *Audio-annotator*[**18**] and *BAT*[**71**] is a web-based audio annotation interface

for crowd-souring which also provides the manual annotation environment. *Sonic Visualizer*, *AudioSculpt* [**8**], *Audio Brush* [**10**], *ASAnnoatation* [**9**], and Praat [**7**] also provide low-level feature information (e.g. pitch content, repeated structure labeling) by using audio signal processing techniques, but they do not use high-level semantic labels (e.g. Bob's voice) and do not allow user-defined labels.

TotalRecall [**56**] is a semi-automatic multimedia annotation tool. It automatically detects speech regions on an audio track (speech or non-speech) for audio segments. It helps a user to find speech sections of an audio track easily, but is hard-coded to find speech and cannot be on-the-fly re-purposed for detecting other kinds of events.

SoundsLike [**32**] is a tool to detect user-selected sound events in a movie. It provides a similarity graph that visualizes which audio segments are similar to the user-selected segment as an aid for easier navigation. The system does not update its similarity estimates based on user feedback. Therefore, if the system thinks two segments are similar and the user doesn't, there is no way to correct the system. They also did not evaluate how much the similarity graph helps the annotation process. Finally, the interface does not provide any machine prediction to speed up the labeling process.

Gulluni [**33**] suggested an interactive approach to analyze electro-acoustic music by interactive machine/human labeling of sound objects within a music track. While Gulluni's system does not allow a user to change the boundaries of segmented regions, my system utilizes boundary adjustment of segments as user feedback to retrain a model. Moreover, their approach uses clustering techniques that require a user to listen to the audio multiple times to determine the best segmentation level. Multiple listenings can be problematic for long audio files (hours long). They also did not conduct human subject studies to evaluate the effectiveness of the system and only tested their system in simulation. In contrast, I

performed a human-subject study where participants used the proposed tool and a manual editor to label audio, letting us observe the effectiveness of my approach for speeding labeling.

Nakano et al. [**76**] presented an interactive annotation interface for music. It helps a user to label pitch contour of a vocal sound in music quickly. Once a user annotates pitch contour of a vocal sound in a section of music, the system shows pitch contour estimation in other sections that are similar to the user-edited section. Then, the user can accept or edit the machine's suggestion. Similar sections are detected by using repeated structure of music. While the annotation in their interface is performed by human-machine interaction, their method is fundamentally different from my work where a machine learning model for detecting target sound events gets improved during rounds of the human-machine collaboration.

2.3. Interactive sound event detection and annotation

In this section, I describe my sound event annotation system that lets a single user greatly reduce the time required to label audio that is tediously long for a human (e.g. 20 hours), has target sounds that are sparsely distributed throughout a long duration recording (10% or less of the audio contains the target), and has too few prior labeled examples (e.g. one) to train a state-of-the-art machine audio labeling system.

2.3.1. System overview

Figure 2.1 shows how the proposed system works with the user to label target sound events. First, a user uploads an audio track into the system and provides an example of the kind of sound they seek (e.g. someone knocking on a door). This can be done either by selecting a region on the audio track containing a good example sound or by uploading a short example

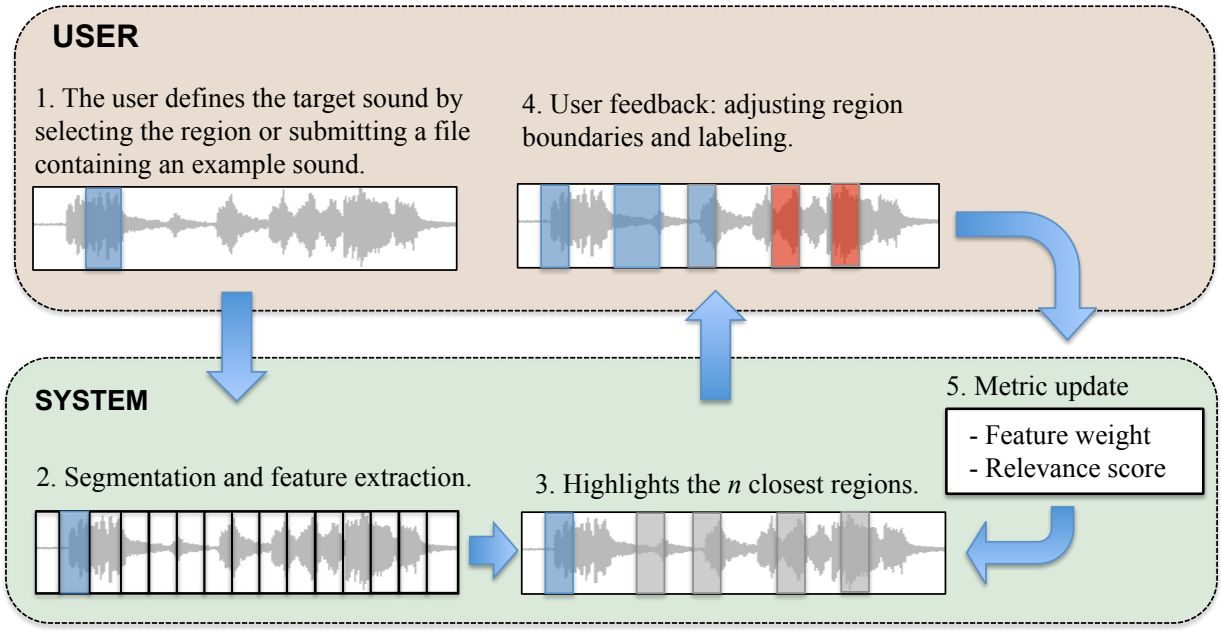


Figure 2.1. System overview of the human-in-the-loop sound event annotation.

audio file containing an example target sound (e.g. someone knocking on a door). Note that the goal is not to find exact copies of the target sound in the audio file to be labeled. The goal is to find other sounds of the same category (e.g. other door knocking sounds) in the audio file.

The system segments the track into small regions whose length is the same as the initial example and measures features of the audio file (see Section 2.3.2). It then finds the n regions with features most similar to the example and directs user attention to them by showing them as candidates. The user labels the candidate regions as positive or negative (see Section 2.3.4), and adjusts the start/stop times of positive examples. Based on this user feedback, the importance of audio features is re-weighted to move positive examples closer together and further from the negative examples. Given this new feature space, the system selects a new set of n relevant regions (see Section 2.3.3) for the user to evaluate. This

process of selecting candidate regions for human evaluation is repeated for some number of rounds. As more examples are labeled by the user and the features are re-weighted every round, the system's ability to suggest good regions improves.

Active Learning [**96**] refers to the case where the learner selects the examples to learn from, rather than passively receiving examples chosen by the teacher. This interaction between the user and the system can be thought of as a kind of active learning where the system is learning the feature weights for audio examples by presenting the unlabeled segments it currently estimates to be most similar to the set of positively labeled segments. The system shows the user the top n results and not the most ambiguous examples because our purpose is to help the user complete the annotation quickly by directing their attention to high-likelihood regions, not to train a machine learning model to fine-tune a decision boundary. When one has a long audio file and only a few target examples, showing the top n results lets users label them more quickly. This is a better strategy to speed up the labeling task at hand.

2.3.2. Segmentation and feature extraction

Once a user provides the initial example to the system (e.g. a 3-second region containing a bird call), the entire track is split into segments whose length is the same as the length of the initial example (e.g. 3 seconds). In the initial phase, all segments have the same length. However, once the user starts labeling the suggested regions, the length of user-labeled segments will vary, because the user is allowed to adjust boundaries of the regions. Given the possibility of varied-length segments, we need a way to measure the distance between segments, regardless of segment length.

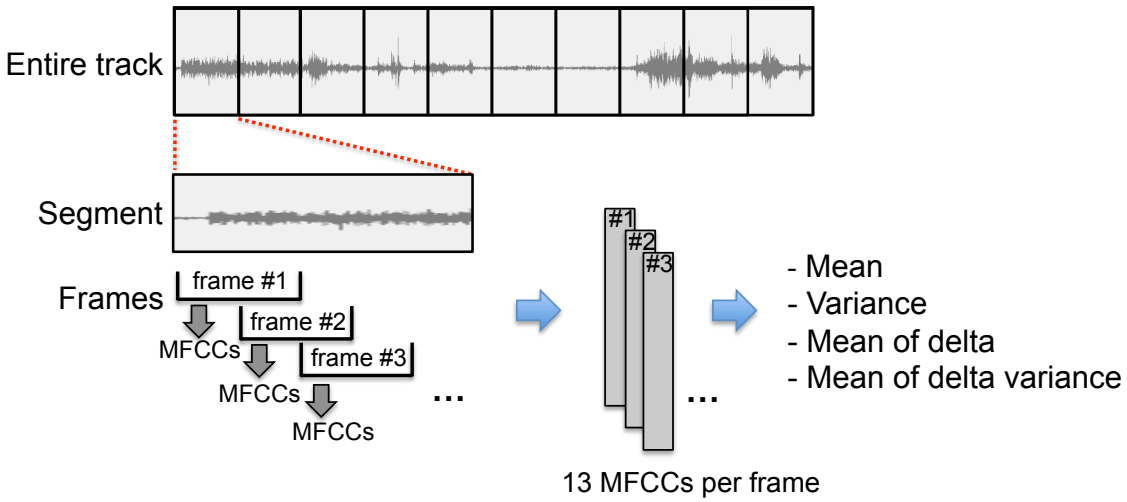


Figure 2.2. Audio feature extraction for all segments of an audio file. MFCCs are extracted frame-wise and pooled over each segment using the means and variances of both the instantaneous and delta values.

To measure distances between the segments including the initial example, audio features are extracted over each fixed-length segment. Our system extracts the first 13 Mel Frequency Cepstral Coefficients (MFCCs). MFCCs are widely used in a variety of sound recognition tasks [**82**]. As shown in Figure 2.2, each segment is split into a sequence of short frames (e.g. a frame-size of $90ms$ with 50% overlap between adjacent frames) and MFCCs are computed on each frame. The MFCC Features extracted frame-wise are pooled over each segment (e.g. 3 seconds) using mean and variance of instantaneous and delta values. The delta values are the difference between feature values of two consecutive frames. These represent basic temporal characteristics of the feature vectors in one segment. As a result, a 52-dimensional feature vector is built for each segment (13 MFCC averages, 13 MFCC variances, 13 MFCC

average delta, 13 MFCC average delta variance). Using this feature extraction method, varied length segments can be represented as a fixed-length of vector (i.e. 52 dimensions) so that distances between the segments are easily measured in the feature space.

2.3.3. Relevance score

In each round, the system measures the distance between each unlabeled segment and the set of positively labeled segments (initially, this set contains only the original target example). Using this distance, it ranks them by decreasing level of relevance, and presents the top n segments to a user. To compute the relevance score, I apply a simple nearest neighbor method [**31**]. The relevance score of an unlabeled audio segment s can be computed as

$$Rel(s) = \frac{d(s, s_n)}{d(s, s_n) + d(s, s_p)} \tag{2.1}$$

where s_p is the nearest positively-labeled segment to s and s_n is the nearest negatively-labeled segment to s. Function $d(a, b)$ is the weighted Euclidean distance between two segments in the feature space. When there is no negative segment (there is always at least one positive example, which is the initial query), the relevance score is computed as

$$Rel(s) = \frac{1}{d(s, s_p)} \text{ if } |neg| = 0, \tag{2.2}$$

where $|neg|$ means the number of negative segments.

To obtain a more accurate relevance score in each round, the system re-weights features using Fisher's criterion [**112**]. The weight of i^{th} feature is computed as

$$w(i) = \frac{(avg(f_i^p) - avg(f_i^n))^2}{std(f_i^p)^2 + std(f_i^n)^2} \tag{2.3}$$

where f_i^p and f_i^n are vectors whose elements are i^{th} feature values in the 52 dimensional (MFCC-based) feature vectors of all positive and negative examples respectively. $avg(x)$ and $std(x)$ indicate the mean and standard deviation of elements in a vector x.

As the *i-th* feature contributes more to better discrimination between positive and negative examples, its Fisher score will increase. The system re-weights each feature with the Fisher score based on the current labeled segments (positive and negative) in each round and the relevance scores (eq. (2.1)) over all segments are computed in the updated feature space. I expect that as more labeled segments are collected, the relevance score would become more accurate.

2.3.4. User relevance feedback

The system presents n segments to be labeled every round and the user adjusts segment boundaries and labels them. Labeling segments plays an important role as feedback for the future rounds because the machine's suggestions for each round depend on the user feedback in past rounds.

A user can provide two types of feedback to the system, as shown in Figure 2.3. One is to apply positive or negative labels to each candidate example. This type of feedback has been widely used in interactive image retrieval systems [**111**]. The other type is to adjust boundaries of the suggested region when the region does not cover the whole duration of a target sound event. This type of feedback is typically not used in document or image retrieval systems, but is useful for improving retrieval of regions of audio files.

The system automatically collects additional negative examples from the user's boundary adjustments. As shown in Figure 2.4, for example, suppose the user changes the position of the region (A) and labels it as positive. In this case, the system can obtain not only one

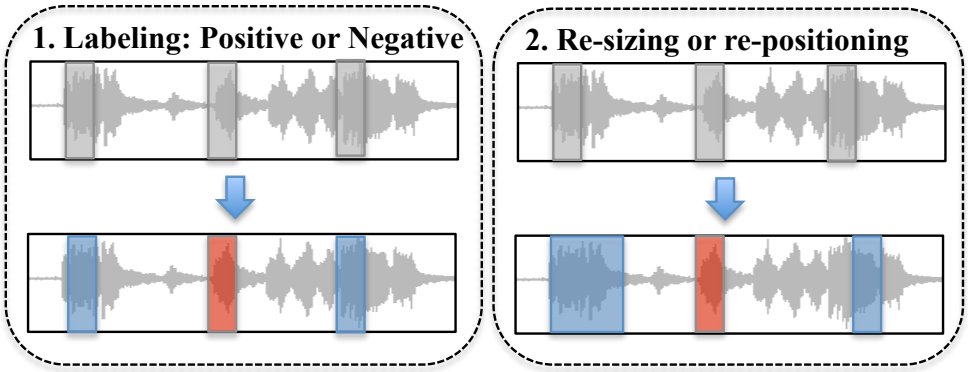


Figure 2.3. As feedback, a user labels regions positive(blue)/negative(red), or changes the time position and size.

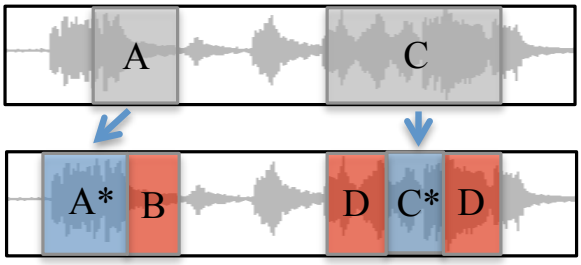


Figure 2.4. The region A and C (gray) are the machine's suggestions. A user listened to them, changed the temporal location of A and adjust boundaries of C, and labeled them positive (A* and C*). Then the system automatically labels the region B and D as negative since it is clear that they do not contain the target sound events.

positive example, but also one negative example which is the region (B) that the user did not select after listening to it. In the same way, adjusting boundaries of region (C) generates negative examples (D). This automatic negative labeling is beneficial in two ways: 1) A user implicitly labels more regions in one iteration, speeding interaction, and 2) Since our system presents the most relevant examples to a user every round, the pool of labeled examples tends to skew towards positive, which could make measuring relevance score problematic. Therefore, adding negative examples automatically helps in computing more accurate relevance scores of unlabeled examples.

2.4. Interface design and implementation

I implemented a web-based interactive annotator and Figure 2.5 shows its main workspace. Readers can watch the demo video and try the application out at `http://www.bongjunkim.com/ised/`. It consists of these main sections: *Navigation Map*, *Annotation Track*, *Listen and Label These*, and *Already Labeled*. The *Navigation Map* displays a waveform of an entire track and the currently labeled regions so that a user navigates and listens to them easily. The *Annotation Track* is a zoomed-in version of *Navigation Map* where a user can select or adjust regions and label them by clicking and dragging a mouse. The *Listen and Label These* displays the top n candidate regions identified by the machine. These are to be labeled by the user every round.

Figure 2.6 describes how a user performs the labeling task in each round. The user listens to new regions by clicking the items in the *Listen and Label These* section. If a region does not contain an example of the target sound class, the user labels it as negative by clicking the *Negative* button. If the region has the target sound, the user first adjusts boundaries of the region so that it fully covers one instance of the target sound class and labels it as positive by clicking the *Positive* button. The user labels all the regions in each round and clicks on the *Find Similar Regions* button to submit the feedback to the system and obtain a new set of candidate regions from the system.

The *Already Labeled* section shows regions labeled positive during past rounds. A user can listen to them by clicking each item in the list. Clicking on the *Export Results* button saves the annotation results to a text file containing start and stop times of all positive regions.

Figure 2.5. Screenshot of the interactive sound annotator.

2.5. Evaluation

I conducted a user study to validate the effectiveness of the proposed interactive annotation approach compared to manual annotation. The experiment seeks to answer the following questions:

- Which interface enables participants to label the given audio track faster?
- How accurately did participants label the target sound events using each interface?

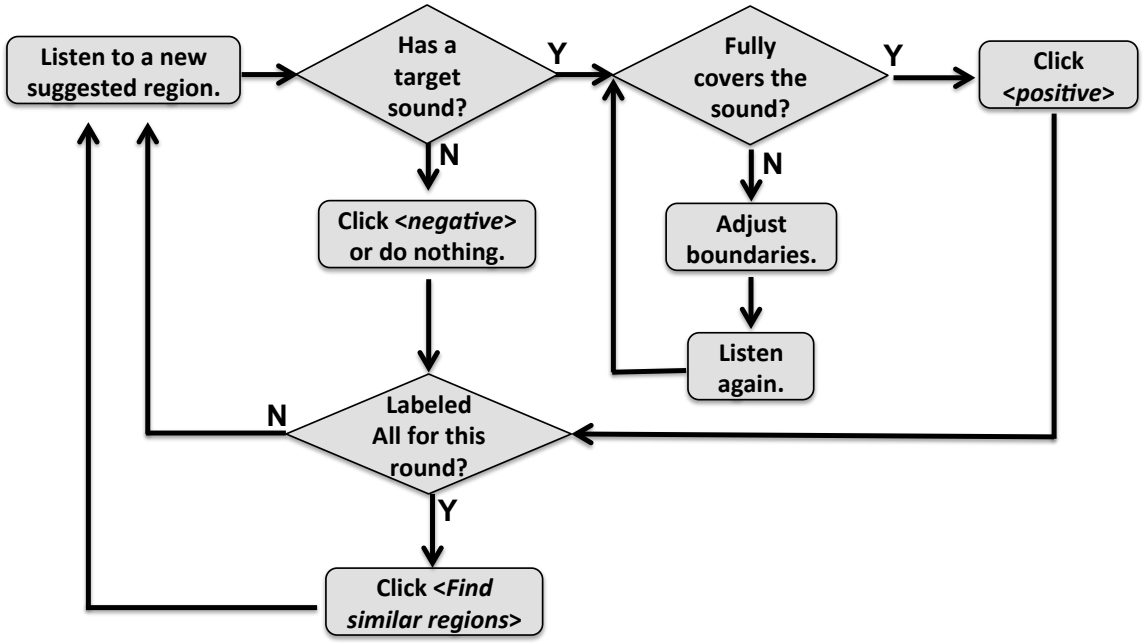


Figure 2.6. User interaction flowchart of the interactive annotator.

- How satisfied are participants with each interface?
- Do participants prefer the proposed interactive annotation to manual annotation?
- What user-interface overhead does the interactive annotation approach impose, compared to manual annotation?

2.5.1. The two interfaces compared in the study

The two interfaces I compared were the proposed interactive annotator and a manual annotator, similar to the standard interfaces currently used for manual annotation. In the interactive annotation tasks, a participant submits a file containing an initial target sound event to the system, and then the system presents the five most relevant regions to the user at each round. If the participant thinks a suggested region contains the target sound events, the participant labels it as positive. If the suggested region contains only some portion of

the target sound, the participant adjusts the position of the candidate regions and labels it as positive. If it does not contain the target sound at all, it is labeled as negative. The participant keeps labeling in this manner for the given amount of time in each task (15 minutes).

In the manual annotation tasks, participants use an identical interface to the interactive annotator, except for the removal of the recommendations from the system. They find target events by listening to the track sequentially (from the beginning to the end) or accessing any time position on the track. If they find the target sound event, they drag a mouse over the region containing the target sound. They keep finding as many sound events as they can for the given amount of time in a task (i.e. 15 minutes).

Both interfaces also provided a control for the user to adjust playback speed (1x, 2x, and 3x). Speeding playback is an alternative way to speed up the search that is commonly available in many multimedia players or annotators such as *Youtube*, *Quicktime Player*, or *Audacity*. It is also known that increasing up to double rate produces no significant loss in comprehension of speech, but higher playback rate produces a loss in comprehension [**77**].

To navigate an audio track, one might use a scrubbing function that is available in modern Digital Audio Workstaion (DAWs), where a user drags a cursor on a waveform and the audio samples the cursor is passing by are played. In the interfaces for this experiment, the scrubbing feature was not provided because it is mostly useful to find a precise spot where sound characteristics dramatically change (e.g. silence between noisy sections), and is not useful for recognizing relatively short sound events overlapping each other in a long audio track, therefore I did not include a scrubbing feature in either interface. By decreasing the chance a user makes mistake with the scrubbing feature and takes more time, I can more fairly evaluate the efficacy of the interactive annotation against the manual annotation.

2.5.2. The audio dataset

To evaluate our system, I used the dataset from the IEEE Audio and Acoustic Signal Processing Technical Committee challenge on Detection and Classification of Acoustic Scenes and Events (DCASE) [**99**]. The DCASE dataset is one of a few public datasets for computational analysis of sound events and scene analysis. Moreover, since it has been used in a public challenge, it is well-designed enough to test submitted algorithms properly in various situations. Using a public dataset also allows future systems to compare to our approach under the same conditions.

To generate testing tracks for this experiments, I chose files used for the Office Synthetic (OS) Event Detection Task of the DCASE challenge.[1] These consist of two-minute duration mono recordings of sequences containing overlapping acoustic events in an office environment (e.g. coughing, drawer, door knock, speech, etc.).

I anticipate 10 minutes as the minimal length of track where someone might wish to speed search. Therefore, I created two different 12 minute-long audio tracks by concatenating six short tracks in the DCASE dataset to create each track. Each track contains 11 different sound classes with 18 examples of each class on the track. All sound events are randomly distributed over a track. The two tracks, while containing similar sound events, order these events differently. This prevents learning ordering details of one track influencing performance on the other.

The sound classes in the two tracks include the following 11 sound events: *door knock, door slam, speech, human laughter, clearing throat, coughing, drawer, keyboard typing, keys, phone ringing, page turning.* The DCASE dataset provides audio files with three different

[1] `http://www.cs.tut.fi/sgn/arg/dcase2016/task-sound-event-detection-in-synthetic-audio`

levels (-6, 0 and 6 dB) of the average Signal-to-Noise Ratio (SNR) of events over the background texture. Readers can find more detailed information about the dataset in [**99**]. I chose audio files with -6 dB SNR (i.e. the target sound is 6 dB softer than intruding background noises) when generating the testing tracks. This reduces the chances that users could obtain additional information about the sound events by looking at the shape of waveform displayed on the screen, as the waveform is not noticeably larger when the target sound class is present if the target class is 6 dB softer than the background. It also makes the problem more challenging for the machine recommender system, as sounds softer than the background noise are more difficult for automated systems to detect.

The summed total time that all of the instances of the target class take up in a track is roughly 4% of the entire length of the track. The density of target events and distractor events are roughly 1.5 and 15 (events per min). The average inter-onset-intervals of the 18 examples of a sound class is 38 seconds. Since environmental recordings are usually long and have many different kinds of sound events happening in our everyday life, I believe that the generated audio tracks, which have sparsely distributed sound events of each class, are appropriate to evaluate the proposed sound annotator.

2.5.3. Participant recruiting

The target users of the proposed tool are people who need to find and label sound events within an audio track for their research activities. These groups of users include speech and language pathologists, who need to analyze relationships between children's language development and their listening environment by recording their everyday life. Another group of potential users are researchers who study machine listening, since building an automated sound event labeler with supervised machine learning usually requires a number of correctly

labeled audio examples of various sound classes as training data. Therefore, I recruited people who study speech and language development or machine learning in the audio domain. I also recruited people who have experienced audio editing or labeling software. Even if people in this group may not be the main target user group, I believe they are appropriate subjects to validate the efficacy of the proposed system against manual annotation, since they are familiar with critical listening to audio recordings.

To ensure participants capable of recognizing sound events for the experiments, I performed a hearing pre-test. Participants were given a labeled target sound event to listen to (e.g. speech) and 10 different sound events (3 speech events and 7 other events) were presented. They were asked to select all sound events that belong to the same label as the target sound (e.g. speech) and they had to correctly label all target sound events to pass the test. I performed the hearing test twice per subject with two different target sounds (i.e. speech and door knock). These were the target sound classes used in the actual annotation tasks. This let the listening test also implicitly train participants on the range of variation to expect for target sounds in the actual tasks.

I did not limit gender and age of participants as long as they belong to the target group mentioned above, but all recruited subjects were over 18, since I recruited people who have experienced at least college-level research activities related sound (e.g. speech and hearing, machine listening). Subjects' native language did not matter as long as they could understand the experimental instructions written in English. In total, 20 subjects who met all requirements were recruited and each subject performed two annotation tasks using two different interfaces. All sessions were conducted using a desktop computer and headphones in a quiet room.

2.5.4. Task procedure

Each subject participated in one session. One session included a hearing test, training on the interfaces, two annotation tasks (one on the proposed interface and one on the manual interface) and survey questions about their experience. Each session lasted about one hour.

To reduce the chances that the order of presentation of interfaces would influence the results, half of the participants tested the proposed interface first and the other half tested the manual interface first. As with interfaces, half of the participants were presented Task 1 (the first audio file) first, and half were presented Task 2 (the second audio file) first. As a result, 20 participants were divided into 4 groups so that task and interface order was balanced:

- Group1: Manual, Task 1 → Interactive , Task 2
- Group2: Manual, Task 2 → Interactive, Task 1
- Group3: Interactive, Task 1 → Manual, Task 2
- Group4: Interactive, Task 2 → Manual, Task 1

I selected two different sound types for users to detect: one from physical objects, *door knock* and the other from human voice, *human speech*. Thus, participants labeled *door knock* in Task 1 and *human speech* in Task 2. These sounds were selected to be quite different from each other, as human speech is complex, varied and harmonic, while door knocks are transient, percussive sounds. I hypothesized that there might be differences in how good the system recommendations would prove for these qualitative different sounds.

Each task includes a training session for a subject to learn how to label audio using each interface. In the training session, participants were given the exact same task as in the testing phase, except for the recording to be labeled. For training, I chose a two-minute long

recording from another DCASE challenge dataset. It contains 15 classes of sound events (including speech and door knock sound) recorded in an office environment. Participants were required to spend at least 4 minutes practicing the labeling task. If they wanted to practice more, they were allowed to practice the labeling task as long as desired. No participant, however, chose to spend more than 4 minutes on training. Participants were also allowed to ask any questions about the task and the interface during and after the training session.

For each task, participants were asked to find as many regions containing the target sound class (e.g. door knock) as they could within 15 minutes. I believe that 15 minutes are enough for a user to label a 12-minutes long audio track even when they listen to the entire track sequentially from the beginning to the end. For the interactive annotator, an example target sound file was provided for the user to submit to the system as the initial query, which is one of two ways of submitting an initial query to the interactive annotator as described in Section 2.3.1. The reason I chose the method is because I only wanted to measure the benefits of the interactive loop against the manual method. Time that users would spend finding the first query in audio to be labeled would vary depending on the position of the target sound on the audio track and how they search for it. The initial query given to users is not one of the 18 examples of the target classes on the audio track. Therefore, regardless of which interface (manual or interactive) was used, each participant was given 18 examples to label in each task.

After each task, the participant was asked to identify their level of agreement with the following statements:

- I had a clear understanding of the task.
- I understood how to use this interface to achieve the given goal.

- I was satisfied with using this interface.
- I was able to label target sound events easily.

To do this, the participant was given a slider for each question that was labeled as ranging from strongly disagree (O) to strongly agree (1). After the entire session (both sound labeling tasks) was done, participants were asked a set of questions comparing the two interfaces:

- Which interface was easier to use?
- Which interface was easier to learn?

Additionally, they were provided a free-form comment box where they could leave any feedback about interfaces or tasks.

2.5.5. Performance measures

The machine's role in the proposed system is to direct the human's attention to the most likely portions of the audio, not to determine whether something is a member of the target class or not. The human makes the final determination. One can think of this system as an attention model instead of a classifier. Therefore I sought to measure how quickly a user found regions containing target sound events within a recording. How the classification accuracy of a machine learning model changes over time is not my focus.

As participants labeled audio, the system recorded positions of labeled regions, whenever labeled regions were added or updated. Based on the recorded log, I evaluated a user's labeling performance by measuring the proportion of sound events correctly detected by a user as a function of time spent on the task (0 to 15 minutes). I considered a sound event correctly labeled if the temporal position of user-labeled region overlaps sufficiently with the temporal position of its ground truth with one-second tolerance.

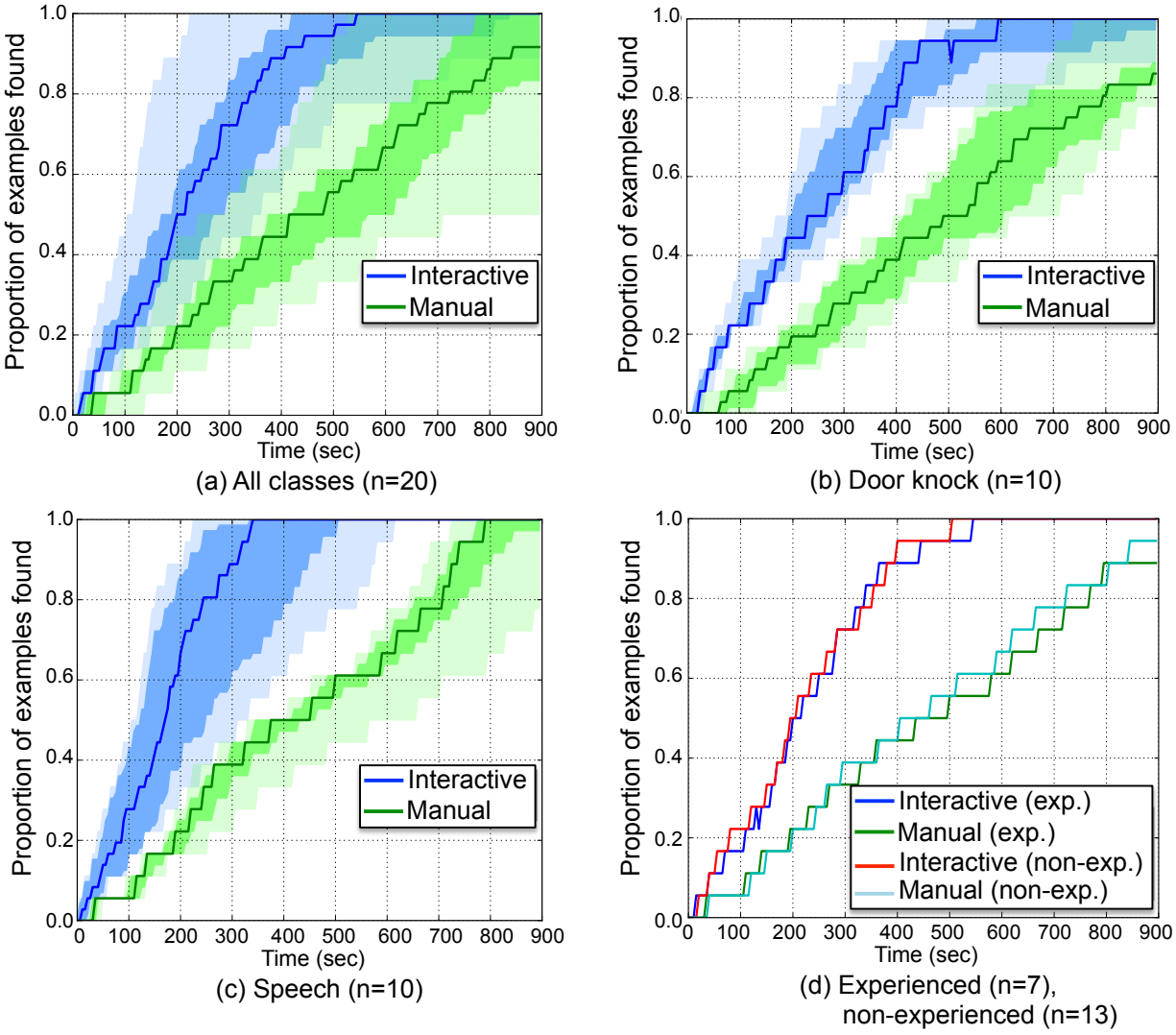


Figure 2.7. The proportion of target sound instances found as a function of time spent on the task (quantized every 5 seconds) using two different interfaces, the proposed interactive interface and the manual annotator. Here, N = 20, as each of 20 participants tried both interfaces in a session. (a)-(c): the proportion of examples found as a function of time spent labeling for all classes (a), knock only (b), and speech only (c). Lines indicate median, and dark and light bands of each color show 75th and 25th percentile. (d): the performance of two different participant groups: experienced and non-experienced. Lines indicate the median value of a pair of each user group and interface.

2.5.6. Results

Log data and answers to questionnaires were collected from 20 participants. The participants reported having an average of 42 months of experience using audio editing or labeling software. Moreover, 13 out of the 20 participants have used audio editing software longer than one year and 7 of them have prior experience using software to label sound events.

2.5.6.1. The absolute performance from log data. Each participant was asked to find as many target sounds as they could for 15 minutes within a 12 minute long recording using two different interfaces, manual and interactive annotator. Figure 2.7 shows how quickly the participants labeled target sound events over time.

Figure 2.7(a) shows the proportion of the target sound events detected by the 20 participants as a function of time they spent. Figure 2.7(b) and (c) shows the performance of the two different tasks respectively (labeling door knock and speech). Figure 2.7(d) shows the performance from two different groups of participants (7 people who have prior experience using software to label sound events and the other 13 people who do not have the experience). In Figure 2.7 (a), (b) and (c), lines indicate median, and dark and light bands of each color show 75th and 25th percentile. Lines in Figure 2.7 (d) indicates median value of a pair of each user group and interface. The median user of the proposed interface labeled all target examples within the time given.

Overall, it took an average of 517 seconds for participants to label all target sound events using the interactive annotator (15 rounds per user). This is roughly half the time it took the manual labelers. Specifically, Figure 2.7(a) shows that the interactive annotator lets one find about 80% of the target sound events in about 350 seconds. To achieve the same performance using the manual annotator, it takes 720 seconds (1-second tolerance).

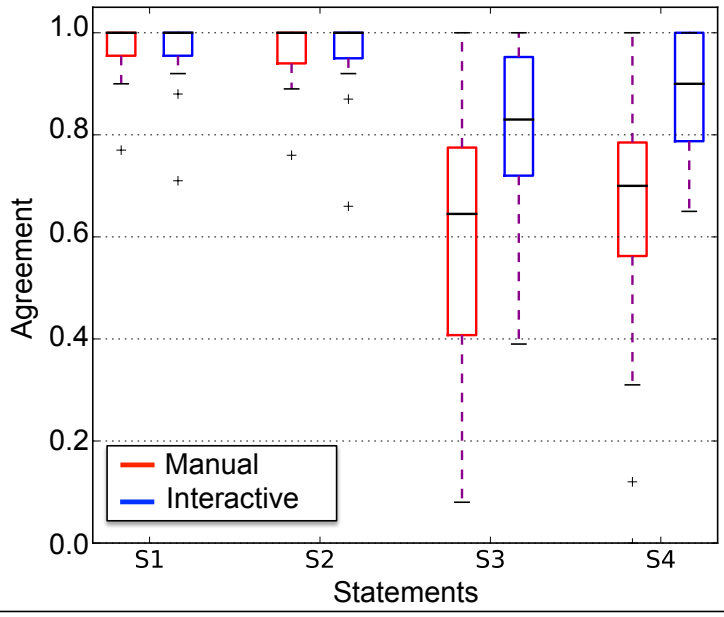
Therefore we can conclude that the interactive annotator helped participants find sound events roughly twice as fast as the manual annotator.

Figure 2.7(b) and (c) show which task took participants more time labeling each target sound. It turned out that labeling speech events took less time than labeling door knock events regardless of which interface they used. This difference probably came from the acoustic characteristics of the two sound events. One instance of knocking sound consists of multiple non-harmonic and short sound events (e.g. *knock-knock-knock*). So participants had to listen to the audio to find the exact start and end position of one instance of knocking sound event. On the other hand, recognizing the start and end position of speech is relatively easy.

I also examine the difference between participants who have prior experience using software to label sound events and participants who do not. As shown in Figure 2.7(d), there is no big difference between them in terms of how quickly they found the target sound events. We can conclude that the interactive annotation interface speeds labeling by roughly the same amount, regardless of a user's prior experience with labeling audio.

2.5.6.2. Self-reported performance. Figure 2.8 shows participants' level of agreement (0 to 1.0) with statements about their experience with each interface. I performed statistical tests to validate whether there is a significant difference between mean responses to questions about experiences with the interactive and manual annotator. I used Wilcoxon signed-rank since the data is not normally distributed.

The responses range from 0 (strongly disagree) to 1 (strongly agree). For statement 1 ("I had a clear understanding of the task." and statement 2 ("I understood how to use this interface to achieve the given goal."), there is no significant difference between the two interfaces. This means that most of the participants clearly understood how to use



S1: I had a clear understanding of the task.
S2: I understand how to use this interface to achieve the given goal.
S3: I was satisfied with using this interface.
S4: I was able to label target sound events easily.

Figure 2.8. User responses to questions about experience with each interface. Responses range from 0 (strongly disagree) to 1 (strongly agree). N=20 for each box plot.

both interfaces in the tasks. On the other hand, for statements 3 and 4, there is a significant difference between their mean responses. The mean responses to statement 3 ("I was satisfied using this interface.") are 0.588 (manual) and 0.785 (interactive), and they are significantly different ($p < 0.05$). The mean responses to statement 4 ("I was able to label target sound events easily.") are 0.667 (manual) and 0.878 (interactive) and they are also significantly different ($p < 0.005$). This indicates that participants felt the annotation task was easier using the interactive annotator.

I also compared responses from the two groups (7 experienced and 13 non-experienced participants). As shown in Figure 2.9, in the non-experienced group, the mean responses to

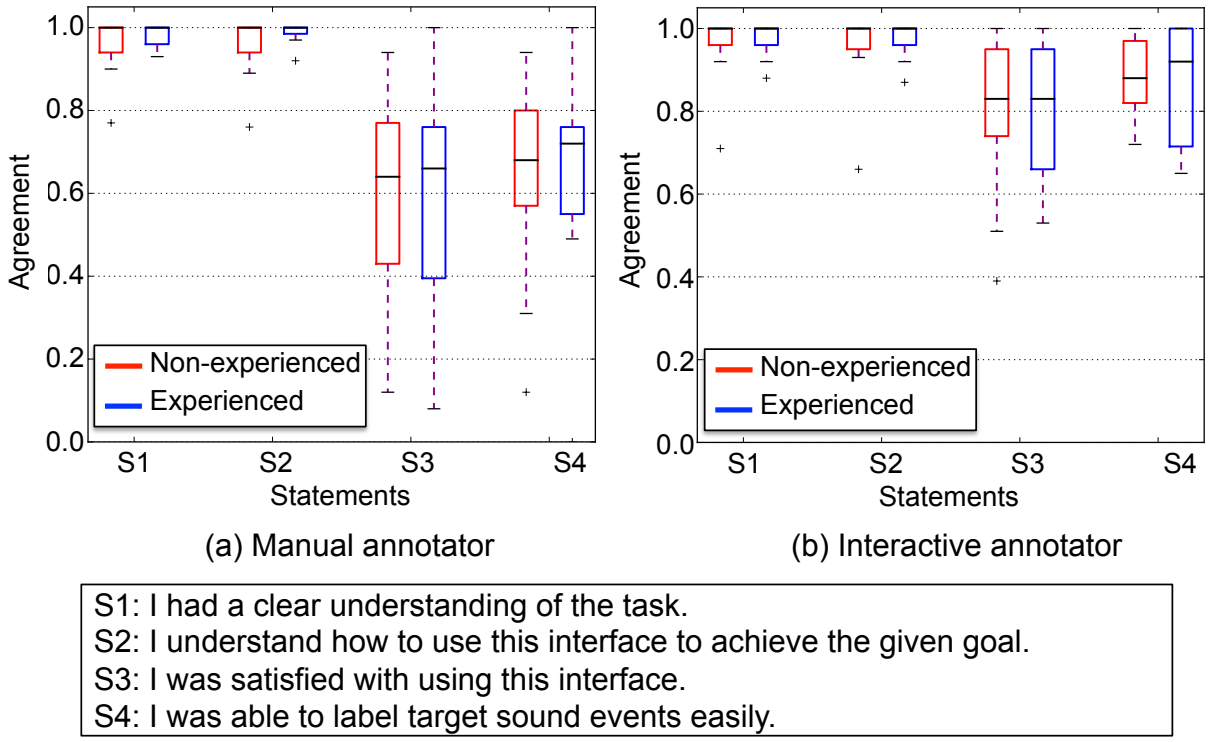


Figure 2.9. Comparison between survey responses from two different participant groups. (a): response about the manual annotator. (b): response about the interactive annotator Responses range from 0 (strongly disagree) to 1 (strongly agree). N=13 for non-experienced participants and N=7 for experienced participants.

statement 3 are 0.578 (manual) and 0.798 (interactive). The mean responses to the statement 4 are 0.657 (manual) and 0.889 (interactive). In the experienced group, the mean responses to the statement 3 are 0.579 (manual) and 0.797 (interactive). The mean responses to the statement 4 are 0.620 (manual) and 0.857 (interactive). Based on this analysis, we can conclude that the interactive annotator was preferred regardless of users' prior experience using labeling software.

2.5.6.3. Participants comments. Participants wrote comments about their experience with each interface in the free-form response box. Many people liked the effectiveness of the

recommendation feature of the interactive annotator, compared to the manual annotator. Representative quotes include:

"Using the manual annotator, I could not listen to the entire sound signal in the given time. It seems the only way of finding all the target moments is to listen to all the signal which includes a lot of irrelevant sound samples."

"The interactive annotator did a great job at quickly finding the relevant regions. I even had time to go back and adjust the boundaries of some of the regions."

"It is hard to say with 100% certainty that I was able to label the audio more accurately with the interactive annotator, but I spent the last 9-10 minutes not even hearing any human voices, which is assuring."

While their log data showed they could find sound events more quickly, some participants felt it is inefficient or boring because it started to suggest only irrelevant regions after all target events are labeled. Representative quotes about this issue include:

"The interactive one gave me many negative parts."

"I found that at the beginning of the test, the software found the target sound correctly, but at the end, accuracy decreased. I guess this happened since at the end there are not many door knock sounds left, so the software recommends any sound even though it is not like a door knock sound. Even in this case, the software should not recommend wrong target sounds."

"Towards the end of the task, I had established for myself that I had found all the target sounds in the last 5 minutes of the recording. While it was nice not to have to listen to the entire track, it was frustrating each time the system suggested a sound that was not the target sound."

From this set of feedback, it is obvious that, for future work the interactive annotator needs to have features where a user can be informed of the confidence of current recommendations or when to stop the iterations.

Some participants were not satisfied with the interactive annotator because it was sometimes difficult to figure out sound events only by listening to small snippets of audio, so they often had to make the suggested regions longer and listen to them. From this feedback, it seems that suggesting longer snippets of audio to users than actual prediction may be a good approach.

2.5.7. Interaction overhead

My interactive approach speeds up the labeling task by a factor of two in our user study. Can we make it even faster? There are two primary ways to speed labeling further. The first is to increase the accuracy of the system's predictions. The second is to lessen the overhead imposed on the user by interaction with the system. In this section, I address the issue of interaction overhead.

Recall that the interactive system presents users specific audio segments to listen to and label. Large portions of the audio were never listened to by users of the interactive system, as the system deemed them irrelevant for the labeling task. To quantify the amount of interaction overhead, I analyzed how long it took the average participant to label the audio they *did* listen to and compared it to how long it took them to label a similar duration of audio using the manual approach.

Figure 2.10 shows how long it took one participant to evaluate (i.e. labeling either positive or negative) a fixed amount of audio. Two different colored solid lines (red and blue) indicate the actual experimental data and the dotted lines are their linear regression

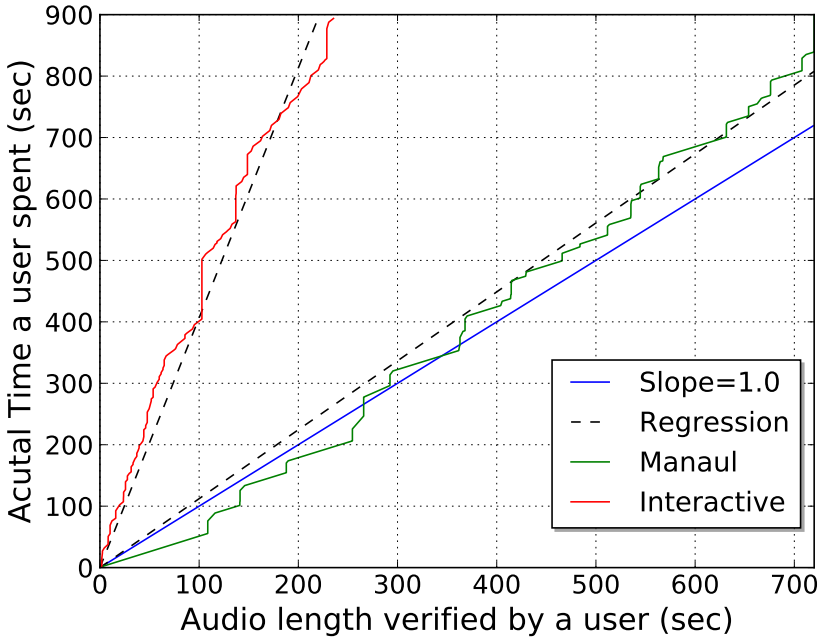


Figure 2.10. Total time a participant (one participant is selected as an example) spent on the task as a function of the total amount of audio verified/labeled. The interactive system requires more time to label a fixed amount of audio than the manual system. This interaction overhead partially offsets the speedup of the overall task the interactive system provides by having the users label the most promising segments of audio first.

lines. For this particular participant, the slopes of the two regression line are 4.07 for the interactive annotator and 1.22 for the manual annotator. The slope value 4.07 means that it takes 4.07 seconds to verify 1 second of audio.

To quantify the extra interaction overhead caused by the interactive approach, compared to the manual approach, I use the ratio of the two slopes (i.e. $overhead_i/overhead_m$). For the participant shown in Figure 2.10, this ratio is 3.62 to 1. I collected the interaction overhead values for 14 participants (I do not have interaction overhead data for the other 6 participants because we started collecting the log for computing the interaction overhead from the 7th participant). As shown in Figure 2.11, this ratio ranges from 2.95 to 7.69. The mean is 4.68 (median:4.22), which means that the interactive system has 4.68 (mean)

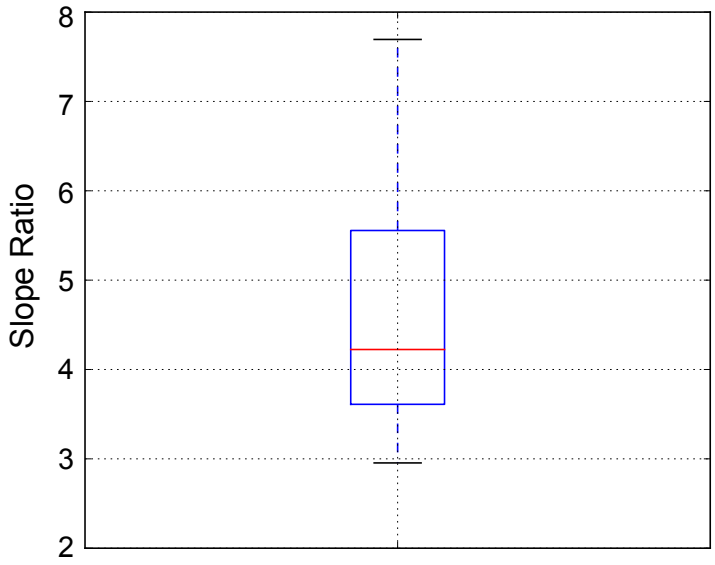


Figure 2.11. Overhead ratio (interactive/manual) for 14 participants (Median: 4.22, mean:4.68).

times more interaction overhead than the manual system. Despite this ratio, the interactive approach still doubles the speed at which users completed the up the overall annotation task, by having the user to label the most promising segments of audio first. It is, however, obvious that reducing the interaction overhead would improve the speed up even further. This way of quantifying interaction overhead is useful when various human-in-the-loop systems need to be compared, as it separates out the influence of the recommendation system and the interface for the user to do labeling.

2.5.8. Machine accuracy

In addition to reducing interaction overhead, increasing machine accuracy is also a key way to speed up labeling further. So it is important to measure the performance of the human-in-the-loop system without considering the interaction overhead. Assuming there is no interaction bottleneck in the interactive annotator, I simulated labeling tasks of a perfect

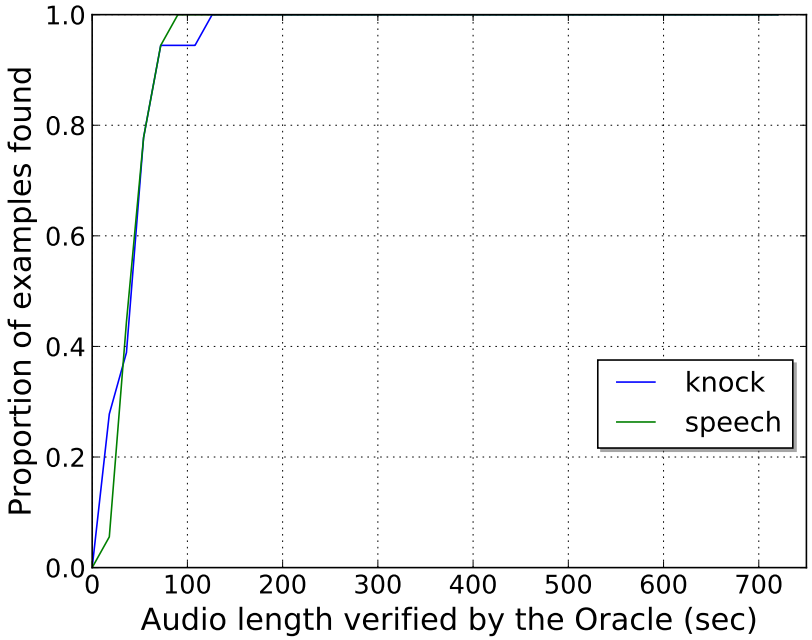


Figure 2.12. The user simulation. The proportion of examples found according to the proportion of audio the oracle evaluated. All sound events are detected by evaluating less than 126 seconds of the audio (17.5% of the audio).

simulated oracle. I measured what proportion of the audio would need to be verified to find all target sounds if the verification was done by the perfect oracle instead of a human without any interaction overhead. The user (i.e. the Oracle) submits one sound target event as the initial query, and then the system presents the 5 most relevant regions to the Oracle at each round. If each suggested region is matched to one in the ground truth with 200ms tolerance, the region is labeled as positive. If they are partially overlapping each other, the Oracle adjusts the position of the candidate regions and labels it as positive. If they are not overlapping at all, the Oracle labels it as negative. The Oracle keeps labeling until all sound events are labeled.

I ran this oracle simulation with the same audio and target sounds (i.e. speech and door knock) as in the user study. Figure 2.12 shows the proportion of detected sound events as a

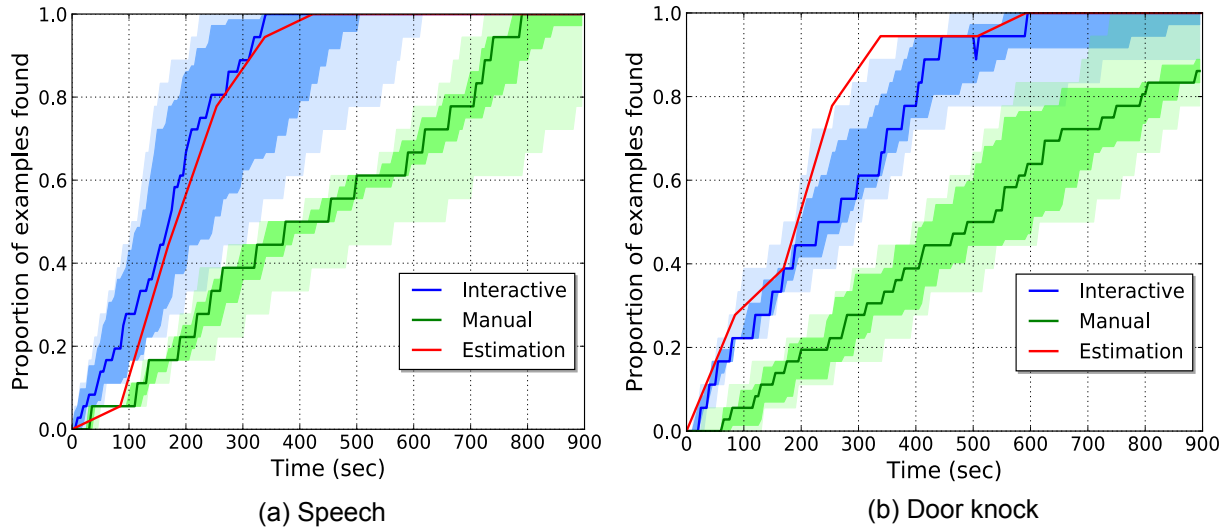


Figure 2.13. The machine-estimated user performance overlayed on the actual user data from Figure 2.7(b) and (c).

function of the proportion of audio verified by the Oracle. As shown in the figure, the oracle found all sound events of the two classes by evaluating less than 126 seconds of the audio (17.5% of the audio).

2.5.9. Estimating actual annotation time using user simulation and interaction overhead

The user simulation is a good way of measuring machine's accuracy. We can run the simulation on many different kinds of audio tracks easily, which costs a lot for the actual user study. However, this is not a proper method for evaluating the overall performance of my interface. We are not able to measure the actual time that a user would spend to achieve the goal. The results from the user simulation are always too optimistic because the interaction overhead is not considered in the simulation.

To solve this issue, I can use the quantified interaction overhead introduced in Section 2.5.7. In the previous section, I figured out that the interaction overhead of the interactive

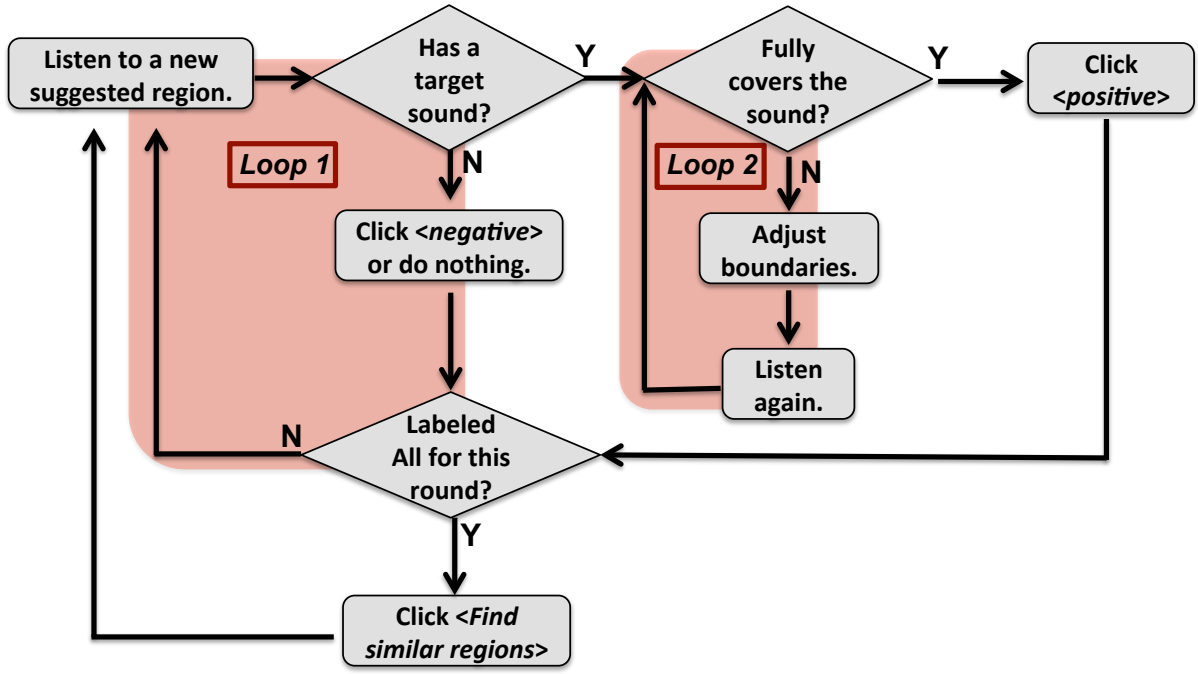


Figure 2.14. There are two bottlenecks (Loop 1 and Loop 2) in the interaction flowchart of the interactive annotation.

annotator is 4.68. I can apply this value to the simulation result by stretching the graph by 4.68 times on the x-axis. Figure 2.13 shows that applying this scaling factor makes the estimated graph very similar to the result from the actual user study.

2.6. Discussion

In this section, I discuss the limitations of the proposed system in terms of user interaction. As presented in Section 2.5.7, the interactive approach causes some amount of interaction overhead compared to a fully automated system or manual labeling. One piece of future work will be to redesign the interface to reduce the interaction overhead. I figured out there are two bottlenecks in the interaction of my tool by analyzing the log of users' behavior and their comments from the survey. Figure 2.14 highlights the two closed-loops

in the interaction flowchart presented in Section 2.4. In the experiment, users spent most of their time in these two loops during the annotation task.

Loop 1 is a process where a user verifies negative examples. In the case that the length of the initial query (i.e. a target sound example) is short and target sounds are sparse in the audio (e.g. 5% of the audio), the user loops the process many times to verify negative examples. In manual annotation, the process that verifies negative examples takes just as much time as it takes to listen to the audio. In my annotation tool, a user has to click the short regions to listen to them at most 5 times per round. In the experiment, it took an average of 15 rounds for participants to label all target sound events. Based on this analysis, one possible design choice for the future work would be to let a user listen to all 5 suggested regions just by clicking a button once, instead of selecting each of selected regions to listen to them. It would help users verify negative regions quickly.

In the *Loop 2*, users keep adjusting the boundaries of the suggested region while repeatedly listening to it. Some participants commented that even if a suggested region covers one instance of the target sound, they had to listen to the audio before and after the suggested region to understand the context and make sure the start and end of the selected region are correctly positioned. In other words, verifying if there is no target sound around the suggested region causes one or more unnecessary listening. Based on this feedback, one solution to this issue for the next version of my tool would be to present a longer region to a user than the actual length of the initial target sound.

Implementing these changes on the two bottlenecks in the interaction loop, the *Loop 1* and *Loop 2* should reduce the interaction overhead, allowing an even greater speedup of the annotation process.

2.7. Conclusions

This chapter of my dissertation presented a new human-in-the-loop sound search method to speed up human annotation of a long audio recording. I built the first general-purpose sound labeling interface, I-SED where an interactive machine learning approach is applied to sound event annotation. It helps a user quickly label target sound events in a long audio file. I performed a human-subject study to evaluate its effectiveness. The result showed that the proposed system lets users find sparsely-distributed target sounds roughly twice as fast as manually labeling the target sounds. The survey response and free-form comments showed that most participants were more satisfied with the interactive annotator against the manual annotator. I expect I-SED to facilitate audio data collection and improve how people understand sound scenes.

I also presented methods to measure interaction overhead and machine accuracy of the proposed system. Since reducing the interaction overhead and increasing the machine accuracy are two primary ways to speed labeling further, it is important to measure them separately to evaluate the interface.

Finally, I discussed the limitations of the user interaction design of the proposed tool. The interactive approach causes some amount of interaction overhead compared to a fully automated system or manual labeling. I introduced two bottlenecks in the interaction of my tool I found by analyzing the log of users' behavior and their comments from the survey. Therefore, a possible future direction of this work is to redesign the interface to reduce the interaction bottlenecks.

CHAPTER 3

Improving sound search using vocal imitation

3.1. Introduction

Imagine a human annotator is labeling a recording using I-SED, presented in Chapter 2. The annotator found an interesting sound event in the recording, selected the event as an initial query, and started the interactive search process to find more regions containing sound events that are similar to the query. Once the query is submitted to the system, it segments the audio recording to be searched into short chunks (e.g. a few seconds), ranks the segmented audio of the recording by content similarity to the query and returns the n most likely segments of the audio for the annotator to evaluate. This initial search process is an example of Query-by-Example (QBE) audio search [**37, 41, 59, 114, 48, 49, 119, 70, 16**], which lets a user provide an audio example (e.g. a recording of a dog bark) as a query to find the desired sound (e.g. similar recordings of dogs barking). A QBE audio search system measures the similarity between a query and recordings in a database and returns a set of recordings sorted by the similarity.

The performance of QBE audio search depends on the quality of a user query (i.e., audio examples). If the query does not provide the right information to sufficiently narrow the search, the target sound(s) will not be top-ranked in the retrieval results. This could potentially slow down the interactive annotation process described above because the search system will fail to find any sound events to be labeled as positive by a human annotator during early rounds of the human-machine interaction loop. Therefore, it is very important

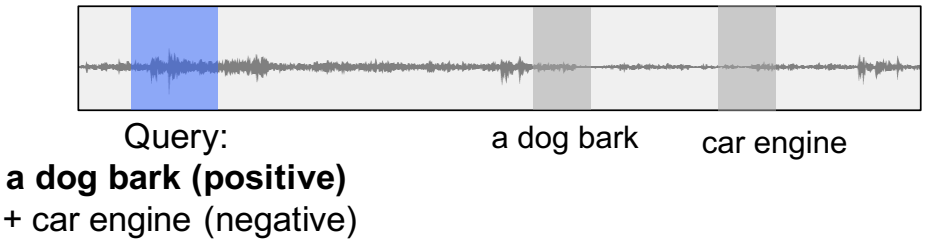


Figure 3.1. An example of the problem on a query containing overlapping sound events. Given such a query, the retrieval system does not know which portion of the query is relevant to a sound event that a user wants to search.

for a user to provide the system with audio examples appropriately representing sound events for which the user is searching so that the system successfully returns relevant sound events to a user.

In this chapter, I focus on how to improve QBE audio search when an initial search result is not satisfactory (e.g., no target sound is highly-ranked in the search results) due to the low quality of a query. If a query is an audio file recorded of an actual sound scene, it often contains multiple sound events overlapping each other. For example, a birdsong recorded in a natural setting is often overlapped with other interspersed sounds in the recording such as other bird species, dog barking, or a lawn mower. If a query contains such overlapping sound events (i.e. positive and negative sounds), how does a search system know which portion of the query is the one to focus on? Simply using a recording containing overlapping sound events as a search key would make the system get confused about what constitutes the positive sound (Figure 3.1). The retrieval results would get even worse when a positive sound in a query is not the most prominent event in the overlapping region.

Figure 3.2 shows magnitude spectrograms of humans coughing, laughing, and a mixture of the two. Figure 3.2(a) and (b) are isolated sounds of human coughing and laughing. If a query to a search system is one of these isolated sound events, the system would capture a

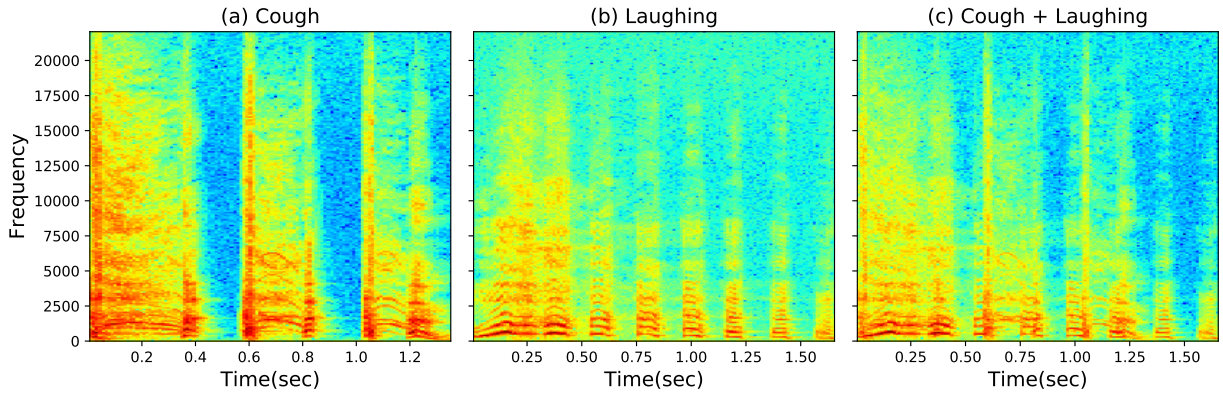


Figure 3.2. Magnitude spectrograms of humans coughing (a), laughing (b), and a mixture of the two.

unique pattern of each sound class and return sound examples similar to the query. However, if a query is mixtures of multiple sound sources including positive and negative sounds as shown in Figure 3.2 (c), it is hard for machines to know which part of the recording is related to the positive sound. The machine may even interpret the combination of sounds as the sound event that does not belong to either the coughing or laughing classes. Thus, it would probably lead to poor search results.

One way of overcoming the issue is to let a user specify the desired sound event in the mixture by selecting sub-regions of a magnitude spectrogram with a visual editing interface and separate the source from the mixture [**11**]. While visually selecting parts of interest in a spectrogram is a good way of specifying a positive sound, a user might find it difficult to know exactly which time-frequency bins of the spectrogram are related to the positive sound in a mixture. As shown in Figure 3.2 (c), overlapping sound events make it hard for a user to select only a single sound event from them because different sound events share many of the same time-frequency bins in the spectrogram. Figure 3.3 shows an example of the ideal selection of human laughing sound (dark red) from a mixture of laughing and coughing sound events, which is hard to be obtained by a user's manual selection. Even if it could be

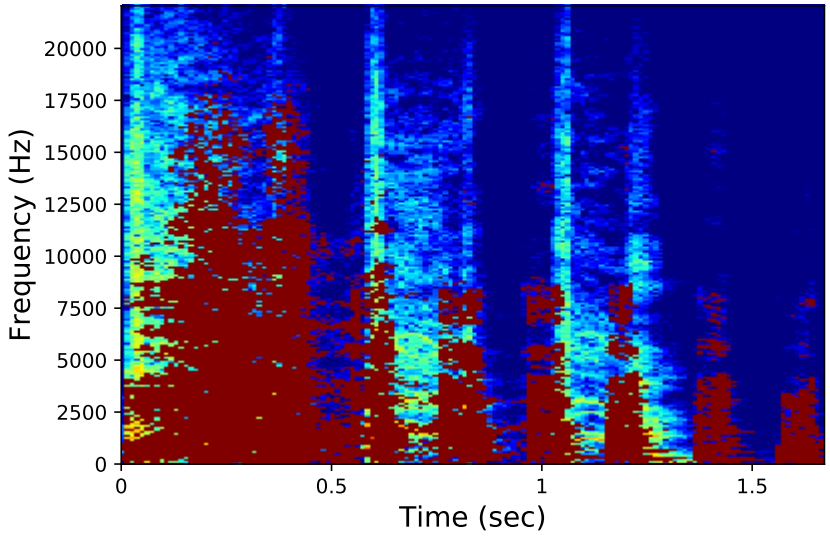


Figure 3.3. Magnitude spectrograms of humans coughing and laughing. The region associated with human laughing are colored in dark red.

possible, it would take a very long time to manually select the regions of the spectrogram belonging to one of the sounds. This would need to be done for every single query where there is overlap.

Another way of improving search accuracy, given a poor initial result, is to get user feedback indicating the relevance or irrelevance of a retrieved item which is known as Relevance Feedback (RF) [**91, 44, 66**]. For example, given a search result (i.e., a list of sound events sorted by the similarity with a query), a user labels recordings returned by the search system as positive or negative. Then, the label information (i.e., relevance feedback) is used to improve the search results. However, if none of the highly-ranked sound events in an initial search result is relevant to a query and there is no sound event to be labeled as positive, then improving the search system by user's relevance feedback on returned items might not be effective. Moreover, if a recording returned by the system contains overlapping sound events,

adding a positive or negative label to the recording does not make it clear what aspect of the audio is positive or negative.

My approach to improving QBE audio retrieval in the case of an example containing multiple overlapped sounds is to let a user provide the system with a new sound example by imitating sound events of interest in a query recording of an audio scene (e.g., adding a vocal imitation of a query recording of a 'dog barking' recording, like the one in Figure 3.1). Even when a query contains multiple overlapping sound events, if a user is able to identify isolated sound events and imitate what they do or do not want, it is equivalent to providing additional positive or negative examples to the search system.

Vocally imitating sounds is a simple and quick way to describe sound events. It is a very effective method to communicate an audio concept between people [**63, 62**]. Vocal imitations also have obtained attention as an input modality in human-computer interaction. For example, vocal imitations have been used for sound synthesis and design [**87, 39, 16, 70**]. They also have been used as a search key for melody search in a music database [**30, 41**] and environmental sound retrieval [**117, 6, 89**].

In my work, I use a user's vocal imitation as additional information to differentiate a positive sound from a negative sound in a query containing both of them for audio search. Even if a user's vocal imitation does not sound identical to the original recording, it has been shown that vocal imitations represent key characteristics of the original sound event and are recognizable by audio search systems [**17, 120**]. Vocal imitations of a sound event can quickly (seconds) highlight what aspect of a query recording is positive or negative, which is impossible to achieve with a simple text label for the whole file (e.g. positive or negative), and very slow (minutes or hours) with text annotation of subregions of the spectrogram.

Therefore, a user can give the system more detailed information about sound events in the query by imitating them to improve the retrieval results.

3.1.1. Contributions

My contributions of this chapter are the followings:

(1) I present a new feature extractor to generate audio embedding to measure the similarity between a vocal imitation and an actual sound event, which is necessary to utilize vocal imitations as additional labeled sound examples for search. I show that the feature extractor outperforms other existing methods by performing an experiment on vocal imitation-based audio retrieval.

(2) I propose a new way of updating query-by-example search results using user's vocal imitations of positive and negative sound events in a query recording containing overlapping sound events. A user can simply provides vocal imitations to illustrate what they do or do *not* want in a query. My work is the first system that uses vocal imitations to update a query and improve QBE search. Especially, none of prior works has focused on imitating what they do *not* want. This chapter is the first study to show the effectiveness of negative vocal imitations for audio search.

(3) I created *Vocal Imitation Set*, a largest crowd-sourced vocal imitation dataset to evaluate the effectiveness of vocal imitations on improving query-by-example audio search. Vocal Imitation Set is the first dataset of vocal imitations that uses a widely-used ontology, AudioSet ontology [**29**] and it has more than double the number of imitations available in existing vocal imitation datasets [**17**].

(4) The experiments show that not only imitating what they want (positive imitation), but also imitating what they do *not* (negative imitation) helps a search system improve the retrieval result.

3.2. Method

3.2.1. Feature extraction and similarity measure

Feature extraction plays a very important role in building a content-based audio retrieval system. Raw recordings including a query and items in a database need to be converted into feature vectors so that the system can find similar items to the query in the feature space. High-performing features that capture acoustic characteristics of various classes of sound events map recordings into the lower dimensional feature space in a way that similar sound events are grouped together in the feature space. Then the system can find relevant recordings by using a simple similarity measure such as cosine similarity (see Figure 3.4).

Recently, deep learning models have been successfully applied to various audio classification tasks and many of them use convolutional layers as a feature extraction part of the model [**38, 54, 20, 55, 83, 19, 68, 57**]. These models consist of convolutional neural networks (CNN) as a base model for audio feature extraction followed by various types of networks for their specific tasks. Once these models are trained on a large dataset, the trained convolutional layers of the models can be reused as a feature extractor of other models [**42, 21, 75, 40, 4, 57**]. This is very useful when a deep learning model needs to be trained on limited data for a specific task. A pre-trained CNN model that captures general audio features helps to reduce the number of trainable parameters of a model.

In my work, I leverage the embedding created from a large set (millions) of audio examples to build a general similarity measure for query-by-example audio retrieval. I utilize the

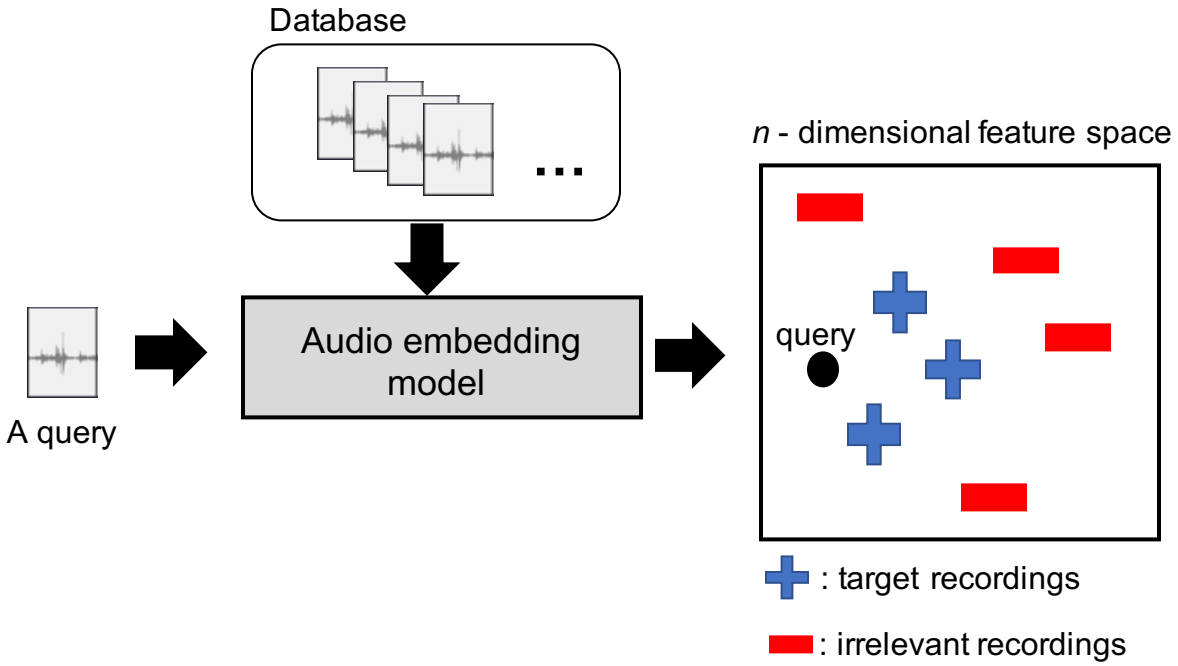


Figure 3.4. An audio embedding model maps recordings into the feature space in a way that similar sound events are grouped together. This makes it easier to find similar sound events to a query using a simple distance or similarity measure such as euclidean distance or cosine similarity

VGGish model which is a publicly-available [1] audio embedding model that has been created by training the modified VGGNet [**98**]. The VGGish model was trained on the audio from 8 million YouTube videos to distinguish 3,000 sound classes. I chose the VGGish model because it has been successfully used in many prior works to generate input features of audio to train a model on limited data for environmental sound classification [**40, 51, 46, 45**]. Figure 3.5 shows the architecture of the VGGish model. It consists of 6 convolutional layers, followed by 3 fully-connected layers. It takes a 1-second segment of an audio recording and output 128-dimensional feature embedding.

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

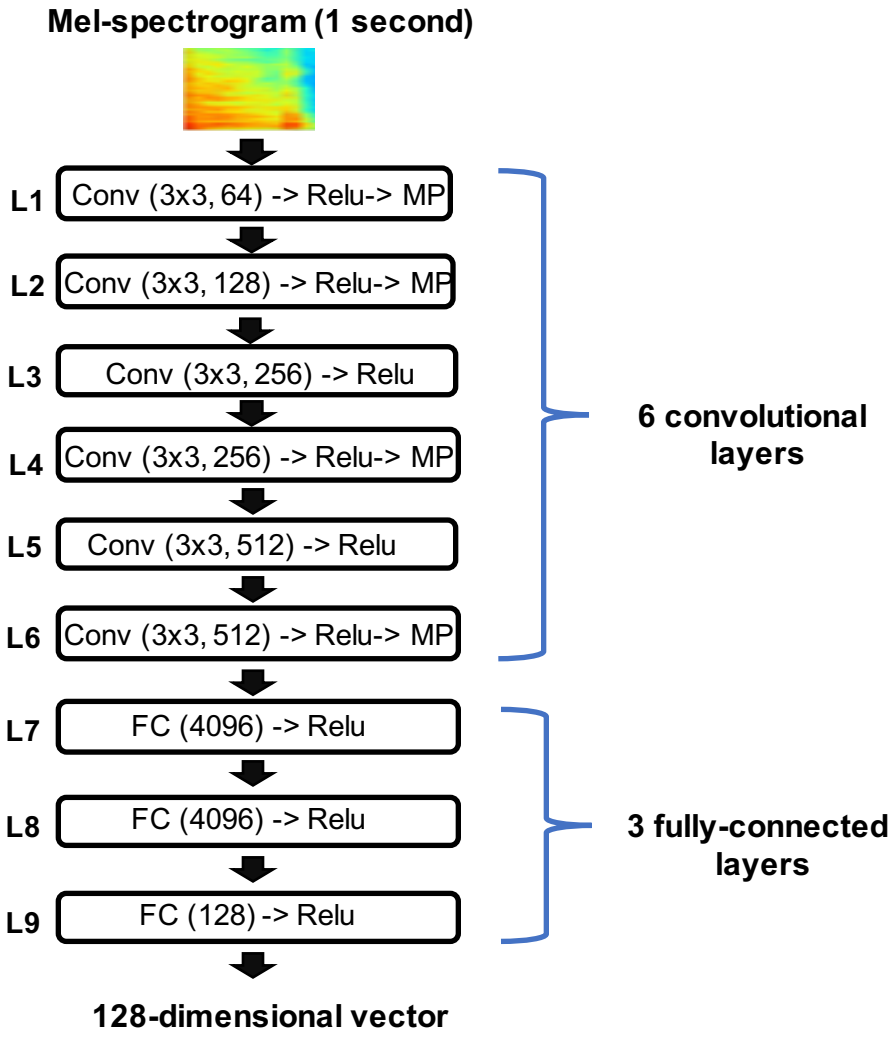


Figure 3.5. The architecture of the VGGish model [**38**]. It consists of 6 convolutional layers followed by 3 fully-connected layers. It takes a 1-second recording and output a 128-dimensional feature vector. The number of filters and their size information of convolutional layers are denoted as (width $\times$ height, channels) in each layer block. All convolution operations are performed with a stride of 1. MP indicates max pooling operation with a kernel size of $2x2$ and a stride of 2. Relu activation functions are used for every convolutional layer.

I modified the VGGish model to build a feature extractor for my retrieval system in the following ways. First, I use only its convolutional layers (L1 to 6 in Figure 3.5). Fully-connected layers of CNNs are considered classification layers for more specific tasks. I believe

using only convolutional layers enables us to extract more general level audio features. Second, the VGGish model takes a fixed length of an audio segment. I changed the input pipeline and added global average-pooling operation so it can take an arbitrary length of a recording while generating a fixed dimensional feature vector. Third, the outputs from each intermediate convolutional layer in the model can be considered different representations of the audio input. I combined outputs from multiple convolutional layers to generate a feature vector. I will call the modified version of VGGish *M-VGGish* throughout the remaining sections of this dissertation.

Another important requirement for designing my retrieval system is that the feature extractor also should capture acoustic characteristics of vocal imitations since my approach to improving QBE search is to use users' vocal imitations. Zhang et al. [**119, 120, 118**] have shown that a Siamese style two-tower network (*TL-IMINET*), given a vocal imitation, successfully finds its original recording (the recording the vocal imitation was an imitation of). The network model takes a vocal imitation as one input and an audio recording from the collection as the other, returning a similarity value for the pair. The limitations of their models are that they were trained on a small number (thousands) of vocal imitations and take fixed-length audio (a 4-second frame) as input. I chose M-VGGish to extract features of vocal imitations through the experiments (The experiment will be presented in Section 3.4.1). The experiment result shows that M-VGGish outperforms TL-IMINET and VGGish in measuring the similarity between a vocal imitation and an actual sound event.

Figure 3.6 shows the architecture of M-VGGish. An audio file is first transformed into a log mel-spectrogram (64 Mel bins, a window size of 25 ms and hop size of 10 ms). This is passed to the M-VGGish model. It takes the whole audio file (the audio may be of variable length) as an input. It uses the outputs of the last two convolutional layers, L5 and L6. I

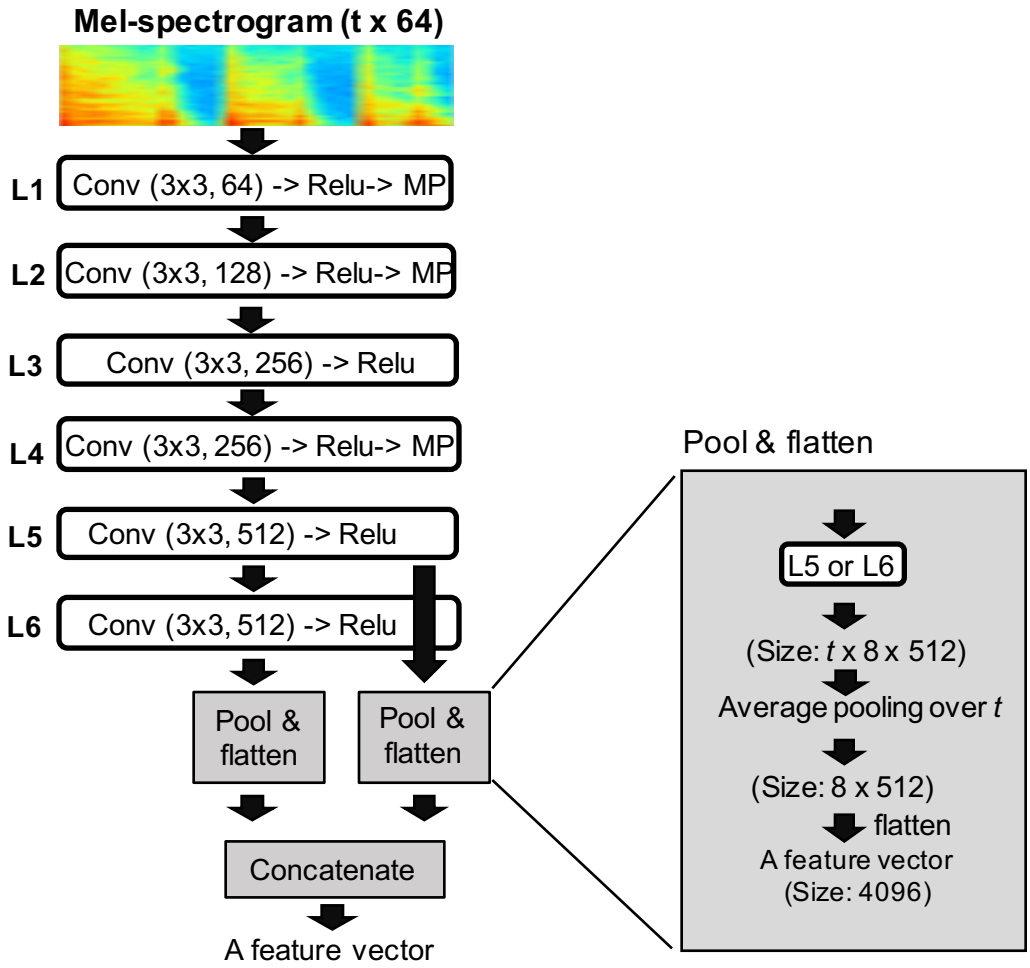


Figure 3.6. The proposed CNN-based feature extractor. The information for each filter is denoted as (width × height, channels) in each layer block. MP indicates max pooling operation. Relu activation functions are used for every convolutaional layer. Outputs from layer 5 and 6 are concatenated to build a single feature vector

explored various combinations of the layers and these showed the best representation powers on a preliminary experiment. I believe that higher layers (L5 and L6) capture relatively more complicated patterns than lower layers (L1 to L4), which leads to generating high-performing features. The size of the output from L5 or L6 is (t, 8, 512) where t depends on the length of input audio. Then, the output is averaged over t (average pooling) and flattened into a

feature vector with the size of 4096. Finally, the outputs from these two layers (i.e., L5 and L6) are concatenated to form a feature vector for an audio example.

The M-VGGish feature extractor is length-agnostic and can take input of arbitrary length. This is very useful features because the lengths of users' vocal imitations often vary across people or multiple trials from a single person. A search system with M-VGGish feature extractor does not need to deal with any segmentation for a vocal imitation. The M-VGGish feature extractor is publicly available.[2]

Once feature embeddings (the output of M-VGGish) are obtained for sound events, the similarity between two sounds is calculated using cosine similarity in the embedding space. I chose cosine similarity over p-norm distance measure because of its normalization factor. Even if two sound events are acoustically similar, their loudness could be very different. The difference might become more obvious when measuring the similarity between a vocal imitation and an actual sound event. The cosine similarity between two sound events, x and y is computed as follows:

$$C(x, y) = \frac{< x, y >}{\|x\|\|y\|}, \tag{3.1}$$

where $< x, y >$ is inner-product between x and y, and $\|x\|$ is euclidean-norm of x.

3.2.2. Updating search results by vocal imitations

In this section, I present methods to use vocal imitations to update the similarity between a query and recordings in a database. Suppose a user has a recording they wish to use as a query, but it contains multiple overlapping sounds. For example, if the query is a birdsong

[2]https://github.com/bongjun/M-VGGish

recorded in a natural setting, the example may also contain dog barking, lawn mowers, etc. Given the noisy query, the search system returns a list of sound events sorted by their similarities to the query and none of the highly-ranked items is the sound event of interest. My approach to this problem is to let a user to imitate the negative (irrelevant) sound event in the query (i.e. negative imitation) or imitate the positive (relevant) sound event in the query (i.e. positive imitation). The user's vocal imitations are considered additional user-provided audio examples of a positive and negative sound event in a query recording.

Positive and negative vocal imitations provide the system with the information about what sound events of interest should and should *not* sound like. It is quite obvious that imitating a positive sound event in a query (positive imitation) can improve the QBE audio search since vocal imitations have been successfully used as a query in exiting audio search systems [**119, 120, 6, 89, 41**]. However, it is also very effective to use negative imitation in improving QBE audio retrieval. Since the poor search result is due to negative sound events overlapped with the positive event in a query, highly-ranked items in the search results could be recordings that sound similar to the negative sound in a query. Negative vocal imitations can be used as information to allow the system to update the initially measured similarity of the query with items in a database in a way that the highly-ranked, but irrelevant recordings are pushed to the lower position in the search result.

My work is the first audio search system that allows a user to provide negative vocal imitations. This is very useful when a positive sound event in a query is hard to vocally imitate (e.g., complex machine sound), but a negative sound event is easy to imitate (e.g., cat meowing). I will report the experimental results in Section 3.4 showing that providing only negative imitations to the search system also improves the retrieval result although the best performance gain can be achieved by providing both positive and negative imitations.

In this section, I present two different methods to update a search result using vocal imitations: *ensemble method* and *query expansion method*. Later in this chapter, I perform experiments using both methods (read Section 3.4.2).

3.2.2.1. Ensemble method. The search system uses each of the two types of queries separately (an actual sound and vocal imitations) and combines the outputs of each retrieval. This technique can be thought of as an ensemble method where slightly different estimations from a model on the same task are combined to generate more robust results. In other words, my search system combines a result from query-by-example search with one from query-by-vocal imitation searches given positive and negative imitation queries.

How the system updates search results, given positive and negative imitations of a query, is the following. First, given a query containing real sound events, the search system measures the similarity between the query and other recordings in a database. When a positive vocal imitation and negative vocal imitation are given, the similarity between the query and a recording in the database is updated by the similarity between each of the vocal imitations and the recording in the database. The updated similarity S between a query q and a recording in a database x is computed as:

$$S(q,x) = C(q,x) + \frac{\alpha}{N_{vp}} \sum_{i=1}^{N_{vp}} C(v_p^i, x) - \frac{\beta}{N_{vn}} \sum_{i=1}^{N_{vn}} C(v_n^i, x) \tag{3.2}$$

where $C(\cdot)$ is cosine similarity, v_n^i is a i^{th} vocal imitation of an irrelevant sound in the query (N_{vn} in total), and v_p^i is a i^{th} vocal imitations of a negative sound in the query (N_{vp} in total). α and β are hyper-parameters controlling the importance factors of positive and negative imitations that can be determined through experiments. (I set $\alpha = 1$ and $\beta = 1$ in the

experiment of Section 3.4). The system can flexibly accommodate one or more negative and positive imitations from a user.

One advantage of processing actual sound query and vocal imitation query separately is that the system can run two different search methods (i.e. feature extractions and matching algorithms) that are designed for each type of query. I used M-VGGish and cosine similarity for both types of quires because the experiments showed that they also outperformed exiting vocal imitation-based search models [**119, 118**] and recently released audio embedding models [**22, 38**] (Read Section 3.4.1 for more details). However, the ensemble method leaves us an opportunity that one can use two different similarity measures for query-by-example search and query-by-vocal imitation search depending on the performances on their own tasks.

While the ensemble method allows to use different similarity measure for vocal imitations, it might be computationally expensive because the system has to process every vocal imitation separately as additional queries. For example, given a query with a positive imitation and a negative imitation, the system performs three different searches in parallel. I will present another method that does not require multiple parallel searches in the next section.

3.2.2.2. Query expansion. To update search results using the ensemble method, each of vocal imitations that are provided to the system is treated as an additional query and multiple searches need to be performed in parallel. In this section, I present another way of using vocal imitations to update the search results where a query itself (a real sound) is expanded by user's vocal imitations before the search process starts so the search can be performed only once with the updated query.

I used the *Ricchio algorithm* [**88**] which is one of the earliest query expansion methods and has been used for interactive document retrieval [**113, 107**]. The idea is to create a new

query vector that is a combination of the original one (a real sound event) and the vocal imitation ones (positive and negative). The new query vector is closer to positive imitation vectors and farther to negative imitation vectors than the original query in the feature space.

Given an old query q_{old} and user's vocal imitations, the new query q_{new} is computed as:

$$q_{new} = q_{old} + \frac{\alpha}{N_{vp}} \sum_{i=1}^{N_{vp}} v_p^i - \frac{\beta}{N_{vn}} \sum_{i=1}^{N_{vn}} v_n^i \tag{3.3}$$

where v_n^i is a i^{th} vocal imitation of an irrelevant sound in the query (N_{vn} in total), and v_p^i is a i^{th} vocal imitations of a positive sound in the query (N_{vp} in total). α and β are the importance factors of positive and negative imitations.

To use the query expansion method with vocal imitations, it is important to generate a feature embedding space where real sound events and vocal imitations can be directly compared. I use my feature extractor presented in Section 3.2.1, M-VGGish to generate feature embedding for both real sound events and vocal imitations, then update the query vector using positive and negative imitation vectors. The updated query vector is compared with items in the database by cosine similarity (Equation 3.1). In the experiment (Section 3.4.1), I will show that M-VGGish features with cosine similarity successfully measure the similarity between real sound and vocal imitations.

3.3. Dataset: Vocal Imitation Set

To evaluate the effectiveness of users' vocal imitations on improving QBE audio search, I need a dataset containing various sound events and human's vocal imitations of them. I created *Vocal Imitation Set*, a new crowd-sourced vocal imitation dataset to use to evaluate the proposed QBE search methods. It is the first dataset of vocal imitations that uses AudioSet ontology [**29**] which is a widely-used ontology of environmental sound. Moreover,

Vocal Imitation Set has more than double the number of imitations available in the largest prior vocal imitation dataset [**17, 69**]. It includes 5,601 vocal imitations of 302 sound classes. In the following sections, I present how the dataset was collected and evaluated.

3.3.1. Data collection

3.3.1.1. Reference audio collection. To collect human vocal imitations, it is an essential step to collect audio recordings to imitate (i.e., reference recordings). Since my goal for this data collection is to create a vocal imitation dataset that can be used to evaluate the effectiveness of vocal imitation on general QBE audio search, the set of sound classes should cover a wide range of sound events. Therefore, I selected sound classes from the AudioSet ontology [**29**]. This ontology contains 632 sound classes that are structured hierarchically with a maximum depth of 6 levels. The top-level categories include *Animal sounds*, *Channel/environment/background sounds*, *Human sounds*, *Music*, *Natural sounds*, *Sounds of things*, and *Source-ambiguous sounds*.

The sound classes in the AudioSet ontology were manually curated to represent a broad set of audio events one might encounter in real-world recordings and each class is assumed to be distinguishable from other classes based on sound alone without any additional information (e.g., visual cue or details of context). For each sound class, AudioSet provides links to YouTube videos that were tagged with the text label for that class. The audio tracks from these videos typically contain multiple, overlapping sounds. Perhaps, for this reason, audio from these YouTube videos has been widely used as a benchmark dataset for sound event detection and scene classification [**57, 54**]. For more details about AudioSet, refer to [**29**].

The AudioSet ontology contains many sound classes that cannot be readily imitated vocally, such as *guitar amplifier* and labels related to music genres. After excluding these

classes, 302 sound classes from the AudioSet ontology remained. AudioSet's actual audio typically contains scenes with multiple sounds, rather than isolated sounds. Since the goal of my data set is to provide clear pairings of vocal imitations to reference sounds, this makes AudioSet's audio sub-optimal. Therefore, I collected sounds from a repository where contributors typically provide isolated, single-sound recordings. For each of the 302 selected sound classes, I collected an average of 10 audio recordings from *Freesound* using the class name as the search key. All files were truncated to a maximum of 20 seconds and encoded in the WAV format with a sample rate of either 44.1 kHz or 48 kHz.

A single high-quality recording was selected from the collected recordings for each class as a reference recording to be imitated by crowd-workers. Each reference audio file was confirmed to contain a clear sound event for the selected sound class and no other sound events. The other recordings that were not used for imitation collection. The other recordings that were not used for imitation collection are also included in the released dataset. In the experiment section of this chapter, these recordings are used as items in a database for QBE audio search where the task is to search for sound events similar to a query.

3.3.1.2. Vocal imitation collection. I collected vocal imitations from crowd-workers through Amazon Mechanical Turk using the VocalSketch interface and protocol presented in [**17**]. Imitators were asked to listen to a reference recording (i.e., one of the 302 collected reference recordings) and imitate the sound. During the recording session, no text-label of the recording was provided to the participants. Once they recorded their imitations, they were required to listen to their imitations to compare them with the reference recording. They were allowed to re-record their vocal imitations unlimited times before submitting the final one. Discarded imitations were saved as *draft recordings* in the released dataset. Finally, each imitator was asked how satisfied they were with their imitations using a 7 level scale.

In each session, imitators were given five reference recordings (one recording from each class) to imitate. Imitators were paid \$0.80 per session. The first imitation of each imitator in a new session was saved as a *training recording*.

I collected a total of $11,242$ recordings from 455 unique people. There were $6,115$ final-submission vocal imitations, $4,444$ draft recordings and 683 training recordings. The $6,115$ final submission vocal limitations resulted in an average of roughly 20 imitations for each of the 302 reference recordings. I focused on this set of final submissions in the quality assessments of the dataset.

3.3.2. Quality assessment

Crowd-sourced data collection suffers from noisy data in many cases. Therefore, I conducted a quality assessment of the $6,115$ final submissions, where experts evaluated the quality of all the final collected imitations. Training and draft vocal recordings were not evaluated. The purposes of the quality assessment are the following: 1) removing non-identifiable vocal imitations from the data set, and 2) measuring perceptual similarity between a reference recording and its imitations. The people who performed quality assessment were experts in audio processing: students and researchers from the Interactive Audio Lab [3] at Northwestern University and the Audio Information Research Lab [4] at the University of Rochester. There were, in total 15 evaluators, who listened to $6,115$ vocal imitations on a web interface designed for this particular listening task.

Figure 3.7 shows the web interface for the quality assessment. A single session consists of listening to a pair of recordings: one reference and one vocal imitation (Sound A and

[3] `http://music.cs.northwestern.edu/`
[4] `http://www.ece.rochester.edu/projects/air/`

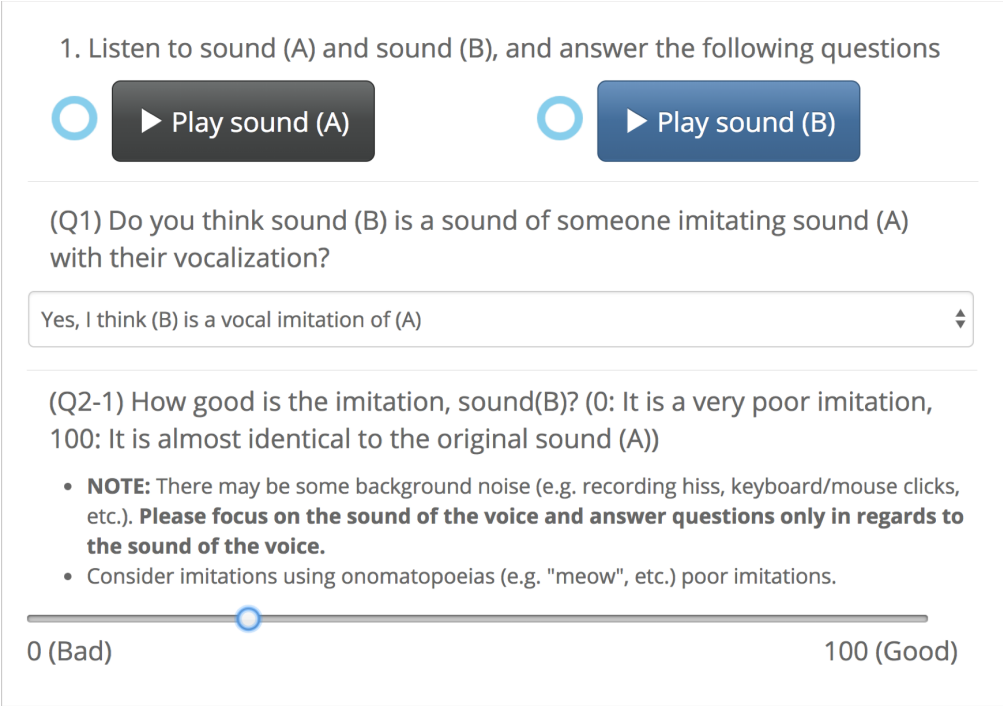


Figure 3.7. A screenshot of the interface for the internal quality assessment

Sound B in Figure 3.7). An evaluator was first asked if the imitation was a vocal imitation of the reference recording. If the answer was "YES", then the evaluator was asked to assess the quality of the imitation on a scale from 0 to 100 (0: a very poor imitation; 100: almost identical to the reference sound). If the answer was "NO", then the recording was not evaluated for quality and it was placed in the *excluded* directory of the released dataset. The evaluator was then asked if the recording was a vocal imitation at all and this answer was saved.

Due to the size of the dataset, each imitation was evaluated by a single person. To measure the consistency and reliability of each evaluator, I designed the task in the following ways. First, an average of 2 out of every 30 pairs evaluated by an individual were incorrect pairs, where I paired an imitation with a reference recording that it was not an imitation of.

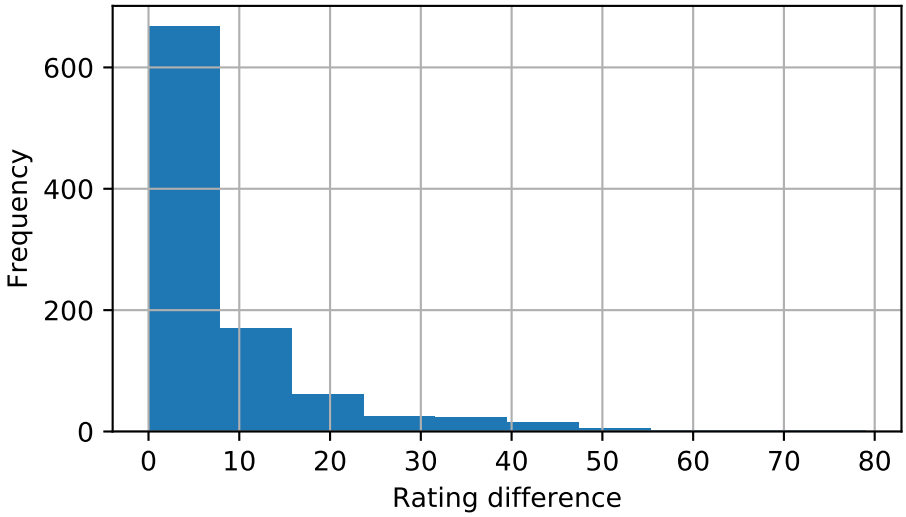


Figure 3.8. Histogram of maximum differences of quality ratings on a 100 point scale between two presentations of the same pairing of reference and imitation recording (Mean: 7.63, SD: 10.96)

This lets us measure how reliably evaluators were able to detect incorrect pairs. Second, an average of 4 out of every 30 pairs presented to an evaluator were repeated pairs, previously presented within the current batch (30 pairs). This let us measure the evaluation consistency for each evaluator.

In total, 452 incorrect pairs were presented to evaluators and 80% of them (363 pairs) were successfully identified as incorrect pairs. The remaining 20% (89 pairs) were incorrectly called correct pairs and they were given an average quality rating of 31.4 out of 100. The mean quality rating across all imitations is 60.3. This indicates that most evaluators correctly identified wrong pairs or gave them low scores if they called them a correct pair. Figure 3.8 shows how consistently evaluators rated repeated pairs. In total, 978 unique pairs of reference and imitation recordings were repeated. We computed the maximum difference of the multiple ratings to each of the 978 repeated pairs. For example, if a pair of reference

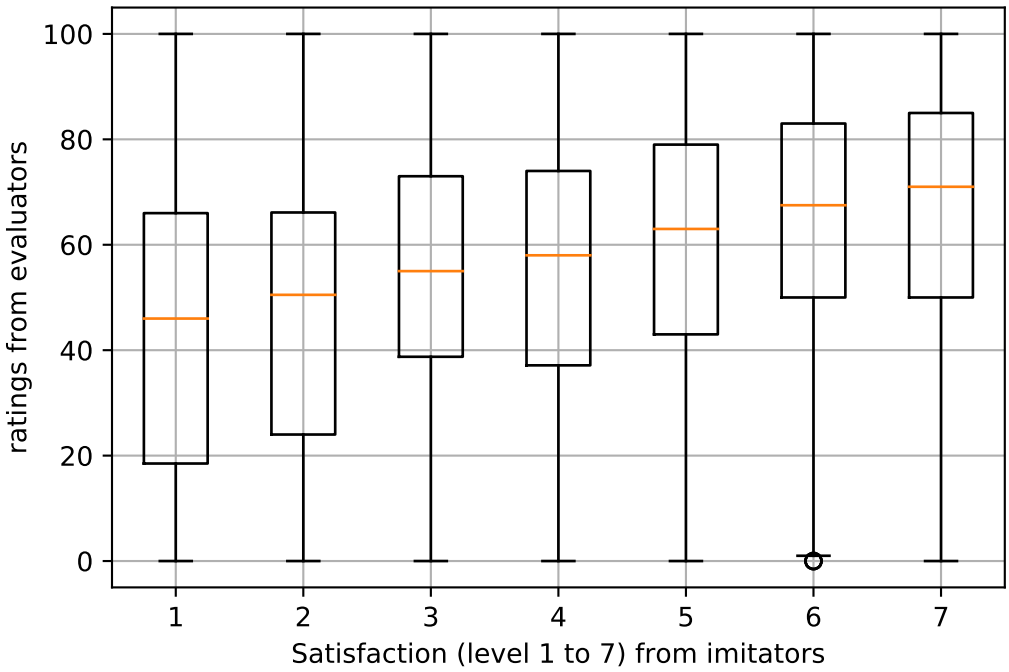


Figure 3.9. Relationship between self-satisfaction scores by imitators and quality assessment by evaluators.

and imitation recording was repeatedly rated three times by an evaluator and the ratings were 50, 60 and 70, then the maximum difference is 20 (70-50). As shown in Figure 3.8, the maximum differences of a majority of repeated pairs is very low (Mean: 7.63, SD: 10.96), which indicates that our evaluators rated vocal imitations with high consistency.

When collecting imitations, imitators were asked how satisfied they were with their own imitation using a 7 level scale. (1 - *completely dissatisfied,* 2 - *mostly dissatisfied,* 3 - *somewhat dissatisfied,* 4 - *neither satisfied or dissatisfied,* 5 - *somewhat satisfied,* 6 - *mostly satisfied,* 7 - *completely satisfied*). Figure 3.9 shows how the evaluator's ratings change with different self-satisfaction levels from imitators. As shown in the figure, there is a positive correlation between the imitators' self-satisfaction levels and evaluators' quality assessment

scores with Pearson correlation coefficient of 0.98 between mean ratings and satisfaction scores. Yet, there are some limitations where the imitator's self-satisfaction disagrees with the quality reported by an evaluator. It would be interesting future work to learn the reason for the dichotomy.

Evaluators reported that 514 vocal imitations were not vocal imitations of the reference sound played to the imitator who made the imitation. These recordings were placed in the *excluded* directory of the released dataset. This left 5,601 recordings that have quality ratings, which are saved in the *included* directory of the dataset. I did not perform any additional filtering by rating values we collected because it is not easy to set thresholds for that. I included all the quality ratings on these 5,601 recordings in the released dataset so that other researchers can refer to them for their own purposes.

3.3.3. Summary of dataset

Vocal Imitation Set is now publicly available[5]. It includes 2,985 original recordings of 302 classes (an average of 9.89 per class) and 11,242 vocal imitations of 302 reference recordings selected from the set of original recordings (1 reference recording per class). The set of vocal imitations consists of 5,601 imitations that passed the quality assessment as well as 5,642 recordings of draft, training recordings, and imitations excluded during the quality assessment. Table 3.1 shows the number of classes, listener-vetted imitations (i.e., imitations that have quality ratings), and original recordings for each top-level classes of the AudioSet ontology. Figure 3.10 shows a histogram of quality assessment ratings of the 5,601 listener-vetted imitations. The collected ratings give researchers another opportunity to build more robust vocal imitation-based interaction systems by using human quality assessments as a

[5] `http://doi.org/10.5281/zenodo.1340763`

Table 3.1. The number of classes, listener-vetted imitations, and original recordings (including reference recordings) for each of the first-level categories in Vocal Imitation Set

Categories	Classes	Imitations	Original Rec.
Animal	31	587	308
Channel, environment and background	4	74	40
Huma sounds	38	714	375
Music	65	1247	646
Natural sounds	10	177	100
Sounds of things	134	2448	1316
Source-ambiguous sounds	20	354	200
Total	302	5601	2985

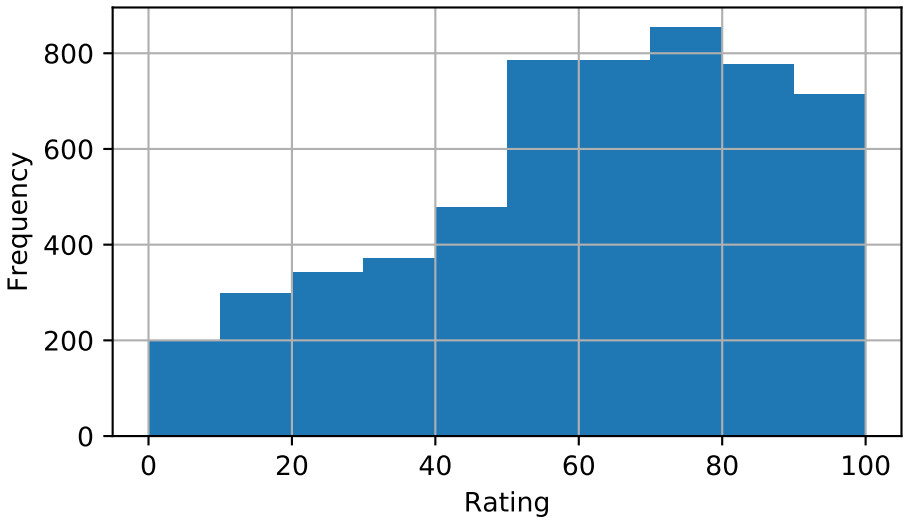


Figure 3.10. Histogram of quality assessment ratings to 5,601 vocal imitations that were vetted by evaluators (Mean: 60.3, SD: 25.3)

training signal. In the next section, I use Vocal Imitation Set to evaluate my approach to improving QBE audio search using users' vocal imitations.

3.4. Evaluation

I perform a set of experiments to evaluate the proposed methods. First, I evaluate the new feature extractor M-VGGish by comparing it with other existing solutions. Then, I perform experiments to evaluate how much vocally imitating sound events in a query improves QBE audio retrieval. The goal is to answer the following questions:

- Q1) Is M-VGGish effective in measuring the similarity between a vocal imitation and a real sound event?
- Q2) how does a query containing positive and negative sounds overlapping each other affect QBE search?
- Q3) Does a vocal imitation of the positive sound in a query (positive vocal imitation) improve the search performance?
- Q4) Does a vocal imitation of the negative sound in a query (negative vocal imitation) improve the search performance?
- Q5) Is using vocal imitation of both the positive sound and negative sound in a query more effective in improving the search performance than using only either of them?

3.4.1. Experiment-1: Evaluation of M-VGGish model

Since my proposed approach to improving QBE audio search is to use user's vocal imitations, it is important to extract high-performing audio embedding to measure the similarity between a vocal imitation and a real sound event. Therefore I presented a new audio embedding model M-VGGish in Section 3.2.1. In this section, I performed the experiments to choose a feature

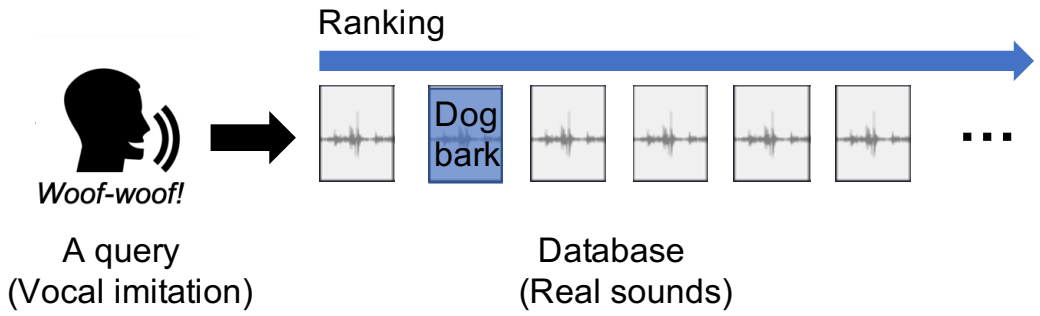


Figure 3.11. Query-by-Vocal imitation audio retrieval. A vocal imitation recording of a category (e.g., Animal) is a query (e.g., a vocal imitation of a dog bark). Then, reference recordings (real sounds) of the category become the database to search through. The task is to find a real recording that is the most similar to a vocal imitation query.

extractor to be used for my proposed method to update audio search, answering the research question Q1. I compare the M-VGGish model with other existing methods.

I evaluate methods to measure the similarity between a vocal imitation and a real sound event by performing vocal imitation-based audio search. The task is to search for a recording containing a sound event in a database given a vocal imitation of the sound event as a query. The QBV retrieval results would tell us how accurately the system measures the similarity between a sound event and its vocal imitation (i.e. query).

3.4.1.1. Dataset. To perform the QBV retrieval task, I selected a subset of Vocal Imitation Set presented in Section 3.3 as a testing set: a set of vocal imitations and their reference recordings. Vocal Imitation Set contains 7 categories of sound events and each category has a different number of sound classes as shown in Table 3.1. I selected 5 categories that contain more than 20 sound classes: *Animal (31), Human sounds(38), Music (65), Sound of things (134), and Source-ambiguous sounds (20).* Each class contains a single reference recording and about 20 vocal imitations of the reference recording.

3.4.1.2. Setting. Figure 3.11 illustrates Query-by-vocal imitation audio retrieval. I perform within-category retrieval with the dataset. Each vocal imitation in a category (e.g., Animal category) is used as a query and the set of reference recordings of the category (e.g., 31 reference recordings of Animal category) become items in a database that the search system needs to search through to find the target recording. For example, given a vocal imitation of a dog bark, the system searches through 31 reference recordings of Animal category to find a real dog bark sound. Each query (vocal imitation) is compared to each reference recording (an actual sound event) in the database and all recordings in the database are then ranked by their similarity to the vocal imitation.

To measure the success of the retrieval, I compute Mean Reciprocal Rank (MRR) of the target reference recording as following

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i}, \tag{3.4}$$

where N denotes the number of queries and $rank_i$ indicates the rank position of the reference recording correspond to the ith vocal imitation query. For example, MRR of 0.5 means that target reference recordings were retrieved at rank 2 on average.

Based on the performance measure, the proposed feature extractor, M-VGGish is compared to three existing deep neural network-based methods:

- TL-IMINET [**119**]: A Siamese style two-tower deep network designed specifically for QBV audio search. It is composed of two convolutional neural network (CNN) towers that feed into several fully connected layers to combine outputs from the two CNN towers. Each tower takes a real recording and a vocal imitation respectively

and outputs the similarity between them. I trained the model following their training procedure as described in [**119**].

- VGGish [**38**]: The original version of M-VGGish. It takes 1 second of audio and output 128-dimensional embedding. The pre-trained model is publicly available [6]. I used the model to extract features from recordings. When extracting features from a recording longer than 1 second, features of a non-overlapping 1-second window of the recording are averaged. To measure the similarity between recordings, I compute Cosine similarity in the feature space.
- OpenL3 [**22**]: An open-source Python library for deep audio embedding. It is based on the approach known as Look, Listen and Learn (L^3-Net) [**3**]. The model is trained in a self-supervised manner to detect matched video and audio pairs. The models provided in the library have been trained on 296K and 195K videos including music and environmental sounds respectively. These videos were selected from AudioSet[7].For more details about the L3-Net model, read [**3**]. The library provides variants of models with different input representations and training data domains. I choose the setting that has shown the best performance on sound classification in [**22**]: Mel-spectrogram with 256 Mel bins for an input representation and music for training data domain. The trained model takes 1-second of audio and outputs 6144-dimensional embedding. When extracting features from a recording longer than 1 second, features of 1-second windows of the recording are averaged. To measure the similarity between recordings, I compute Cosine similarity in the feature space.

[6]https://github.com/tensorflow/models/tree/master/research/audioset/vggish
[7]https://research.google.com/audioset/

Table 3.2. MRRs of the three retrieval systems on 6 sub-categories of Vocal Imitation Set. The number of queries (vocal imitations) and items (real sounds) to search through for each category is following: Animal (587 queries, 31 items), Human sound (714 queries, 38 items), Music (1247 queries, 65 items), Sound of things (2448 queries, 134 items), and Source-ambiguous (354 queries, 20 items).

Category	M-VGGish	TL-IMINET [**119**]	VGGish [**38**]	OpenL3 [**22**]
Animal	**0.351**	0.189	0.225	0.278
Human sound	**0.372**	0.194	0.249	0.263
Music	**0.181**	0.119	0.117	0.141
Sound of things	**0.113**	0.071	0.066	0.087
Source-ambiguous	**0.302**	0.269	0.217	0.233

3.4.1.3. Results. Figure 3.2 compares within-category retrieval performances of M-VGGish, TL-IMINET, VGGish, and OpenL3. It shows that the proposed similarity measure, M-VGGish outperformed the three models for all 6 categories of sound. The performance difference between M-VGGish and other models is statistically significant for all the categories (a Wilcoxon signed-rank test, $p < 0.05$). Moreover, M-VGGish has a fewer number of parameters (4.5M) than the second and third best models, OpenL3 (4.5M) and VGGish (62M), which makes M-VGGish even more preferable.

In the remaining experiments, I use M-VGGish model as a feature extractor for both real sound event recordings and vocal imitation recordings when measuring the similarity between them. I also use the same method to compare two real sound events, which is not possible with TL-IMINET.

3.4.2. Experiment-2: Evaluating of query-by-example search with vocal imitations

In this section, I evaluate the effectiveness of vocally imitating sound events in a query in query-by-example audio retrieval to answer the research questions Q2 to Q5 listed at the beginning of Section 3.4. The experiment result will provide evidence that user's vocal imitations can improve the initial QBE search of the interactive annotation process presented in 2.

3.4.2.1. Dataset. I use Vocal Imitation Set presented in Section 3.3 as a testing set. It contains about 10 original recordings (real sounds) per class. One of the original recordings that has its vocal imitations is called a reference recording and it is used as a query. This experiment also used the same 5 categories of Vocal Imitation Set as the previous experiments did in Section 3.4.1. The number of queries for each category is the following: *Animal (31), Human sounds(38), Music (65), Sound of things (134), and Source-ambiguous sounds (20).* All the other original recordings that are not a reference recording become a set of items in a database that the system needs to search through to find relevant items (recordings) given a query. The number of items to search through for each category is the following: *Animal (277), Human sounds(337), Music (581), Sound of things (1,182), and Source-ambiguous sounds (180)*

To simulate the situation where QBE search might fail because of a query containing overlapping sound events, I created a set of difficult queries by mixing two recordings selected from a set of reference recordings for each category. Each reference recording is mixed with another randomly selected reference recording of a category. I call the set of the queries *Mixed-queries.* I also call the original reference recordings (not mixed), *Clean-queries.* The

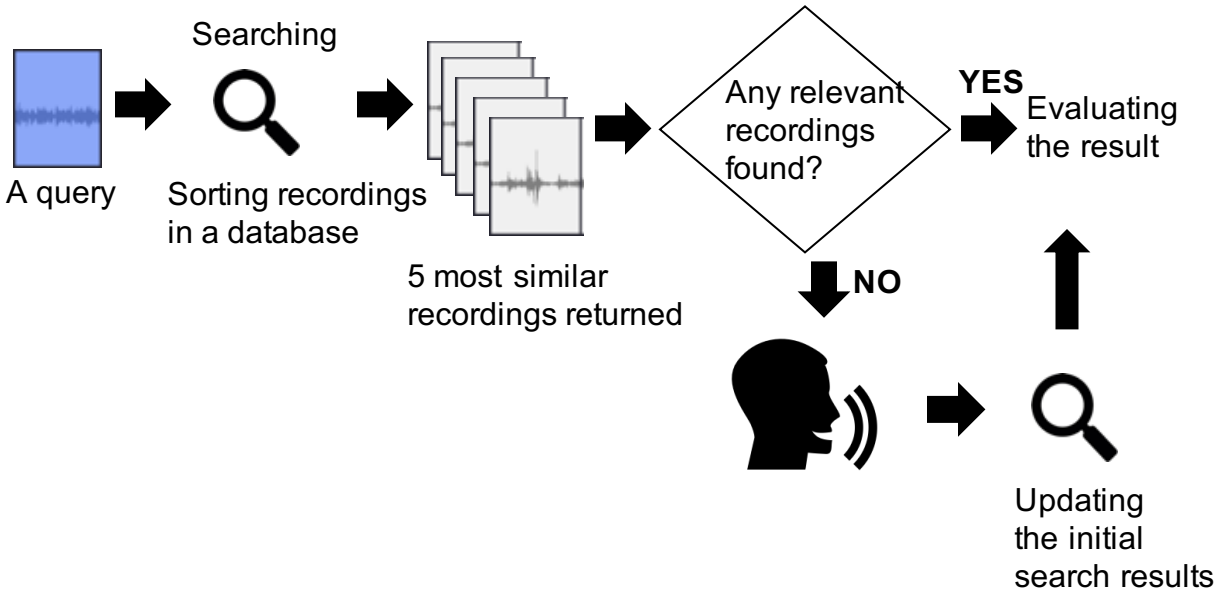


Figure 3.12. The experiment scenario to evaluate the effectiveness of user's vocal imitations in query-by-example audio retrieval. The database to search through includes sound events that are *similar* (belonging to the same class) to sound events in a query. The task is to find several sound events that are similar to the positive sound in the query. If the 5 top-ranked items do not include sound events of interest, the search results get updated by vocal imitations.

Clean-queries is used to obtain upper-bound retrieval performance of my search system as a reference in this experiment.

3.4.2.2. Setting. The task in this experiment is to find multiple relevant recordings (an average of 9 per query) that sound similar to a sound event of interest in a mixed query. Any of the relevant recordings in a database is not identical to any sound events in a query. They just belong to the same class. For example, if a query contains a dog bark and car engine sounds and the event of interest is a dog bark, then the goal is to find roughly 9 different dog barks in a database.

To measure the performance gain by the user's vocal imitations, I simulated a user's interaction with the search system as shown in Figure 3.12. Given a query recording, the

Table 3.3. Mean Recall@10 of within-category QBE retrievals for the five different search scenarios. The Ensemble method was used to update the search results with vocal imitations.

Category	Clean	Mixed	Mixed+Pos	Mixed+Neg	Mixed+Pos+Neg
Animal	0.434	0.262	0.315	0.272	**0.369**
Human sound	0.376	0.192	0.223	0.238	**0.282**
Music	0.38	0.21	0.254	0.251	**0.286**
Sound of things	0.298	0.157	0.178	0.177	**0.189**
Source-ambiguous	0.389	0.106	0.144	0.133	**0.167**

system returns a list of recordings in the testing set ordered by similarity with the query. If the n top-ranked recordings do not include any of the relevant recordings, vocal imitations are provided to improve the search results. I set $n = 5$ for this experiment to simulate the first round of the interactive searching presented in Chapter 2.

To measure the performance of QBE retrieval, I compute two performance metrics: *Recall@10* and *Max-MRR*. Recall@10 measures recall within the top 10 items in search results. For example, Recall@10 of 0.7 means that 70% of target recordings are retrieved within the top 10 items of the search results. Recall@10 is a more appropriate measure than precision@10 since many classes in the testing set contain less than 10 target sound events so I am measuring the proportion of retrieved relevant items to the total number of target events for each class. I report the Mean Recall@10 across all the queries of a category. Max-MRR measures MRR of the highest-ranked target item in a search result. It evaluates how quickly a user can find the first sound event of interest during rounds of the interactive annotation process.

3.4.2.3. Results. Table 3.3 and Table 3.4 shows MeanRecall@10 and Max-MRR for the following 5 different search scenarios:

Table 3.4. Max-MRR of within-category QBE retrievals for the five different search scenarios. The Ensemble method was used to update the search results with vocal imitations.

Category	Clean	Mixed	Mixed+Pos	Mixed+Neg	Mixed+Pos+Neg
Animal	0.77	0.529	0.653	0.563	**0.722**
Human sound	0.627	0.358	0.43	0.467	**0.543**
Music	0.74	0.431	0.501	0.508	**0.54**
Sound of things	0.589	0.357	0.425	0.423	**0.458**
Source-ambiguous	0.836	0.264	0.317	0.316	**0.452**

- Clean: Clean-queries are tested to see the best possible retrieval performance given the testing dataset.
- Mixed: Mixed-queries are tested to see how much a query with overlapping sound events leads to poor retrieval results given the testing dataset.
- Mixed+Pos: The retrieval results by Mixed-queries are updated using a positive vocal imitation only.
- Mixed+Neg: The retrieval results by Mixed-queries are updated using a negative vocal imitation only.
- Mixed+Pos+Neg: The retrieval results by Mixed-queries are updated using both a positive and a negative vocal imitation.

The performances on Clean-queries and Mixed-queries can be thought of as upper-bound and lower-bound performances of my search system given the testing set. The results show that adding a positive and a negative vocal imitation to a mixed query improved both metrics (Recall@10 and Max-MRR) for all the categories of the testing set. Interestingly, only using negative vocal imitation also helps to improve the search results. It confirms that a user can

Table 3.5. Recall10 comparison between the two methods to update the initial search result using vocal imitations: Ensemble and Query expansion method presented in Section 3.2.2.

	Recall@10		Max-MRR	
Category	Ensemble	Query expansion	Ensemble	Query expansion
Animal	**0.369**	**0.369**	**0.722**	0.701
Human sound	0.282	**0.290**	**0.543**	0.532
Music	**0.286**	0.288	0.540	**0.551**
Sound of things	**0.189**	0.185	**0.457**	0.449
Source-ambiguous	**0.167**	0.156	**0.452**	0.450

improve the retrieval by providing negative imitations when a positive event in a query is hard to imitate.

I performed a Wilcoxon signed-rank test on the results. The statistical test showed that the search results updated by a positive and negative vocal imitation are significantly greater than the initial search results ($p < 0.05$) for all the categories except for the Source-ambiguous category with the p value of 0.1. I guess the higher p-value of the Source-ambiguous category is caused because human's vocal imitations of source-ambiguous sound events might not be enough to help the search system distinguish source-ambiguous sounds from another.

I also compared the two update methods, *Ensemble* and *Query expansion* presented in Section 3.2.2 to see which method is more effective in improving QBE search results. Table 3.5 shows within-category Recall@10 and Max-MRR after an initial search result gets updated by each method with positive and negative vocal imitations. Even if the Ensemble method shows better results for most of the categories, the difference is very small. I ran a Wilcoxon signed-rank test and they did not show a statistically significant difference. I can conclude that both methods can successfully improve QBE retrieval using vocal imitations.

One might prefer Query-expansion because the Ensemble method requires multiple searches in parallel (each for a single vocal imitation) while the Query-expansion method only requires a single search regardless of the number of given vocal imitations. However, the Ensemble method can also be preferred for someone who has a better way of measuring the similarity between a vocal imitation and a real sound (i.e. better query-by-vocal imitation systems) and want to use another method separately only for measuring the similarity between real sounds.

3.5. Conclusion

I presented a new CNN-based feature extractor M-VGGish that takes a variable length of audio and generate audio embedding for both real sounds and vocal imitations. I performed vocal imitation-based audio search to evaluate the M-VGGish model. The experiment showed that M-VGGish outperformed other existing methods in measuring the similarity between vocal imitations and real sounds.

I also presented two ways of using vocal imitations of a positive and a negative sound event in a query to update QBE audio search results. Positive and negative vocal imitations provide the system with the information about what sound events of interest should and should *not* sound like. Negative vocal imitations are helpful especially when a poor search result is due to negative sound events overlapped with the positive event in a query. Highly-ranked items in the search results could be recordings that sound similar to the negative sound in a query. In this case, negative vocal imitations can help the system to update search results in a way that the highly-ranked, but irrelevant recordings are pushed to the lower position in the search result. Therefore, the proposed methods allow users to improve

the search result simply by providing the search system with their vocal imitations of the positive and negative sound events in a query recording.

I performed an experiment of Query-by-example audio retrieval to evaluate if vocal imitations of the positive and negative sound events in a query improve the search performance. The experiment results showed that the performance gain could be achieved only by either positive or negative imitation, but using both types of vocal imitations improve the search performance even more. Therefore, users can update poor search results caused by a query containing overlapping sound events by imitating what they do or do *not* want in the query. This work would solve the cold-start problem that the interactive sound event annotation presented in Chapter 2 might face when the initially selected region contains multiple sound events so the system fails to return any relevant sound event during early rounds of the interactive search.

To perform all the experiments in this chapter, I created a new crowd-sourced vocal imitation dataset, *Vocal Imitation Set*. It has more than double the number of imitations available in prior vocal imitation datasets. While I used vocal imitations as an additional input modality for a user to improve QBE search results, I also expect that this dataset will help the research community obtain a better understanding of human vocal imitations and build systems that can understand imitations as humans do.

My work in this chapter provides strong evidence that a user's vocal imitation can help a user to find sound events of interest quickly. One implication of this work is that developers can build systems that apply this new user interaction to help users find desired sound events from a large dataset with less effort. However, as presented in Chapter 2, a new interaction also causes additional interaction overhead. Therefore, as future work, it will be important

to design the search interfaces in a way that a user can seamlessly provide the system with vocal imitations for searches.

I have shown that positive and negative vocal imitations of a query improve the performance of audio search when the query does not have an isolated positive sound event. This is the first work that studied the effectiveness of human vocal imitations in improving QBE search. Especially, none of the prior works has focused on negative vocal imitations for audio search. As future works, this new approach also can be applied to an audio classification task. When no isolated sound event of a certain class is available as training examples, it might be difficult for a machine learning model to learn proper mapping functions between the audio signal and its label. Since positive and negative vocal imitations can be thought of as additional labeled examples, it would be an interesting future work to see how the user-generated examples can be used a part of training data for audio classification tasks.

CHAPTER 4

Sound event detection using point-labeled data

4.1. Introduction

In previous chapters, I have focused on how to quickly collect a set of sound events of interest from a large amount of unlabeled audio data. The presented solutions can be used in a situation where the goal of annotation is to directly quantify sound events of interest in a recording and use the information for direct analysis. In this chapter, I focus on another situation where sound event annotation needs to be done as a precursor to building a machine learning model that performs Sound Event Detection (SED).

SED is a task of identifying a class of sound events and estimating the time position (i.e. start and end) of each occurrence of that class in an audio recording (see Figure 4.1). Automatic SED is an essential task in many areas that require an audio-based understanding of our environment. Sound is very useful information to understand the environment especially in a situation where visual information (e.g., video or images) is not available. A microphone can capture sound events coming from a dark environment or in blind spots where visual objects are occluded from a camera's field of view. Applications of SED include detecting source of noise in urban areas [**5**], identifying bird species singing in nature recordings [**100**], sound-based home monitoring [**103, 95**], office monitoring [**34**], gunshot detection in city recordings [**104**], and detecting anomalous machine sound from a manufacturing system [**35, 52**].

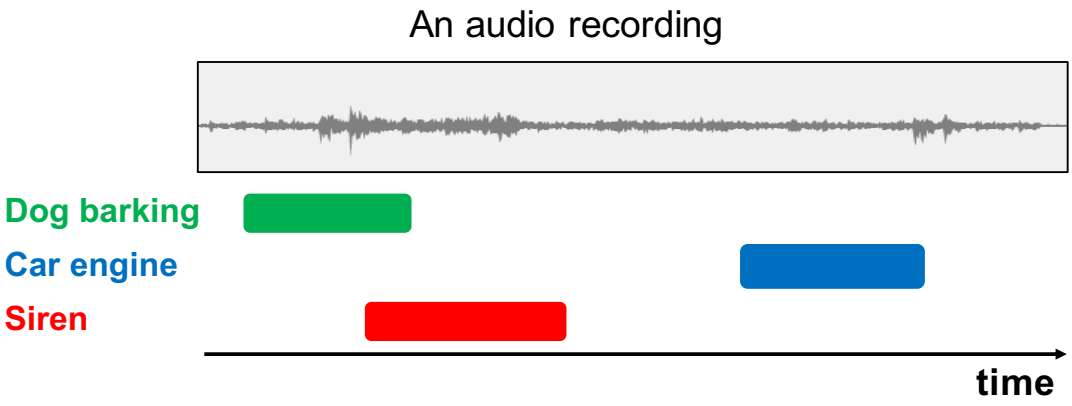


Figure 4.1. Examples of Sound Event Detection (SED). Given an audio recording and a fixed set of sound classes, a SED system automatically identifies the classes of sound events and estimate temporal locations (i.e. start and end) of the events within the recording. There could be multiple sound classes overlapping each other (i.e., polyphonic environment)

The typical approach to building automatic sound recognition systems is to train a machine learning model with labeled data. Examples include neural networks [**81, 38**], Gaussian Mixture Models (GMM) [**109**], decision trees [**60**], Hidden Markov Model (HMM) [**36**], Non-negative Matrix Factorization (NMF) [**23**] and Support Vector Machines (SVM) [**84**]. Recent SED systems often use deep neural networks [**68, 13, 53**] and they are the current state-of-the-art. A machine learning model learns a mapping function between the audio signal and its label (e.g., a recording of a dog bark and its text label 'dog bark'). Once a model is trained on a labeled training dataset, it can estimate a label of a new audio signal.

For a SED system to be maximally effective at detecting sound events and indicating their onset/offset times, it needs to be trained on audio data with time-coded labels that indicate the start and stop times of sound events. This is illustrated in Figure 4.2. Labeling that provides exact onset and offset times of a sound event is called *Strong labeling*. Collecting strongly-labeled data by manually annotating each sound's onset and offset within a

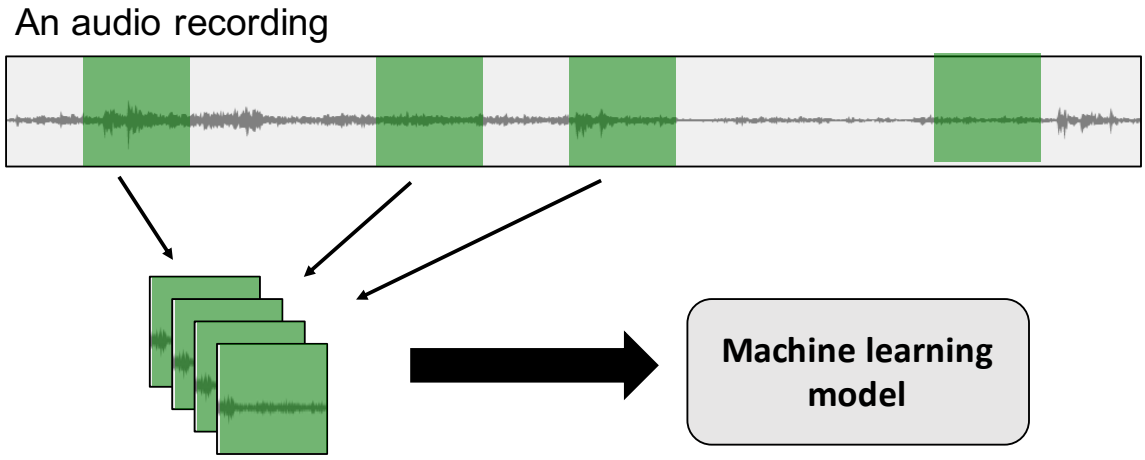


Figure 4.2. For a machine learning model to learn a proper mapping function between an audio signal and its label, each of training examples needs to be well-segmented with correct onset and offset times of an audio event. Therefore, it requires *strong-labels* of sound events in a recording.

recording is a very time-consuming task. A human annotator might need to listen to a sound event multiple times to set correct time boundaries of event labels on a visual interface [**49**]. The annotation would become more challenging in a polyphonic environment where multiple sound events can overlap each other.

To overcome the high cost of collecting strong labels, many researchers have presented models that do not require strongly-labeled audio data for model training. As an alternative to strong labels, *weak* labels have obtained much attention for sound event classification and detection [**54, 68, 58, 57, 46**]. Weakly labeled data names the sounds within an audio recording without specifying anything about onset or offset times (e.g. "there is a dog bark somewhere within this 30-second recording of a park scene"). Collecting weak labels is easier and faster than collecting strong labels, since the human annotator does not need to indicate the exact time boundaries of events, which takes a lot of time. To collect weak labels, one might just need to listen to a sound clip once and record what events are anywhere in the

clip [**15**]. Models trained on weak labels, however, typically do not achieve the performance of models trained on strongly labeled data. This is because there are intrinsic limitations in terms of the information the models are trained on.

In this chapter, I present a new type of audio labeling, called *point labeling* which contains more information than weak labels, but still takes less human time to produce than strong labels. I also present a SED model that can be trained on *point-labeled* training data and show that its performance is similar to the performance of the strong model.

4.1.1. Contributions

My contributions in this chapter are the following:

(1) I present a new type of sound event labeling, point labeling, which (to the best of our knowledge) has not been addressed in prior works on audio labeling

(2) I present a new method to train a machine learning model (in this case, a Fully Convolutional Network) on the point labeled audio data.

(3) I present a strategy to automatically expand point labels so they can cover a greater portion of a sound event, which should lead to performance improvement on SED tasks.

(4) I report the experimental results showing that a model trained on point-labeled audio data significantly outperforms one trained on weak labels and achieves comparable results to a model trained on strongly labeled data.

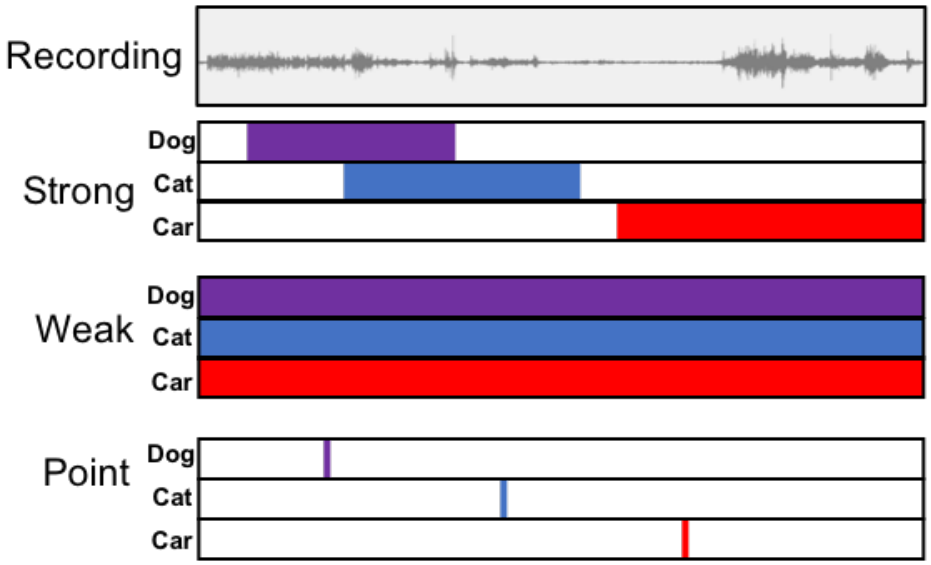


Figure 4.3. Examples of different types of audio annotation. Strong labels contain names of sound events and their time information. Weak labels only have clip-level presence or absence of events without their timing information. Point labels contain names of sound events at a single time point per sound event instance within a recording. The position of each point can vary within each instance.

4.2. Point Labeling

In this section, I present my new method of labeling sound events (i.e. point labeling) and compare it with existing labeling methods (i.e., strong and weak labeling). Figure 4.3 shows examples of strong, weak, and point labels.

Strong labels contain names of sound events and their temporal boundaries. To maximize the SED performance of a model, training audio examples need to be strongly-labeled. However, collecting strong labels is very time-consuming. Specifically, finding the correct onset and offset of each event in a polyphonic environment requires a lot of human effort [**18**].

Weak labels only indicate the presence or absence of sound events within an audio clip, without their temporal location within a recording. Weak labels are relatively easy to collect. A human annotator just needs to judge whether or not a target sound event is at all present in the recording. When modern SED systems are built on a weakly labeled dataset, a multiple-instance learning (MIL) formulation is typically applied, assuming each class of sound events is present during the entire clip if the event is present anywhere within the clip (see *Weak* in Figure 4.3). This information might be enough for *audio tagging* tasks where a trained model is expected to estimate labels for an entire clip, not its temporal information within the clip. However, the lack of time information on a training set would degrade the model performance on SED tasks where the model is expected to estimate time information of events within an audio clip.

My idea to reduce the gap between a strong label and a weak label is *point* labels. As shown in Figure 4.3, point labels contain names of sound events at a single time point per sound event instance. A human annotator can indicate a sound occurred (e.g., by clicking a mouse button or hitting a key in an annotation interface) anywhere within the area of the sound event. I believe that this user interaction takes much less time and effort than that required to find and mark the exact start and stop times for each labeled sound in a recording. In fact, in the right scenario, point labeling could take as little effort as weak labeling, since the time/effort difference between clicking at the time one hears a sound (providing a point label) and waiting until the end of a recording before indicating the presence of a sound (weak labeling) may be very small, if the labeling task is structured appropriately. Therefore, point labeling has the potential to be faster than strong labeling and training a model on point labels should be better than training one on weak labels.

In the remaining sections of this chapter, I will present how to build a SED system on point-labeled audio data and compare the system with other models built on strongly-labeled and weakly-labeled data.

4.3. Training a SED model on point-labeled data

In this section, I present a method to build a machine learning model on point-labeled audio data. First, I present a new loss function *point-label loss* to train neural networks on point labeled audio data. Then, I present a method to expand point labels so they can over a greater portion of a sound event within an audio clip.

4.3.1. point-label loss

To build a SED system in this work, I use deep neural networks, as several recent prior works [**68, 13, 53**] have done. Neural networks are trained to minimize the loss between model predictions and ground-truth labels. During training, a model needs to compute losses between encoded ground-truth labels and the model outputs. In this section, I present how to compute losses for model training when point labels are available. I call the loss *point-label loss*.

A SED model takes an audio clip and outputs class probabilities for each time segment. The output contains the likelihood of each class being present at each time step. During training, the training loss is computed based on the difference between the output matrix and ground-truth labels of the audio clip. Therefore, to compute the point-label loss, L_{point}, point labels of each recording need to be encoded in the form of the output of a typical SED model so they can be directly compared.

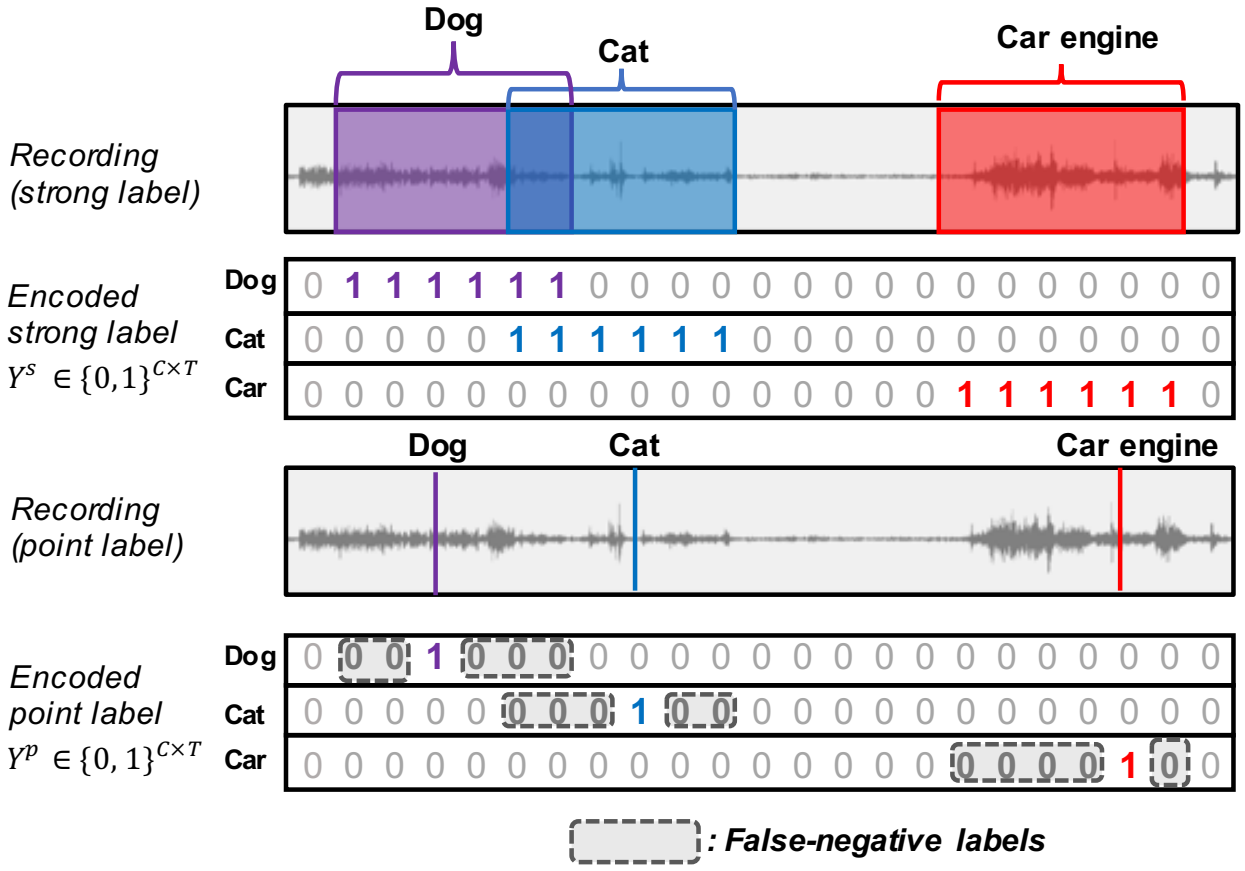


Figure 4.4. Examples of how to encode point labels to compute point-label loss. In this example, three sound events are present at different time location within an audio clip, which result in three point labels at three different time positions. It also illustrates false-negative labels generated by the encoded point labels.

An output of a SED model is denoted as $\hat{Y} \in \mathbb{R}^{C\times T}$ where C is the number of classes and T is the number of time segments. Then point labels are encoded in a way that the encoded label has the same size as the model output. Let $Y^p \in \{0,1\}^{C\times T}$ be the encoded point labels of a recording. If a point label of the event c is located at the t-th time-segment, then 1 is assigned to $Y^p_{c,t}$, otherwise, 0 is assigned to it. Figure 4.4 shows an example of the point label encoding and compares the encoded point labels with strong labels when the

number of sound classes C is 3. The time resolution of encoded point labels depends on configurations of a SED model. Given the model I use for the experiment (see Section 4.4), each time segment of encoded point labels covers 1/3 seconds of an input recording.

The encoded point labels Y^p only contain information of the presence of an event during a short time period, not the absence of an event, which means that some of 0s in Y^p might be false-negative labels. Figure 4.4 also shows an example of false-negative information of encoded point labels. Therefore, the losses between point labels Y^p and a model output $\hat{Y}$ (i.e., an estimate of the label matrix) might mislead the model training. My solution to this problem is to only compute losses at true-positive labels. To do so, I first filter out any prediction probabilities in $\hat{Y}$ that cannot contribute to correct loss computation. The filtered version of model outputs $\hat{Y}^f \in \mathbb{R}^{C \times T}$ is computed as follows

$$\hat{Y}^f = \hat{Y} \odot Y^p, \tag{4.1}$$

which is element-wise multiplication between the model output $\hat{Y}$ and point labels Y^p. Since point labels Y^p contains 0 or 1, element-wise multiplying the model output $\hat{Y}$ by the ground-truth point labels Y^p results in a matrix $\hat{Y}^f$ that has non-zero elements only at points in time where there is a positive point label. The result is that the loss function is only presented with differences between model output and the ground truth at the point labels.

Once $\hat{Y}^f$ is computed, point-label loss L_{point} can be defined as binary-cross entropy between $\hat{Y}^f$ and Y^p. The binary-cross entropy loss is computed as:

$$L_{point} = -\sum_{i=1}^{T}\sum_{c=1}^{C} \hat{Y}_{i,c}^{f} log Y_{i,c}^{p} + (1 - \hat{Y}_{i,c}^{f}) log(1 - Y_{i,c}^{p}), \tag{4.2}$$

where $\hat{Y}_{i,c}^{f}$ is a model output for event class c at ith time step and $Y_{i,c}^{p}$ is a point label of event class c at ith time step within an audio clip.

Since point-loss L_{point} only computes loss on the time-segments where there is a point label that covers a very short time period within an audio clip, most of the audio in a training example is not trained on. Therefore, rather than simply ignore that audio, I combine a weak loss L_{weak} function and point-label loss L_{point} in training. Since the weak loss only cares about the presence or absence of an event within a clip, point-label loss will provide the final loss function with the time information of each event within the clip. The weak loss L_{weak} is binary cross-entropy loss between weak labels, $y \in \{0, 1\}^C$ and clip-level predictions, $\hat{y} \in \mathbb{R}^C$, where C is the number of classes. The L_{weak} is computed as

$$L_{weak} = -\sum_{c=1}^{C} \hat{y}^c log y^c + (1 - \hat{y}^c) log(1 - y^c), \tag{4.3}$$

where $\hat{y}^c$ is the likelihood of event class c and y^c is ground truth label for the clip (i.e. weak label). Weak labels are obtained by treating point labels as clip-level ground truth labels. The clip-level predictions from the model are obtained by applying time frame-wise max pooling operation on the model's output $\hat{Y}$ (i.e., segment-level predictions).

Finally, my SED model is trained by minimizing the following loss:

$$Loss = (1 - \alpha) L_{weak} + \alpha L_{point}, \tag{4.4}$$

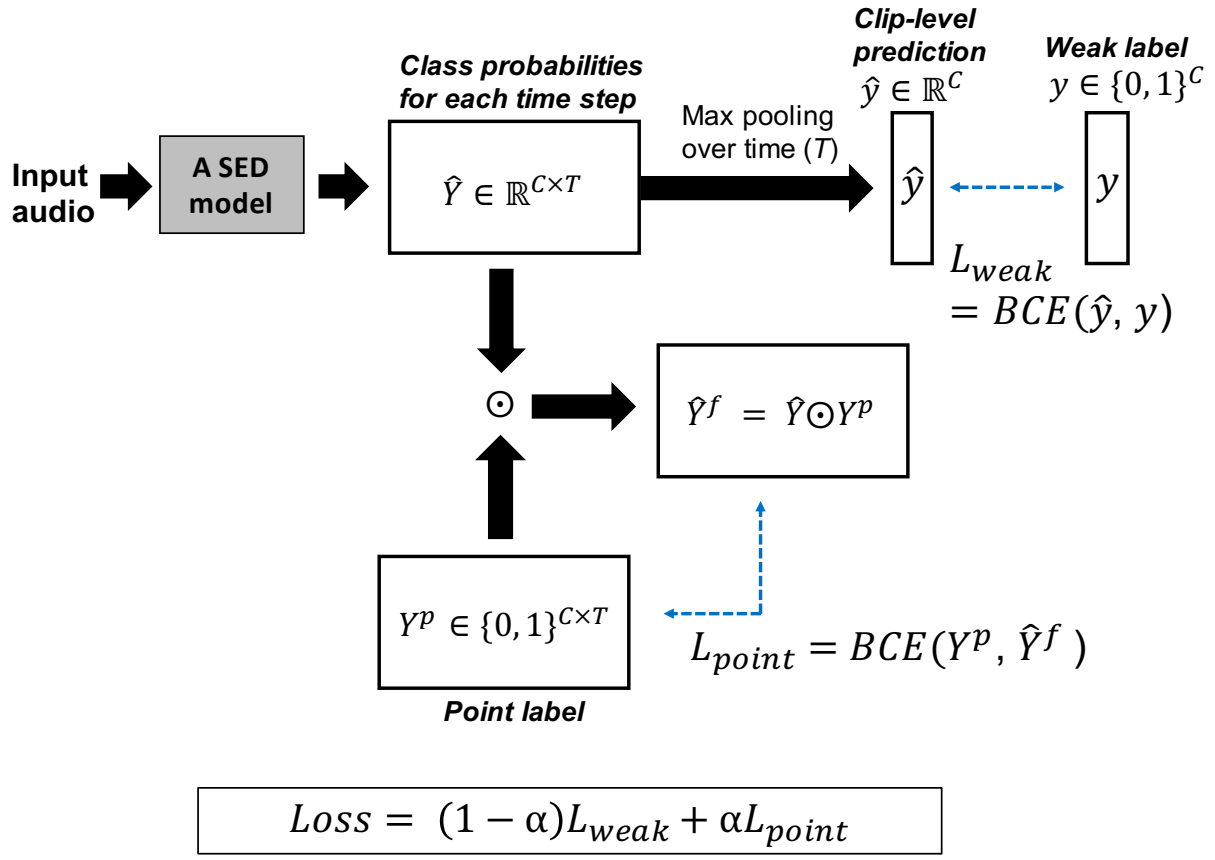


Figure 4.5. Summary of computing losses on point-labeled audio data during model training. It is trained to minimize the combination of weak loss and point-label loss. *BCE refers to Binary-Cross Entropy loss

where α is a hyper-parameter to determine the contribution of each loss to the final loss. The α can be determined through experiments (see Section 4.4.4). Figure 4.5 shows the summary of how the final loss is computed given an audio clip.

4.3.2. Expanding point labels

While point labels contain time information of sound events that weak labels do not have, there is still a gap between strong labels and point labels. A single point label encoded for a SED model only covers a single time-step. For example, given the model architecture used in

the experiment (Section 4.4), one time-step lasts 1/3 of a second. However, many real-world sound events are longer than 1/3 of a second. In this section, I present a method to expand point labels so they can cover a greater portion of a sound event.

The idea is to expand point labels to label more adjacent time frames, which can improve the advantage point labels have over weak labels. However, if I expand a point label too far, I may label adjacent segments where the labeled sound does not occur, creating false-positive "ground truth" labels that would harm learning. Therefore, it is necessary to have a systematic way of determining when to stop the expansion. The idea is to measure similarities between a point labeled segment and its neighbor segments, and copy the point label only to similar neighbor segments.

To measure similarities between segments, I first build a SED model on a training set with only weak labels (i.e., weak model). Note that this training set is the one that the point model is trained on later. The weak model is trained by minimizing only *weak loss* (L_weak in Figure 4.5). Then I apply this *weak* model to label each audio clip in the *training set* at the segment level (1/3 of a second) to obtain segment-level class probabilities for each training example (i.e., $\hat{Y}$). The class probabilities for each segment can be thought of as feature embedding where the similarities between segments can be measured. Figure 4.6 shows an example of the proposed point label expanding method.

For each point-labeled segment at time t, I measure cosine similarity between that segment's class probabilities (S_t) and the class probabilities of the segments immediately before (S_{t-1}) and after it (S_{t+1}). If a neighbor's similarity is above a user-adjustable threshold (0.5 in my experiments), the point label is copied to that neighbor. Then I again compare new neighbor segments (S_{t-2} or S_{t+2}) to the original point-labeled segment (S_t) to decide whether the point label can be copied to the new segments. The expansion is applied to

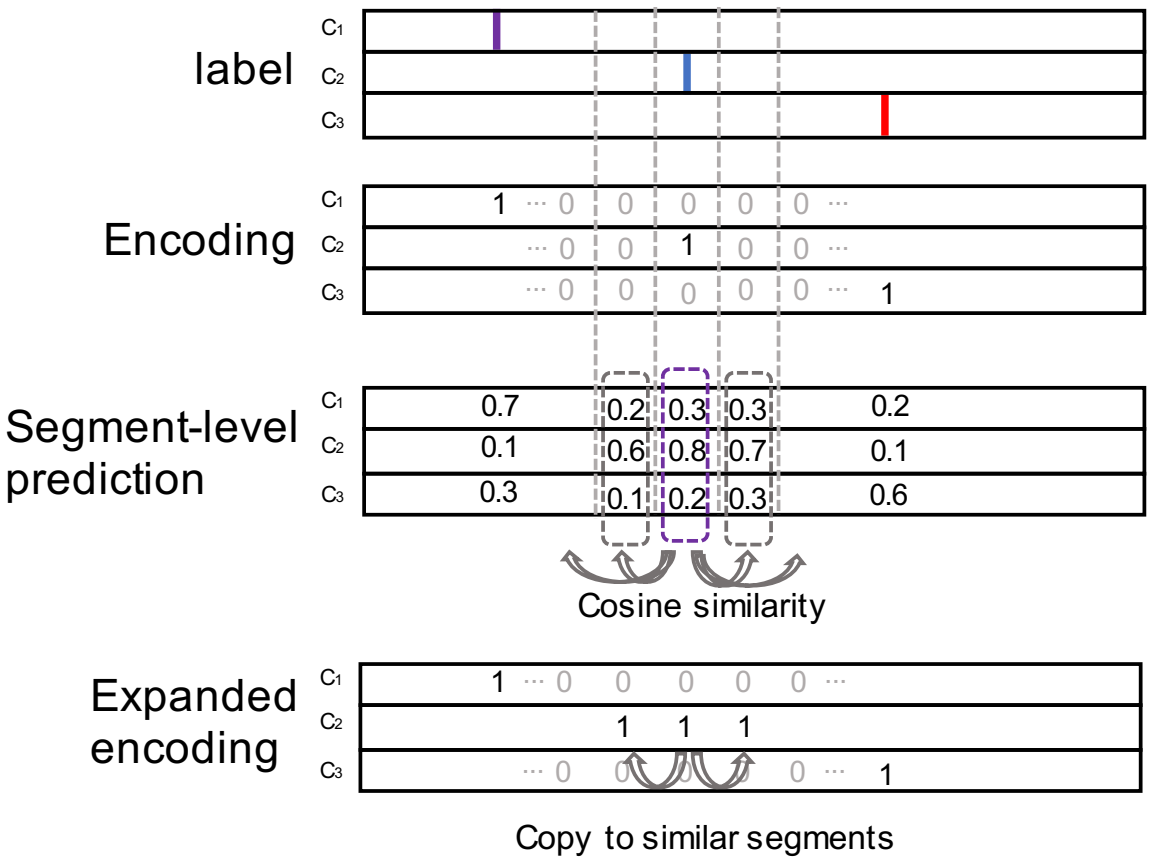


Figure 4.6. An example of expanding point labels to similar neighbor segments.

every new neighbor segment that falls above the similarity threshold, until a segment with similarity less than a certain threshold is found. Once point-labels are expanded, a model can be trained in the same way that how a model is trained on original point labels

This similar segment picking strategy is very useful since the positions of point labels can vary with different human annotators. When evaluating the proposed system in the experiment (Section 4.4), I set the point labels at random locations within an event to reflect the real-world scenario. I will report the experiment results showing that a SED model trained on the expanded version of point labels outperforms one trained on original

point labels. It confirms that the point label update method successfully expands point labels in a way that limits false-positive labels.

4.4. Experiments

I now present an experiment designed to shed light on the efficacy of point labels for providing a training signal to a deep model for sound event detection. If point labels prove more effective than weak labels, this indicates such labels have the potential to be an effective alternative labeling method that combines the advantage of strong labels (accurate labeling at fine time granularity) with weak labels (easy to create).

In the experiments, I train models on point-labeled data, weakly-labeled data and strongly-labeled data and compare the SED performances of them. I will call each of the models *point model*, *weak model*, *strong model* in this section.

The goal of the experiments is to answer the following questions:

- Do point models output performs weak models on sound event detection?
- Are point models comparable to strong models on sound event detection?
- Does the point-label expansion method improve the SED performance of point models trained on single point labels?

4.4.1. Model architecture

Table 4.1 shows the architecture of the model used in the experiments. It is a fully convolutional network (CNN) consisting of 8 convolutional layers. CNN takes as an input image. Therefore, to train a CNN model, an audio clip needs to be transformed into an image. In my experiment, an audio clip at a sampling rate of 16kHz is first transformed into a mel-spectrogram (64 Mel bins, a window size of 25 ms and a hop size of 10 ms). It represents

Table 4.1. Model architecture. *MP: 2D-Max Pooling (kernal size: 2×2, stride: 2), *N: the number of classes in the training dataset (N=10 in our experiment). The output shape column shows the size of tensor from each layer, given a 10-second recording as input.

Layers	**Components**	**Output shape**
Input	Mel-spectrogram	998×64
Layer-1	Conv (3×3, 64) → Relu → MP	499×32, 64
Layer-2	Conv (3×3, 128) → Relu → MP	249×16, 128
Layer-3	Conv (3×3, 256) → Relu	249×16, 256
Layer-4	Conv (3×3, 256) → Relu → MP	124×8, 256
Layer-5	Conv (3×3, 512) → Relu	124×8, 512
Layer-6	Conv (3×3, 512) → Relu → MP	62×4, 512
Layer-7	Conv (2×2, 1024) → Relu → MP	30×1, 1024
Layer-8	Conv (1×1, C) → Sigmoid	30×1, C

how energies of different frequencies of a sound change over time and allow CNNs to learn spectral patterns of different sound events. The model can take a variable length of audio.

The first 6 convolutional layers are the same as VGGish model [**38**] which has proven to be very effective in recent prior works [**40, 51, 46, 45**]. Then the last two layers are newly added to obtain segment-level predictions of an audio clip. I use the same model architecture to learn from weak, strong, and point labels with different loss functions because my focus is not evaluating model architectures, but evaluating the efficacy of point labels compared to strong and weak labels.

In the table, Convolution operations for each layer are denoted as *Conv (the size of filters, the number of filters)*. The number of filters on the last layer depends on the number of classes in the training data. Strides of all the convolutional layers are set to 1. Zero-padding with a size of 1 is applied to layer-1 to 6. All the convolutional layers except for

the last one are followed by Rectified Linear Unit (ReLU) activation. The last layer uses Sigmoid activation to compute the class-wise probability of sound events.

Given a recording, the network outputs a matrix $\hat{Y} \in \mathbb{R}^{C \times T}$ where C is the number of classes and T is the number of time segments, which represents class probabilities for each time segment. T depends on the input audio length. Table 4.1 also shows an example of output shapes from each layer when the network takes a 10-second recording as an example. The network outputs $\hat{Y}$ ($C \times 30$ matrix) where each segment represents class probabilities for 1/3 seconds of audio.

4.4.2. Dataset and point-label generation

4.4.2.1. Dataset. I evaluate models on URBAN-SED dataset (Version 2.0) [1] which contains 10,000 soundscapes generated using the Scaper soundscape synthesis library [**94**]. I chose the dataset because it contains strong labels of all the soundscapes, which enables me to generate point labels as well as weak labels. On top of that, this dataset has been widely used to evaluate recent SED systems [**68, 80, 79, 67**].

Each file in the dataset is 10 seconds long (about 28 hours in total) and contains between 1 to 9 sound events from 10 classes in the UrbanSound8K dataset [**93**]. They are polyphonic soundscapes, which means that sound events within an audio clip can be overlapped each other. The total number of annotated sound events in the dataset is close to 50,000. The 10 sound classes in the dataset are the following: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gunshot, jackhammer, siren, and street music. The dataset is pre-divided into train, validation, and test sets containing 6000, 2000, and 2000 files respectively.

[1] http://urbansed.weebly.com/

4.4.2.2. Point-label generation. To train a model on point-labeled data, I generated point labels for audio clips in training data. When generating point labels, it is important to reflect the real annotation scenario where different human annotators might choose different time locations for a point label. Therefore, I randomly set the time position of point labels within a sound event. No matter where a point is located within an event, I expect that a point label can be properly expanded by the point label expansion technique presented in Section 4.3.2.

In this work, I assume that only a single point label per sound event is collected, although I believe that multiple point labels per event should not hurt the performance and could lead to further performance improvement. In a real-world scenario, depending on the labeling budget, a human annotator might be able to provide more than one point for a single sound event.

4.4.3. Performance metric

To measure the performance of the sound event detection (SED) models, I compute a segment-based F_1 score with a segment granularity of 1 second, which is an official evaluation method in the DCASE challenge [**73**], an annual evaluation of SED models. Figure 4.7 shows an example of computing F_1 score on estimations of labels for a 5-second audio clip with 3 sound classes. The segment-based F_1 score is computed based on the numbers of True-Positive (TP), False-Positive (FP), and False-Negative (FN) values of every class at every second over the testing set. These numbers are first used to compute Precision (P) and Recall (R). Then F_1 score is computed as follows

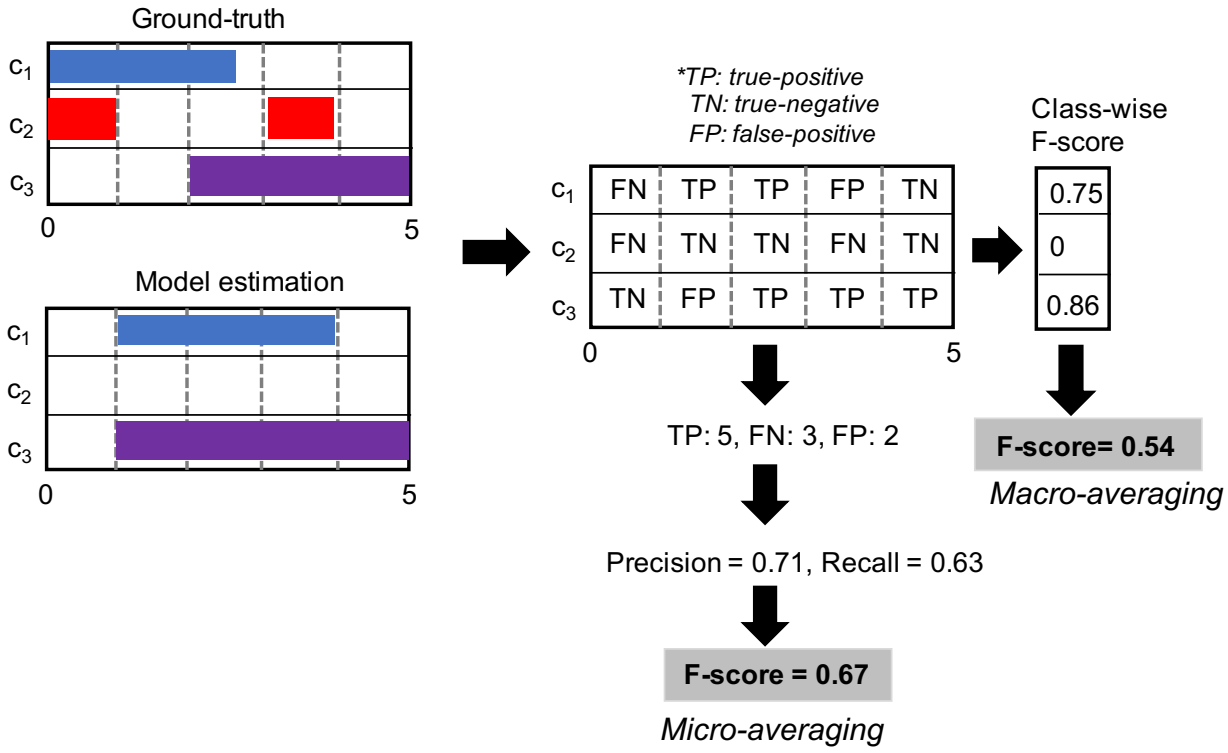


Figure 4.7. An example of computing segment-based F_1 scores with micro and macro averaging on a 5-second audio clip.

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F = \frac{2 \cdot P \cdot R}{P + R} \tag{4.5}$$

F_1 score could vary depending on the choice of averaging. I report two F_1 scores with different averaging methods: micro and macro averaging. In micro averaging, F_1 score is computed for each audio clip and they are averaged over the entire dataset. Macro F_1 score is a class-wise averaging metric. So F_1 score for each sound class is computed over the entire dataset and they are averaged over classes.

4.4.4. Settings for training and testing

Each audio file was resampled to 16kHz mono and represented by a log-scale Mel-spectrogram with 64 Mel bins, a window size of 25 ms and hop size of 10 ms. All models were trained on mini-batches of 32 examples using the Adam optimizer with a learning rate of 0.0001. The training stopped if the model performance on the validation set did not improve for 20 epochs.

I also applied transfer learning because 6,000 training examples are relatively small dataset given the model architecture. When training the models, I initialized the networks with the set of weights from a VGGish pre-trained model [**38**] that has been trained on 3,000 sound classes of 8 million YouTube videos because the VGGish model has been successfully used in recent prior works on transfer learning for sound classification [**40, 51, 46, 45**].

Layers 1 to 6 (see Table 4.1) were initialized with the weights from the VGGish model. The rest of the layers were randomly initialized. Since lower layers of a CNN model capture very basic patterns of audio, pre-trained weights of lower convolutional layers can be used as fixed feature extractors and therefore were not updated during training on the specific dataset used for these experiments. It also helps to reduce the number of trainable parameters, which makes it possible to effectively train models on the limited number of training examples. Through a preliminary experiment, I found fixing the first three layers to be the most effective. The rest of the layers were fine-tuned on the training set. To obtain labels from the network output, I applied the likelihood threshold 0.5 to class probabilities to determine the presences or absences of an event. Figure 4.8 shows an example of how a trained network estimates event activities over time in a test audio example.

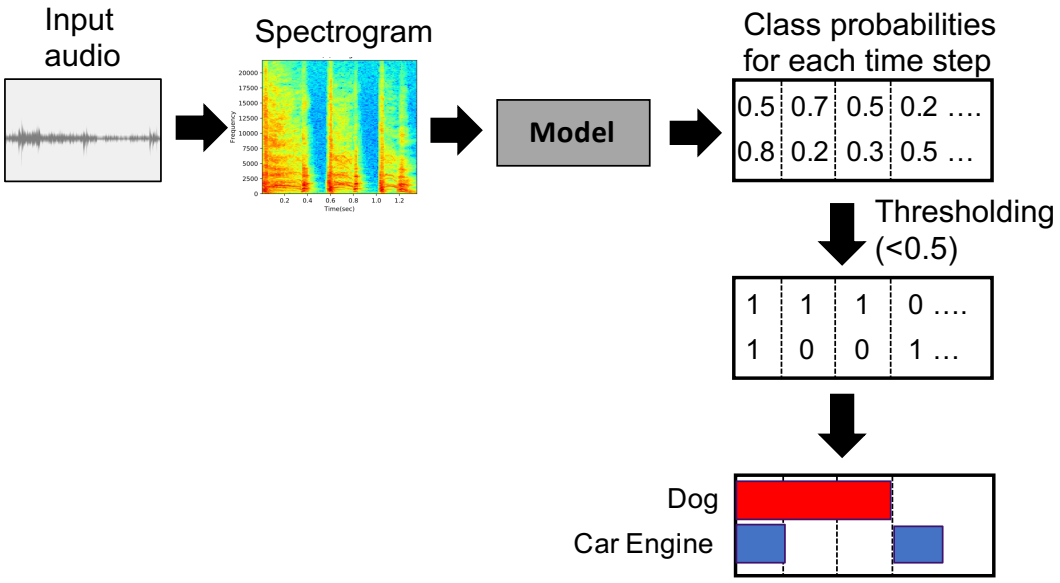


Figure 4.8. An example of how a trained model performs sound event detection given a test audio clip

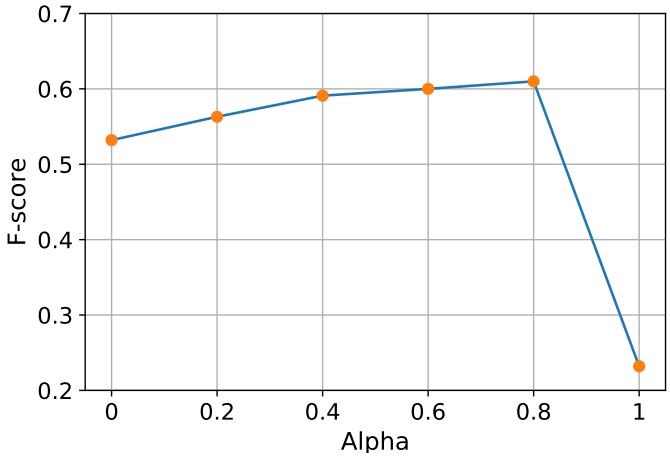


Figure 4.9. SED performance of point models on the validation set with different alpha values.

When point models are trained, the contributions of weak loss and point loss to the final loss function need to be determined (α in equation 4.4). When α is set to 0, only weak losses are considered during training. When α is set to 1, only point losses are computed. I

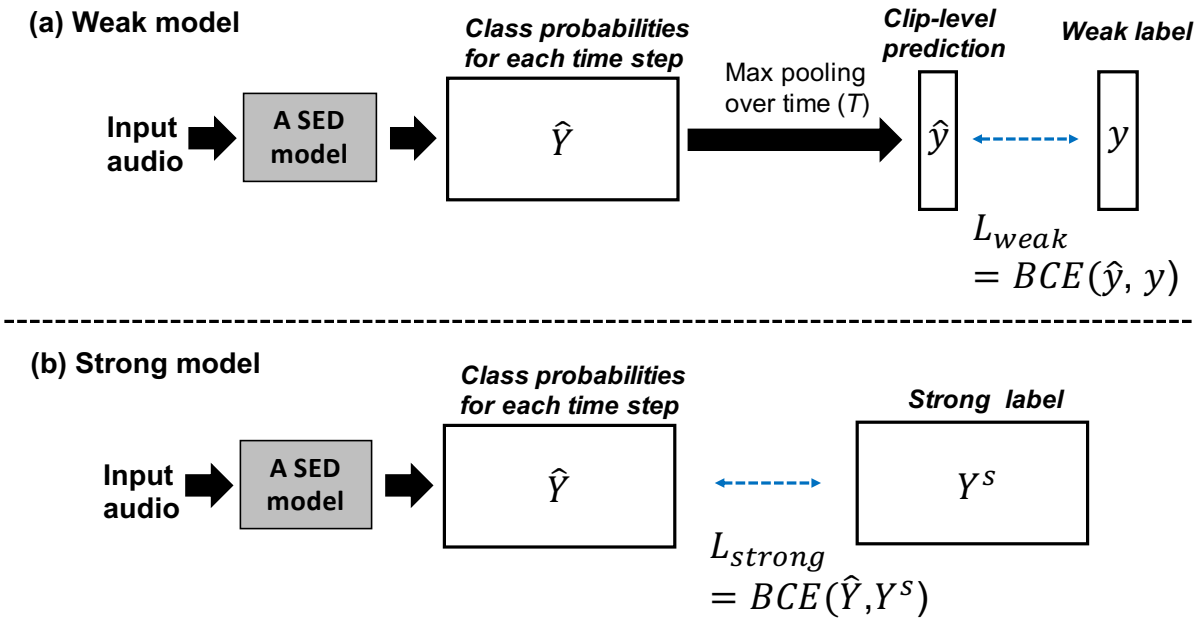


Figure 4.10. Two different ways of computing losses for weak model and strong models. Weak losses L_{weak} is computed by the difference between clip-level predictions and weak labels. Strong losses L_{strong} are computed between segment-level predictions and strong labels.

picked the best α value based on the model performance on the validation set. Figure 4.9 shows F-score of point models on the validation set with different α values. When the loss function is a combination of weak loss and point loss with $\alpha = 0.8$, the model obtained the highest F-score. I set $\alpha = 0.8$ when testing point models on the testing set in the remaining sections.

In addition to the point model, I also trained two other models, a weak model and a strong model to compare the three of them. Figure 4.10 shows how weak losses and strong losses are computed. Weak models are trained by minimizing weak loss L_{weak} which can be thought of as the final loss for a point model with $\alpha = 0$. For strong models, I encode the strong label matrix Y^s in the same way as point label encoding, but with the exact

Table 4.2. F_1 score, precision, and recall (higher is better for all three) for each model. Weak and Strong models are trained by minimizing weak loss and strong loss respectively. All point models are trained by minimizing the loss function (Equation 4.4) which is the combination of weak loss and point loss with α of 0.8. *Micro F_1 scores are not available in [**68**].

	Macro			**Micro**		
Model	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
Weak	0.577	0.803	0.458	0.581	0.795	0.457
Strong	**0.637**	0.681	0.607	**0.638**	0.675	0.605
Point single	0.604	0.768	0.513	0.610	0.763	0.509
Point expanded	**0.636**	0.701	0.596	**0.636**	0.684	0.594
Strong (McFee et al.[**68**])	0.551	0.693	0.458	n/a	n/a	n/a

time boundary information of events and compute loss between $\hat{Y}$ and Y^s. I use binary cross-entropy to compute the loss for weak, strong, and point labels.

4.4.5. Results

I compared two variants of point models. *Point-single* model uses only a single point label at a random time position within a sound event. *Point-expanded* model uses the updated point labels expanded by the proposed methods in Section 4.3.2. Both models were trained by minimizing the loss function with α of 0.8 (see equation 4.4) that showed the best performance on the validation set, as reported in Figure 4.9. I also built a *Weak* model as a baseline and *Strong* model as the best possible model.

Table 4.2 shows F_1 scores with precision and recall. My two point models outperform the weak model regardless of averaging methods (i.e., macro and micro), which shows the point labels help models localize sound events more accurately. Compared to the two point models, the proposed point label expansion improve the performance even further. The

Figure 4.11. Class-wise F_1 scores of weak and point-expanded models.

point expanded model achieved $F_1 = 0.636$ which is nearly identical to the strong model's score ($F_1 = 0.637$ for macro and 0.636 for micro). This performance gain was achieved even though I randomly set the positions of point labels, which proves that the point models are robust to the position of point labeling which might vary in real annotation scenario. To provide the current state of the art as context, I also report results on the same data from a strong recent model by McFee et al. [**68**]. All models showed a better F1 score than McFee et al. I believe this is due to more capable network architecture and transfer learning from a model pre-trained on a much larger dataset. Figure 4.11 compares class-wise F_1 scores of weak and point-expanded models. It shows that the point-expanded model outperforms the weak model for most classes of sound events.

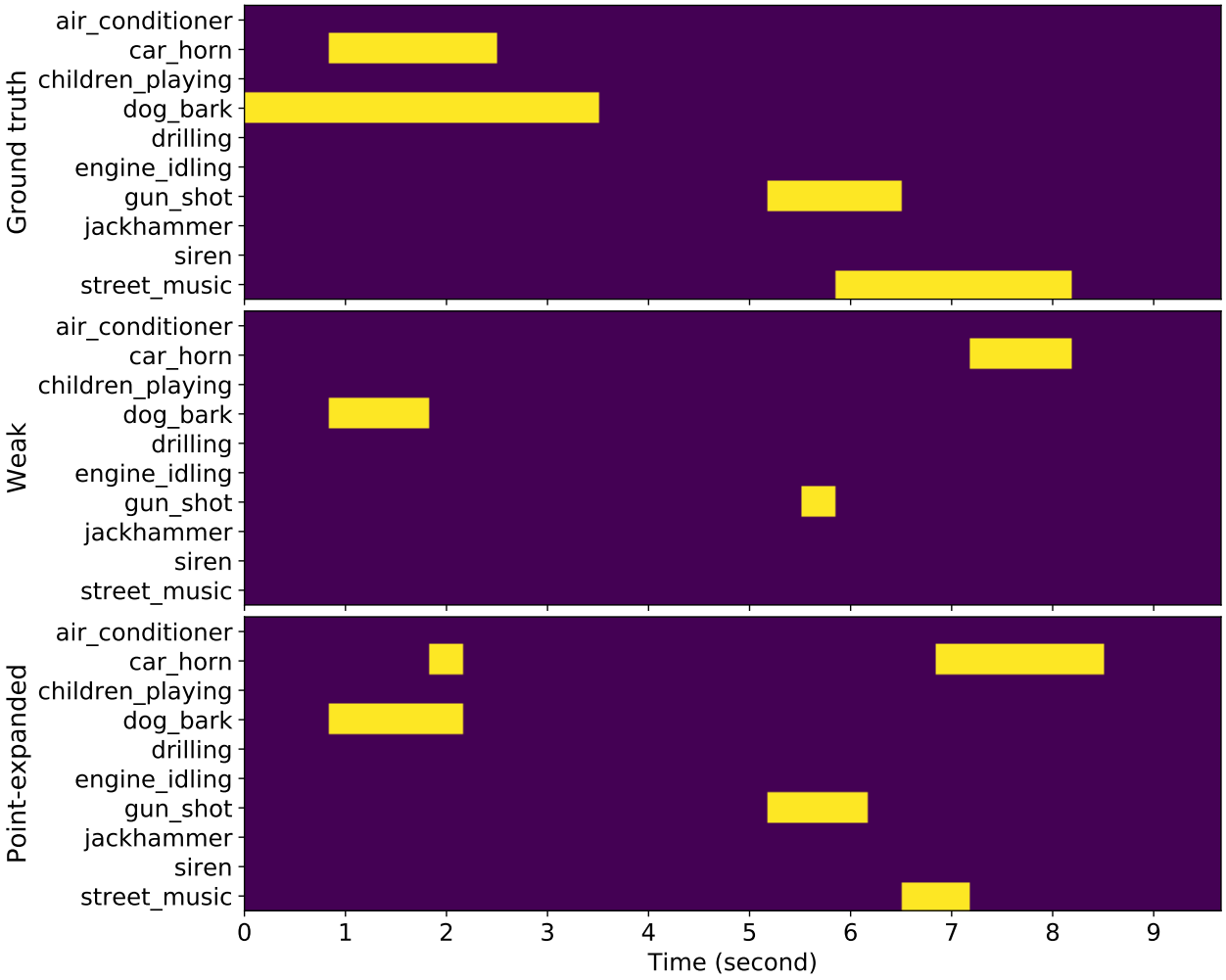


Figure 4.12. An example of SED performed by *weak* model and *Point-expanded* model for a 10-second audio clip in the testing set. In each figure, yellow bars represents event activities of each class over time. The top figure shows the ground-truth labels. The middle and bottom figures shows estimations from weak and point models.

Figure 4.12 visualizes an example of the predictions performed by the weak and point-expanded model given a 10-second recording from the testing set. The recording contains 4 different sound events. The weak model failed to detect *street music*, but the point model successfully detected it. The figure also shows that the point model also made more accurate predictions of temporal boundaries of the identified events. While *gun shot* and *dog bark*

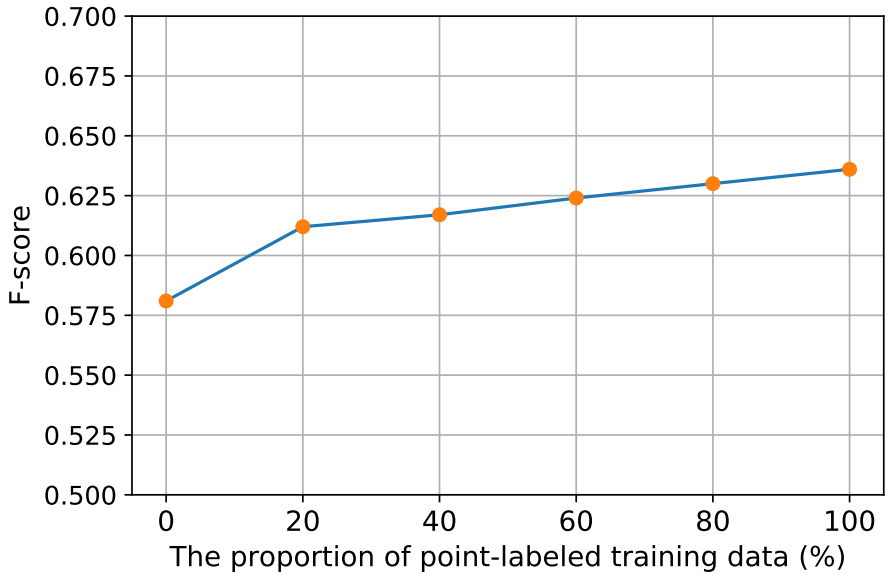


Figure 4.13. SED performance of point models trained on different amount of point-labeled data. The total number of training examples is 6,000. The proportion of point-labeled training data increases by 20% (1,200 examples) from 0 to 100%.

sounds were identified by both models, the point model covers greater portions of their true labels.

All experiments performed to evaluate point models so far assume that point labels are available for all training examples. However, one might already have access to weakly labeled data and just want to add point labels to a subset of exiting training data to improve the model performance. My point model can be trained even when point labels are available only for some amount of training data. This can be done by dynamically setting $\alpha = 0$ in the proposed loss function (equation 4.4) for training examples without point labels during training.

Figure 4.13 shows SED performance of point models with different amounts of point-labeled training data. Staring with only weak labels (0% of point labeled data), I increased

the proportion of point-labeled data by 20% which is equivalent to 1,200 examples (The total number of training examples is 6,000). For example, the proportion 40% means that 40% of training examples have point labels and 60% of them only have weak labels, which is equivalent to 2,400 point-labeled and 3,600 weakly labeled training examples.

As shown in the figure, the performance increases as the proportion of point-labeled data in the training set increases. Interestingly, the largest performance gain was obtained even by adding point labels to 20% of training examples. It confirms that even a small amount of point labeled data also helps to improve SED performance.

4.5. Conclusion

I presented a new type of audio labeling, called *point labeling* which creates more information than weak labels, but still faster to collect than strong labels. While strong labeling requires finding accurate time-boundaries of a sound event, point labels only contain names of sound events at a single time point per sound event instance in a recording. I also presented a new method to train neural networks on point-labeled data for sound event detection. I performed experiments to evaluate the efficacy of point labels for building a sound event detection system using the proposed training method. The results showed that models trained on point-labeled data outperformed one trained on weakly labeled data.

To improve SED performance of point models even further, I presented a strategy to automatically expand point labels so they can cover a greater portion of a sound event. The experiments showed that a model trained with the expanded version of point labels outperformed one built with original point labels. Moreover, it achieved comparable results to a model built on strongly labeled data.

I believe that this chapter of my dissertation showed strong evidence that high-performing SED systems can be built with much less human labeling costs. I expect this work will open new opportunities to researchers who have been working on sound classification and detection with weak supervision.

Point labeling is a new audio annotation method that has not been addressed in prior works on audio labeling. In this work, I synthetically generated the point labels. Therefore, it might be interesting future works to design a new audio annotation tool for point labeling and let users to collect point labels for real audio data. Moreover, it should be worth it evaluating my method on real soundscapes containing a larger number of sound classes.

CHAPTER 5

Conclusion

My research goal is to reduce the human effort required for sound event detection and annotation. A typical way for a human to annotate a sound event of interest in unlabeled audio recordings is simply to listen to the audio until one hears it and then label it with the onset and offset of the event, which makes sound event annotation labor-intensive. To reduce such effort, in this dissertation I presented methods to speed up the sound event annotation process.

My specific goals can be divided into two, in terms of what the annotated data is used for. One might want to label audio to quantify sound events of interest in unlabeled recordings for a direct analysis. In this case, my goal is to build a system that helps a user to find sound events of interest and annotate them quickly. Alternatively, one might need to annotate audio as a precursor to training a machine learning model for automatic sound recognition. My research goal for this situation is to help human annotators spend less time labeling training data, but still build a high-performance machine learning model with less annotation effort.

To achieve these goals, in Chapter 2, I presented a new human-in-the-loop sound search method to speed up human annotation of a long audio recording. I built the first general-purpose sound labeling interface, I-SED, where an interactive machine learning approach is applied to sound event annotation. Starting with a user-selected sound example, the system directs the user's attention to the most promising regions of audio for labeling. The user labels these regions and gives the system feedback by labeling and adjusting region boundaries.

The system learns from this feedback and updates future recommendations for high-interest regions. This interactive process helps a user quickly label target sound events in a long audio file. I performed a human-subject study to evaluate its effectiveness. The result showed that the proposed system lets users find sparsely-distributed target sounds roughly twice as fast as manually labeling the target sounds. The survey response and free-form comments showed that most participants were more satisfied with the interactive annotator than the manual annotator. **I-SED pairs a simple ML model with a human, allowing one to label much more data than would otherwise not be possible, allowing us to bootstrap data-hungry machine learning models with less effort and pointing the way for other mixed-initiative systems that combine the strengths of humans and those of algorithmic approaches.**

In Chapter 3, I addressed the issue that might lead to poor initial query-by-example (QBE) search results when using a QBE system like I-SED, which was presented in Chapter 2. QBE search could fail if a query recording contains multiple overlapping sound events and only one of them is a positive example. To solve the issue, I presented a new way of improving QBE search results using user's vocal imitations which would help a user to collect sound events of interest quickly. A user can simply provide vocal imitations to illustrate what they do (positive imitation) or do *not* want (negative imitation) in a query. No prior works on audio search had used negative vocal imitations. To evaluate the effectiveness of vocal imitations on QBE audio search, I also created *Vocal Imitation Set*, the largest crowd-sourced vocal imitation dataset. I reported experimental results showing that not only positive imitation, but also a negative imitation of a query helps a search system improve the retrieval result. **This work is the first study to show the effectiveness of negative query examples for audio search. This points the way for future**

systems to incorporate negative information (what the user does not want) into multimedia search, in general.

Finally, in Chapter 4, I presented a new type of audio labeling, called *point labeling*, which makes it easier for human annotators to provide ground truth labels to train a machine learning sound event detection system. Point labels provide more information than weak labels, but are still faster to collect than strong labels. While strong labeling requires the human to notate accurate time-boundaries for every sound event, point labels only require the human to label each sound event at a single time point, anywhere within the event. The goal of my work in Chapter 4 is to make it possible to build a high-performance sound event detection model on point-labeled data, since it requires much less effort to collect than strongly labeled data. I presented a new method to train neural networks on point-labeled data for sound event detection. I performed experiments to evaluate the efficacy of point labels for building a sound event detection system using the proposed training method. The results showed that models trained on point-labeled data outperformed one trained on weakly labeled data. To improve sound event detection performance of point models even further, I also presented a strategy to automatically expand point labels so they can cover a greater portion of a sound event. The experiments showed that a model trained with the expanded version of point labels outperformed one built with original point labels. Moreover, it achieved comparable results to a model built on strongly labeled data. **Moving forward, point labeling, when combined with mixed-initiative labeling (as embodied in I-SED), promises to be transformative in reducing the difficulty of human labeling of audio events.**

5.1. Limitations and future work

While the human-in-the-loop audio annotation system, I-SED presented in Chapter 2 helps a user to annotate a long audio recording quickly, the interactive approaches cause some amount of interaction overhead due to the human-machine collaboration, compared to a fully automated system or manual labeling. I addressed interaction overheads of my interface that I found from the user study in Chapter 2. It would be an interesting future work to investigate more types of interaction overhead that can be caused by human-in-the-loop systems. Then, the audio annotation interface can be re-designed to reduce the interaction overhead, allowing an even greater speedup of the annotation process.

A machine learning model in I-SED keeps being updated by user feedback every round of the interactive annotation. However, the only way for a user to know the search ability of the model in I-SED is to listen to audio segments suggested by the machine for each round. It is hard for a user to understand the behavior of the model, which would limit the user's interaction with the system. Recently, an explainable machine learning model has obtained much attention from research communities [**86, 65**]. They have presented methods to have a machine learning model make explainable predictions and incorporate them to model training. I believe that such techniques can be applied to I-SED, allowing a user to better understand model suggestions every round of the human-in-the-loop and provide more detailed feedback to the system.

Another limitation of the current version of I-SED is that it might not be a useful tool if one's goal is to achieve 100% recall without listening to all the audio. To help users find all the target sound events in a recording quickly, the system should give the users some cues which help them to decide when to stop listening to the remaining audio. Therefore, a

possible future version of I-SED might visualize such information so that a user can monitor the status of the model and decide when to stop the annotation process.

I-SED can also be updated to support other types of labeling such as point labeling presented in Chapter 4. The current version of I-SED presented in Chapter 2 was designed to help a user to collect strongly-labeled sound events quickly. Therefore, I-SED assumes that user's positive feedback (i.e., audio segments with positive labels) is strongly labeled positive examples. However, one might want to use the interface to speed up other types of labeling such as point labeling for the purpose of training a SED system. I have shown that a SED system built on point-labeled data is comparable to one trained on strongly labeled data in Chapter 4. Therefore, a possible future direction of my research is to build the next version of I-SED where user's point labeled feedback (e.g., point-labeled positive examples) can be appropriately processed to speed up the labeling task.

The point-label expansion technique presented in Chapter 4 also has a limitation. If a sound event dynamically changes over time, its point label might fail to be expanded to its adjacent segments. This is because the expansion is performed based on the similarity between segments. Moreover, the threshold to determine the expansion is set empirically in the experiment. Therefore, research on systemic ways of setting the threshold or a new way of determining the expansion would be an important future work.

Another limitation of my work is that my experiments were performed on synthetically generated soundscapes with a limited number of sound classes. As future works, it should be worth it evaluating the proposed systems on real soundscapes containing a larger number of sound classes.

I hope that my work in this dissertation will be a valuable resource for researchers and practitioners who are looking for new annotation methods under a limited budget. I expect

my work to facilitate their audio data collection and solve data scarcity which is one of the major bottlenecks for deep learning models. This will eventually increase the range of sound-objects that could be automatically identified by AI systems. Moreover, I also believe that my works will encourage researchers to release a lot of public audio datasets with various types of labels. This will open a lot of new audio research opportunities.

Sound is one of the most important mediums to understand the environment around us. In the future, I expect that AI systems, like humans, can perceive sound-objects, extract meaningful information from them, and make a proper decision. I look forward to exploring new possibilities that were not addressed in this work to improve how people label audio as well as how AI systems understand sound scenes.

References

[1] Amershi, S., Fogarty, J., Kapoor, A., and Tan, D. Examining multiple potential models in end-user interactive concept learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2010), ACM, pp. 1357–1360.

[2] Amershi, S., Lee, B., Kapoor, A., Mahajan, R., and Christian, B. Cuet: Human-guided fast and accurate network alarm triage. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2011), CHI '11, ACM, pp. 157–166.

[3] Arandjelovic, R., and Zisserman, A. Look, listen and learn. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 609–617.

[4] Aytar, Y., Vondrick, C., and Torralba, A. Soundnet: Learning sound representations from unlabeled video. In *Advances in Neural Information Processing Systems* (2016), pp. 892–900.

[5] Bello, J. P., Silva, C., Nov, O., Dubois, R. L., Arora, A., Salamon, J., Mydlarz, C., and Doraiswamy, H. Sonyc: A system for monitoring, analyzing, and mitigating urban noise pollution. *Commun. ACM 62*, 2 (Jan. 2019), 68–77.

[6] Blancas, D. S., and Janer, J. Sound retrieval from voice imitation queries in collaborative databases. In *AES 53rd Conference on Semantic Audio* (London, UK, 27/01/2014 2014), Audio Engineering Society, Audio Engineering Society.

[7] Boersma, P., and Weenink, D. Praat: doing phonetics by computer (version 6.0.37), 2018.

[8] Bogaards, N., Röbel, A., and Rodet, X. Sound analysis and processing with audiosculpt 2. In *International Computer Music Conference (ICMC)* (2004), pp. 1–1.

[9] Bogaards, N., Yeh, C., and Burred, J. J. Introducing asannotation: a tool for sound analysis and annotation. In *ICMC* (2008), pp. 1–1.

[10] Boogaart, C., and Lienhart, R. Audio brush: a tool for computer-assisted smart audio editing. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia* (2006), ACM, pp. 115–124.

[11] Bryan, N. J., Mysore, G. J., and Wang, G. Isse: an interactive source separation editor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2014), ACM, pp. 257–266.

[12] Cai, C. J., Reif, E., Hegde, N., Hipp, J., Kim, B., Smilkov, D., Wattenberg, M., Viegas, F., Corrado, G. S., Stumpe, M. C., et al. Human-centered tools for coping with imperfect algorithms during medical decision-making. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (2019), ACM, p. 4.

[13] Cakir, E., Parascandolo, G., Heittola, T., Huttunen, H., and Virtanen, T. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 25*, 6 (2017), 1291–1303.

[14] Cannam, C., Landone, C., Sandler, M. B., and Bello, J. P. The sonic visualiser: a visualisation platform for semantic descriptors from musical signals. In *ISMIR* (2006), pp. 324–327.

[15] Cartwright, M., Dove, G., Méndez, A. E. M., and Bello, J. P. Crowdsourcing multi-label audio annotation tasks with citizen scientists. In *Proceedings of the ACM on Human-Computer Interaction* (2019).

[16] Cartwright, M., and Pardo, B. Synthassist: an audio synthesizer programmed with vocal imitation. In *Proceedings of the 22nd ACM international conference on Multimedia* (2014), ACM, pp. 741–742.

[17] Cartwright, M., and Pardo, B. Vocalsketch: Vocally imitating audio concepts. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (2015), ACM, pp. 43–46.

[18] Cartwright, M., Seals, A., Salamon, J., Williams, A., Mikloska, S., MacConnell, D., Law, E., Bello, J. P., and Nov, O. Seeing sound: Investigating the effects of visualizations and complexity on crowdsourced audio annotations. *Proceedings of the ACM on Human-Computer Interaction 1*, CSCW (2017), 29.

[19] Chen, S., Chen, J., Jin, Q., and Hauptmann, A. Class-aware self-attention for audio event recognition. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval* (New York, NY, USA, 2018), ICMR '18, ACM, pp. 28–36.

[20] Chou, S.-Y., Jang, J.-S. R., and Yang, Y.-H. Learning to recognize transient sound events using attentional supervision. In *IJCAI* (2018), pp. 3336–3342.

[21] Chung, Y.-A., Wu, C.-C., Shen, C.-H., Lee, H.-Y., and Lee, L.-S. Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder. *arXiv preprint arXiv:1603.00982* (2016).

[22] Cramer, J., Wu, H.-H., Salamon, J., and Bello, J. P. Look, listen, and learn more: Design choices for deep audio embeddings. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), IEEE, pp. 3852–3856.

[23] Dessein, A., Cont, A., and Lemaitre, G. Real-time detection of overlapping sound events with non-negative matrix factorization. In *Matrix Information Geometry*. Springer, 2013, pp. 341–371.

[24] Durrieu, J.-L., and Thiran, J.-P. Musical audio source separation based on user-selected f0 track. In *International Conference on Latent Variable Analysis and Signal Separation* (2012), Springer, pp. 438–445.

[25] Fiebrink, R. Machine learning as meta-instrument: Human-machine partnerships shaping expressive instrumental creation. In *Musical Instruments in the 21st Century*. Springer, 2017, pp. 137–151.

[26] Fiebrink, R. A. *Real-time human interaction with supervised learning algorithms for music composition and performance*. PhD thesis, Princeton, NJ, USA, 2011. AAI3445567.

[27] Fogarty, J., Tan, D., Kapoor, A., and Winder, S. Cueflik: interactive concept learning in image search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2008), ACM, pp. 29–38.

[28] Fuhrmann, F., Haro, M., and Herrera, P. Scalability, generality and temporal aspects in automatic aecognition of predominant musical instruments in polyphonic music. In *ISMIR* (2009), pp. 321–326.

[29] Gemmeke, J. F., Ellis, D. P., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., and Ritter, M. Audio set: An ontology and human-labeled dataset for audio events. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on* (2017), IEEE, pp. 776–780.

[30] Ghias, A., Logan, J., Chamberlin, D., and Smith, B. C. Query by humming: musical information retrieval in an audio database. In *Proceedings of the third ACM international conference on Multimedia* (1995), ACM, pp. 231–236.

[31] Giacinto, G. A nearest-neighbor approach to relevance feedback in content based image retrieval. In *Proceedings of the 6th ACM international conference on Image and video retrieval* (2007), ACM, pp. 456–463.

[32] Gomes, J., Chambel, T., and Langlois, T. Soundslike: movies soundtrack browsing and labeling based on relevance feedback and gamification. In *Proceedings of the 11th european conference on Interactive TV and video* (2013), ACM, pp. 59–62.

[33] Gulluni, S., Essid, S., Buisson, O., and Richard, G. An interactive system for electro-acoustic music analysis. In *ISMIR* (2011), pp. 145–150.

[34] Harma, A., McKinney, M. F., and Skowronek, J. Automatic surveillance of the acoustic activity in our living environment. In *2005 IEEE international conference on multimedia and expo* (2005), IEEE, pp. 4–pp.

[35] Hayashi, T., Komatsu, T., Kondo, R., Toda, T., and Takeda, K. Anomalous sound event detection based on wavenet. In *2018 26th European Signal Processing Conference (EUSIPCO)* (2018), IEEE, pp. 2494–2498.

[36] Heittola, T., Mesaros, A., Eronen, A., and Virtanen, T. Context-dependent sound event detection. *EURASIP Journal on Audio, Speech, and Music Processing 2013*, 1 (2013), 1.

[37] Helén, M., and Virtanen, T. Query by example of audio signals using euclidean distance between gaussian mixture models. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on* (2007), vol. 1, IEEE, pp. I–225.

[38] Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J., and Wilson, K. Cnn architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (March 2017), pp. 131–135.

[39] Houix, O., Monache, S. D., Lachambre, H., Bevilacqua, F., Rocchesso, D., and Lemaitre, G. Innovative tools for sound sketching combining vocalizations and gestures. In *Proceedings of the Audio Mostly 2016* (2016), ACM, pp. 12–19.

[40] Humphrey, E., Durand, S., and McFee, B. Openmic-2018: An open data-set for multiple instrument recognition.

[41] Huq, A., Cartwright, M., and Pardo, B. Crowdsourcing a real-world on-line query by humming system.

[42] Jansen, A., Plakal, M., Pandya, R., Ellis, D. P., Hershey, S., Liu, J., Moore, R. C., and Saurous, R. A. Unsupervised learning of semantic audio representations. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), IEEE, pp. 126–130.

[43] Kabra, M., Robie, A. A., Rivera-Alba, M., Branson, S., and Branson, K. Jaaba: interactive machine learning for automatic annotation of animal behavior. *Nature methods 10*, 1 (2013), 64.

[44] Karimzadehgan, M., and Zhai, C. Improving retrieval accuracy of difficult queries through generalizing negative document language models. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (2011), ACM, pp. 27–36.

[45] Kim, B. Convolutional neural networks with transfer learning for urban sound tagging. Tech. rep., DCASE2019 Challenge, September 2019.

[46] Kim, B., and Ghaffarzadegan, S. Self-supervised attention model for weakly labeled audio event classification. In *2019 27th European Signal Processing Conference (EUSIPCO)* (2019), IEEE, pp. 1–5.

[47] Kim, B., Ghei, M., Pardo, B., and Duan, Z. Vocal imitation set: a dataset of vocally imitated sound events using the audioset ontology. In *Proceedings of the 2018 Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2018)* (2018).

[48] Kim, B., and Pardo, B. I-sed: an interactive sound event detector. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces* (2017), ACM, pp. 553–557.

[49] Kim, B., and Pardo, B. A human-in-the-loop system for sound event detection and annotation. *ACM Transactions on Interactive Intelligent Systems (TiiS) 8*, 2 (2018), 13.

[50] Kim, B., and Pardo, B. Improving content-based audio retrieval by vocal imitation feedback. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (May 2019), pp. 4100–4104.

[51] Kim, B., and Pardo, B. Sound event detection using point-labeled data. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (2019), IEEE, pp. 1–5.

[52] Koizumi, Y., Murata, S., Harada, N., Saito, S., and Uematsu, H. Sniper: Few-shot learning for anomaly detection to minimize false-negative rate with ensured

true-positive rate. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), IEEE, pp. 915–919.

[53] Kong, Q., Xu, Y., Sobieraj, I., Wang, W., and Plumbley, M. D. Sound event detection and time–frequency segmentation from weakly labelled data. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) 27*, 4 (2019), 777–787.

[54] Kong, Q., Xu, Y., Wang, W., and Plumbley, M. D. Audio set classification with attention model: A probabilistic perspective. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), IEEE, pp. 316–320.

[55] Kong, Q., Yu, C., Xu, Y., Iqbal, T., Wang, W., and Plumbley, M. D. Weakly labelled audioset tagging with attention neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 27*, 11 (2019), 1791–1802.

[56] Kubat, R., DeCamp, P., Roy, B., and Roy, D. Totalrecall: visualization and semi-automatic annotation of very large audio-visual corpora. In *ICMI* (2007), vol. 7, pp. 208–215.

[57] Kumar, A., Khadkevich, M., and Fügen, C. Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), IEEE, pp. 326–330.

[58] Kumar, A., and Raj, B. Audio event and scene recognition: A unified approach using strongly and weakly labeled data. In *2017 International Joint Conference on Neural Networks (IJCNN)* (2017), IEEE, pp. 3475–3482.

[59] Lallemand, I., Schwarz, D., and Artières, T. Content-based retrieval of environmental sounds by multiresolution analysis. In *SMC2012* (2012), pp. 1–1.

[60] Lavner, Y., and Ruinskiy, D. A decision-tree-based algorithm for speech/music classification and segmentation. *EURASIP Journal on Audio, Speech, and Music Processing 2009* (2009), 2.

[61] Lee, C.-H., Han, C.-C., and Chuang, C.-C. Automatic classification of bird species from their sounds using two-dimensional cepstral coefficients. *Audio, Speech, and Language Processing, IEEE Transactions on 16*, 8 (2008), 1541–1550.

[62] Lemaitre, G., Houix, O., Voisin, F., Misdariis, N., and Susini, P. Vocal imitations of non-vocal sounds. *PloS one 11*, 12 (2016), e0168167.

[63] Lemaitre, G., and Rocchesso, D. On the effectiveness of vocal imitations and verbal descriptions of sounds. *The Journal of the Acoustical Society of America 135*, 2 (2014), 862–873.

[64] Li, B., Burgoyne, J. A., and Fujinaga, I. Extending audacity for audio annotation. In *ISMIR* (2006), pp. 379–380.

[65] Liu, F., and Avci, B. Incorporating priors with feature attribution on text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), pp. 6274–6283.

[66] Ma, Y., and Lin, H. A multiple relevance feedback strategy with positive and negative models. *PloS one 9*, 8 (2014), e104707.

[67] Martín-Morató, I., Mesaros, A., Heittola, T., Virtanen, T., Cobos, M., and Ferri, F. J. Sound event envelope estimation in polyphonic mixtures. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), IEEE, pp. 935–939.

[68] McFee, B., Salamon, J., and Bello, J. P. Adaptive pooling operators for weakly labeled sound event detection. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) 26*, 11 (2018), 2180–2193.

[69] Mehrabi, A. *Vocal imitation for query by vocalisation*. PhD thesis, Queen Mary University of London, 2018.

[70] Mehrabi, A., Dixon, S., and Sandler, M. B. Vocal imitation of synthesised sounds varying in pitch, loudness and spectral centroid. *The Journal of the Acoustical Society of America 141*, 2 (2017), 783–796.

[71] Meléndez-Catalán, B., Molina, E., and Gómez, E. Bat: An open-source, web-based audio events annotation tool. In *Proceedings of 3rd Web Audio Conference* (2017).

[72] Mesaros, A., Diment, A., Elizalde, B., Heittola, T., Vincent, E., Raj, B., and Virtanen, T. Sound event detection in the dcase 2017 challenge. *IEEE/ACM Transactions on Audio, Speech and Language Processing* (2019).

[73] Mesaros, A., Heittola, T., and Virtanen, T. Metrics for polyphonic sound event detection. *Applied Sciences 6*, 6 (2016), 162.

[74] Mettes, P., Koelma, D. C., and Snoek, C. G. The imagenet shuffle: Reorganized pre-training for video event detection. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval* (2016), ACM, pp. 175–182.

[75] Morrison, M., and Pardo, B. Otomechanic: Auditory automobile diagnostics via query-by-example. In *Proceedings of the 2019 Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2019)* (2019).

[76] Nakano, T., Koyama, Y., Hamasaki, M., and Goto, M. Autocomplete vocal-fo annotation of songs using musical repetitions. In *Proceedings of the 24th International Conference on Intelligent User Interfaces: Companion* (2019), ACM, pp. 71–72.

[77] Orr, D. B., Friedman, H. L., and Williams, J. C. Trainability of listening comprehension of speeded discourse. *Journal of educational psychology 56*, 3 (1965), 148.

[78] Ozerov, A., Févotte, C., Blouet, R., and Durrieu, J.-L. Multichannel nonnegative tensor factorization with structured constraints for user-guided audio source separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on* (2011), IEEE, pp. 257–260.

[79] Pankajakshan, A., Bear, H., and Benetos, E. Onsets, activity, and events: A multi-task approach for polyphonic sound event modelling. In *Proceedings of the 2018 Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2019)* (2019).

[80] Pankajakshan, A., Bear, H. L., and Benetos, E. Polyphonic sound event and sound activity detection: A multi-task approach. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (2019), IEEE, pp. 323–327.

[81] Parascandolo, G., Huttunen, H., and Virtanen, T. Recurrent neural networks for polyphonic sound event detection in real life recordings. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016), IEEE, pp. 6440–6444.

[82] Peeters, G. A large set of audio features for sound description (similarity and classification) in the cuidado project. *Technical report, IRCAM* (2004).

[83] Pons Puig, J., Nieto, O., Prockup, M., Schmidt, E. M., Ehmann, A. F., and Serra, X. End-to-end learning for music audio tagging at scale. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018; 2018 Sep 23-27; Paris, France. p. 637-44.* (2018), International Society for Music Information Retrieval (ISMIR).

[84] Portelo, J., Bugalho, M., Trancoso, I., Neto, J., Abad, A., and Serralheiro, A. Non-speech audio event detection. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing* (2009), IEEE, pp. 1973–1976.

[85] Reynolds, D. A., Quatieri, T. F., and Dunn, R. B. Speaker verification using adapted gaussian mixture models. *Digital signal processing 10*, 1 (2000), 19–41.

[86] Ribeiro, M. T., Singh, S., and Guestrin, C. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (2016), pp. 1135–1144.

[87] Rocchesso, D., Mauro, D. A., and Drioli, C. Organizing a sonic space through vocal imitations. *Journal of the Audio Engineering Society 64*, 7/8 (2016), 474–483.

[88] Rocchio, J. Relevance feedback in information retrieval. In *The SMART Retrieval System, Experiments in Automatic Document Processing* (1971), pp. 313–323.

[89] Roma, G., and Serra, X. Querying freesound with a microphone. In *Proceedings of the First Web Audio Conference (Ircam, Paris, France), submission* (2015), vol. 39.

[90] Rother, C., Kolmogorov, V., and Blake, A. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)* (2004), vol. 23, ACM, pp. 309–314.

[91] Rui, Y., Huang, T. S., Ortega, M., and Mehrotra, S. Relevance feedback: a power tool for interactive content-based image retrieval. *IEEE Transactions on circuits and systems for video technology 8*, 5 (1998), 644–655.

[92] Russakovsky, O., Li, L.-J., and Fei-Fei, L. Best of both worlds: human-machine collaboration for object annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 2121–2131.

[93] Salamon, J., Jacoby, C., and Bello, J. P. A dataset and taxonomy for urban sound research. In *22nd ACM International Conference on Multimedia (ACM-MM'14)* (Orlando, FL, USA, Nov. 2014), pp. 1041–1044.

[94] Salamon, J., MacConnell, D., Cartwright, M., Li, P., and Bello, J. P. Scaper: A library for soundscape synthesis and augmentation. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (2017), IEEE, pp. 344–348.

[95] Saraswathy, J., Hariharan, M., Yaacob, S., and Khairunizam, W. Automatic classification of infant cry: A review. In *2012 International Conference on Biomedical Engineering (ICoBE)* (Feb 2012), pp. 543–548.

[96] Settles, B. Active learning literature survey. Tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 2009.

[97] Settles, B. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the conference on empirical methods in natural language processing* (2011), Association for Computational Linguistics, pp. 1467–1478.

[98] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)* (2015).

[99] Stowell, D., Giannoulis, D., Benetos, E., Lagrange, M., and Plumbley, M. D. Detection and classification of acoustic scenes and events. *Multimedia, IEEE Transactions on 17*, 10 (2015), 1733–1746.

[100] Stowell, D., Wood, M., Stylianou, Y., Glotin, H., et al. Bird detection in audio: a survey and a challenge. In *Proceedings of the 26th IEEE International Workshop on Machine Learning for Signal Processing* (2016), IEEE Computer Society, pp. 1–6.

[101] Talbot, J., Lee, B., Kapoor, A., and Tan, D. S. Ensemblematrix: interactive visualization to support machine learning with multiple classifiers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2009), ACM, pp. 1283–1292.

[102] Thomee, B., and Lew, M. S. Interactive search in image retrieval: a survey. *International Journal of Multimedia Information Retrieval 1*, 2 (2012), 71–86.

[103] Turpault, N., Serizel, R., Salamon, J., and Shah, A. P. Sound event detection in domestic environments with weakly labeled data and soundscape synthesis. In *Proceedings of the 2019 Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2019)* (2019).

[104] Valenzise, G., Gerosa, L., Tagliasacchi, M., Antonacci, F., and Sarti, A. Scream and gunshot detection and localization for audio-surveillance systems. In *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on* (2007), IEEE, pp. 21–26.

[105] Vijayanarasimhan, S., and Grauman, K. Multi-level active prediction of useful image annotations for recognition. In *Advances in Neural Information Processing Systems* (2009), pp. 1705–1712.

[106] Vijayanarasimhan, S., and Grauman, K. Large-scale live active learning: Training object detectors with crawled data and crowds. *International Journal of Computer Vision 108*, 1-2 (2014), 97–114.

[107] Vinay, V., Wood, K., Milic-Frayling, N., and Cox, I. J. Comparing relevance feedback algorithms for web search. In *Special interest tracks and posters of the 14th international conference on World Wide Web* (2005), pp. 1052–1053.

[108] Von Ahn, L., and Dabbish, L. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2004), ACM, pp. 319–326.

[109] Vuegena, L., Van Den Broeckb, B., Karsmakersb, P., Vanrumsteb, J. G. B., et al. An mfcc-gmm approach for event detection and classification. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA).* (2013), pp. 1–3.

[110] Wang, H., Gong, S., Zhu, X., and Xiang, T. Human-in-the-loop person re-identification. In *European conference on computer vision* (2016), Springer, pp. 405–422.

[111] Wang, X.-Y., Zhang, B.-B., and Yang, H.-Y. Active svm-based relevance feedback using multiple classifiers ensemble and features reweighting. *Engineering Applications of Artificial Intelligence 26*, 1 (2013), 368–381.

[112] Wu, E., and Zhang, A. A feature re-weighting approach for relevance feedback in image retrieval. In *Image Processing. 2002. Proceedings. 2002 International Conference on* (2002), vol. 2, IEEE, pp. II–581.

[113] Xu, Z., and Akella, R. Active relevance feedback for difficult queries. In *Proceedings of the 17th ACM conference on Information and knowledge management* (2008), pp. 459–468.

[114] Xue, J., Wichern, G., Thornburg, H., and Spanias, A. Fast query by example of environmental sounds via robust and efficient cluster-based indexing. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on* (2008), IEEE, pp. 5–8.

[115] Ye, G., Li, Y., Xu, H., Liu, D., and Chang, S.-F. Eventnet: A large scale structured concept library for complex event detection in video. In *Proceedings of the 23rd ACM international conference on Multimedia* (2015), ACM, pp. 471–480.

[116] Yimam, S. M., Biemann, C., Majnaric, L., Šabanović, Š., and Holzinger, A. Interactive and iterative annotation for biomedical entity recognition. In *International Conference on Brain Informatics and Health* (2015), Springer, pp. 347–357.

[117] Zhang, Y., and Duan, Z. Supervised and unsupervised sound retrieval by vocal imitation. *Journal of the Audio Engineering Society 64*, 7/8 (2016), 533–543.

[118] Zhang, Y., and Duan, Z. Iminet: Convolutional semi-siamese networks for sound search by vocal imitation. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (2017), IEEE, pp. 304–308.

[119] Zhang, Y., and Duan, Z. Visualization and interpretation of siamese style convolutional neural networks for sound search by vocal imitation. In *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on* (2018), IEEE.

[120] Zhang, Y., Pardo, B., and Duan, Z. Siamese style convolutional neural networks for sound search by vocal imitation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 27*, 2 (2018), 429–441.