Carnegie Mellon University

Computational Audition with Imprecise Labels

Ankit Parag Shah

CMU-LTI-24-018
October, 2024

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis committee:
Prof. Bhiksha Raj (Chair)
Prof. Rita Singh (Co-Chair)
Prof. Shinji Watanabe
Dr. Anurag Kumar, Meta
Dr. Jonathan Le Roux, MERL

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Language and Information Technology.

Keywords: Audio Understanding, Computational Audition, Machine Listening, Imprecise Label learning, Weak Label learning, Artificial Intelligence

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my advisors, Professor Bhiksha Raj and Professor Rita Singh, for their invaluable guidance, support, and encouragement throughout the course of my doctoral journey. Their expertise, patience, and insightful feedback have been instrumental in shaping this work and my academic growth. Professor Bhiksha Raj's mentorship has been a steadfast source of inspiration for over a decade—starting from the early days at the NITK CMU Winterschool to guiding me through every step of the Ph.D. program. His unwavering belief in my abilities, generous sharing of knowledge, and consistent mentorship transformed countless challenges into opportunities for personal growth. He provided not only technical expertise but also the confidence and perspective needed to keep pushing boundaries. Professor Rita Singh's incisive and constructive feedback challenged me to refine my ideas, improve my rigor, and never settle for less than my best. Her high standards and insightful perspectives have made me a more thorough researcher, while her motivational support helped me maintain resilience during the most demanding phases of the project.

I would also like to express my sincere gratitude to my thesis committee members, Anurag Kumar, Prof. Shinji Watanabe and Dr. Jonathan LeRoux. I am indebted for their valuable feedback, comments, and technical discussions on my dissertation. Professor Shinji Watanabe's feedback on my presentations, as well as the resources and guidance he provided, significantly helped me hone my communication skills. His influence extended beyond the research itself, playing a key role in shaping my ability to effectively convey complex ideas and become a more confident and articulate communicator. Dr. Jonathan LeRoux's critical insights and thorough questions prompted me to delve deeper, ensuring a more robust and substantial body of work. Anurag Kumar's timely inputs and constructive suggestions enriched both the content and the scope of my research. Their collective wisdom has not only improved this dissertation but will continue to guide my future research endeavors. I extend my heartfelt thanks to Professor Alexander Hauptmann, whose mentorship and support during my Master's degree enabled me to pursue research in Language Technologies and multimedia analysis. His advice and feedback have had a lasting impact on my research journey.

I am blessed to have my family's unwavering support. To my parents, Parag and Ragini Shah: you have been my backbone throughout this journey, making incredible sacrifices to ensure my success. Your unwavering love and confidence in my abilities have sustained me during challenging times. I am forever indebted to you.

I am grateful to the Language Technologies Institute (LTI) at Carnegie Mellon University for providing a vibrant and intellectually stimulating environment to conduct my research. Special thanks are due to Stacey Young, Kate Schaich, and Nick Hernandez for their unwavering support with administrative tasks, paperwork, and visa-related matters. Their efficient handling of logistics allowed me to focus on my research, reducing stress and making my doctoral experience much smoother.

A big thank you goes to my colleagues and collaborators, including the members of the Robust Audio and Speech Processing Lab, for their camaraderie, insightful discussions, and unwavering support. Working alongside such talented individuals has

been a privilege and a source of inspiration. Many thanks to Anurag Kumar, Benjamin Elizalde, Chirag Nagpal, Hao Chen, Roshan Sharma, Hira Dhamyal, Muhammad Ahmed Shah, Yang Gao, Yandong Wen, Raphael Oliver, Rohan Badlani, Mahmoud Alismail, Soham Deshmukh, Oscar Chang, Justin Salamon, Romain Serizel, Nicolas Turpault, Olivier Siohan, Chiori Hori, Taakaki Hori, Dmitriy Serdyuk, Avner May, and many others for their invaluable collaborations, feedback, and encouragement over the years.

Reflecting on this journey, it feels apt to remember that, *"Success is not final, failure is not fatal: it is the courage to continue that counts."* The Ph.D. path is filled with moments when the work seems impossible. Yet, as Nelson Mandela said, *"It always seems impossible until it's done."* What once felt out of reach eventually became a reality, thanks to the guidance, patience, and encouragement of those who supported me. Over the years, I have grown from a curious student into a confident researcher who is ready to embrace the next challenge. I have learned that perseverance, resilience, and the willingness to learn from one's mentors and peers are what truly define success in the realm of research.

Abstract

Sounds are essential to our physical environment and play a critical role in allowing us to interact with it effectively. Throughout our lives, we develop the ability to interpret and understand the myriad sounds surrounding us, allowing us to navigate and function seamlessly within our environments. The goal of computational audio processing or computational audition is to accurately emulate this ability of the human brain (and even that of animals). At the broadest level, this thesis explores the challenge of teaching machines to interpret and understand the acoustic landscape.

For machines to accurately interpret the acoustic environment, they must be able to distinguish all types of sounds. However, the range of possible sounds in this world is vast and their complete variety is unknown. Current audio understanding and interpretation approaches are constrained to recognizing a limited subset of "known" sounds (or sound events) in digital audio recordings. Even within the subset of known sounds, current modeling approaches for detecting and interpreting them require large labeled datasets to achieve good performance. In real-world scenarios, the scarcity of labeled data often hampers the effectiveness of supervised learning algorithms, limiting their scalability and applicability.

A central contribution of this thesis is the development of methods for detecting known sound events without the need for extensive and accurately labeled data. Models trained in the absence of accurately labeled data often perform suboptimally, regardless of the size of the dataset used. This work addresses the problem of formulating effective strategies for modeling sound with imprecisely labeled training data. In other words, we address the problem of building accurate models from weakly labeled data. The term weak labels also refers to labels that only indicate the presence or absence of an event without specifying the exact temporal boundaries of an event.

Accurate labeling of large datasets is both time-consuming and costly and is sensitive to human judgment, thus requiring trained annotators with domain knowledge. Unfortunately, large training datasets almost always contain examples with inaccurate or imprecise labels. Inaccurate labels are also called "noisy" labels. Algorithms trained using noisy labels typically underperform in detection tasks. In addition to label noise, which comprises inaccuracies in label identity and quality, there are often also additional sources of noise that result in the degradation of model performance. These stem from poor signal quality, which can result from many factors (e.g., coding, channel, and transmission errors). In the presence of weak labels and signal noise, we encounter a paradoxical situation where providing more data to deep networks or other data-driven learning frameworks actually degrades performance and introduces biases that are essentially tantamount to memorizing training label noise patterns.

One of the solutions explored in this thesis is to strengthen the available annotations. However, it is also equally, if not more important to develop techniques that impose less stringent labeling requirements on training data, while allowing the models to learn better. For the large part, we focus on these in our work. Our specific strategies include the following:

1. Studying the effects of label noise, label corruption, and label density in the process of learning with weak labels.

2. Building a novel co-training approach to learn sound events from the web (internet) data without any human labeling (we show that our solution results in significant improvement over models that are trained directly with weakly labeled data).

3. Developing strategies to exploit additional cues that add negligible annotation or computational overhead. Examples of such cues are counts of sound events, proportions of the sound events (in terms of occurrence), duration of sound events, etc. We refer to such cue-strengthened labels as semi-weak labels.

4. Developing strategies based on the notion of *negatives* for sound labels (weak or strong), including exploiting all available information in the recording to complement or contrast the target label.

5. Improving audio segmentation using imprecise labels.

6. Examining the importance of the proximity of counts in a bag for semi-weak label learning by controlled factoring of the distribution of the *count* of weak labels, we examine how such factoring of the recordings influences learning.

As a final major advancement in this thesis, we also address the problem of learning *without* labels. We devise a novel method to perform unsupervised adaptation on multimedia datasets as a partial solution to this problem. The strategy we propose also generates new labels for the data, using which we are in the process of creating what we believe is currently the largest dataset of sound events with labels that have been computed in a completely unsupervised fashion.

Furthermore, we explore the feasibility of devising a unified framework to encapsulate all aforementioned approaches, namely learning with weak labels, learning without labels, combining weak and strong labels, using weak labels with additional cues (semi-weak), learning in the presence of noise, etc. In this unified framework, all these strategies will become special cases that can be invoked as necessary during the modeling.

This thesis establishes a robust and coherent foundation for future research in computational audition, offering innovative approaches for modeling and understanding the world of sounds under varying degrees of label precision, label quantity, and data quality.

Contents

List of Figures

List of Tables

1

Introduction

Imagine standing on a street corner in the city. With your eyes closed, you can hear and recognize a succession of sounds: cars passing by, people speaking, their footsteps when they walk by, and the continuously falling rain. Recognition of all these sounds and interpretation of the perceived scene as a city street soundscape come naturally to humans. However, it is the result of years of "training": encountering and learning associations between the wide variety of sounds in everyday life, the sources that produce these sounds, and the names given to them.

Our everyday environment consists of many sound sources that create a complex mixture of audio signals. Human auditory perception is highly specialized in the segregation of sound sources and the direction of attention to the source of the sound of interest. The goal of automatic sound event detection (SED) methods is to recognize what is happening in an audio signal and when it is happening in a similar manner.

Sounds are an essential component of our physical environment. They convey important information about our surroundings and enable us to perceive and interpret events that occur around us. Our perceptions of sound are multi-tiered – we perceive entire sound scenes (i.e. overall soundscapes such as the acoustic environment of an airport or inside a house) as well as individual sound events (e.g. car honks, footsteps, speech, etc.). *Interpreting* the acoustic environment around us requires us to be able to distinguish, or discriminate between, all types of sound.

The term "sound event detection" refers to the detection of sound events within an audio recording, or the task of identifying and temporally locating the occurrences of different types of sounds, and recognizing the categories they belong to. Some common examples of sound events include speech, music, laughter, or the sound of a door closing. Sound event detection is an important area of research and is used in many applications such as noise monitoring in smart cities [6, 7], surveillance [8], urban planning [6], multimedia information retrieval [9, 10], in domestic applications such as smart homes, in health monitoring systems and home security solutions [11, 12, 13] etc., to name a few. In recent years, this area has gained increasing focus from the broader machine learning, artificial intelligence, and audio processing research communities.

Machine learning models typically need large, labeled datasets to achieve state-of-the-art performance. This presents a bottleneck: supervised learning approaches do not scale well to real-world scenarios due to a lack of labeled data. Labeling large datasets accurately requires a significant investment of time and money. Furthermore, skilled annotators with domain expertise are required for the task, and the accuracy and completeness with which they can label data is limited by their judgment and biases. Hand-labeled data also pose challenges in building applications in a systematic, privacy-preserving, or compliant manner [14, 15].

For audio and video data, in addition to the class label, it is also necessary to identify the time intervals within which the sound occurs; *i.e.* accurate temporal labels are required during the model learning processes. This is true even for the most current machine learning and deep learning-based models: their success largely depends on the quality of labeled training data used to train them. The presence of label errors (label noise) in the training data can greatly reduce the accuracy of even the best of these models. Unfortunately, large training datasets almost always contain inaccurate or incorrect labels [16, 17]. These tend to be memorized by most models, resulting in relatively poor model performance in practice [18].

In light of the considerations above, it is useful to develop techniques that impose less stringent labeling requirements on the data while still achieving good performance. Labels that are not required to be very specific, or temporally precise are generally much easier to generate (automatically or by humans).

A popular approach for this is the use of "weak" labels. The term "weak labels" generally refers to labels that are assigned to "bags" (i.e., sets or collections) of instances. A bag is labeled positive for a class if at least one instance in the bag is positive for the class and is labeled negative otherwise.

In the context of audio analysis (originally proposed in [19]), weak labels are defined as those that provide information about the presence or absence of a sound event or

class in a recording but do not provide finer granularity of information, such as *where* these events occur in the recording, or even how many times they occur. Such labels are considerably less expensive to collect, and it has been shown in prior studies that such labels are sufficient to learn effective classifiers [20, 21].

However, classifiers that are trained with weakly-labeled data perform significantly more poorly than those that are trained using strongly (or precisely) labeled data. This is true despite recent advances, and even when the weak labels are accurate. Additional suboptimalities arise when the weak labels themselves are imprecise (i.e. of uncertain quality). This happens generally in scenarios where they are too easily, or automatically obtained, e.g. by harvesting labels from metadata or comments associated with videos uploaded to services on the internet. Weak labels from such sources are likely to be noisy in many ways, e.g. indicating the presence of a sound when there is none, or missing the occurrences of sounds in a recording.

In addressing these challenges, there is great potential for improvement in techniques for training audio classifiers using *imprecisely* (or *non*-strongly) labeled data. This is the focus of this thesis.

Our work focuses on devising techniques to improve audio classification using weak labels. Within this framework, we address a variety of problems such as devising basic formalisms for training sound event detectors from weak labels, determining the relative importance of class-positive and class-negative audio for the classifier, and dealing with labeling ambiguities that arise when labels are collected from noisy sources such as the internet, etc.

The research presented here can be viewed as part of a broader effort to minimize annotation costs while still achieving strong empirical results. As mentioned above, a strong advantage of weak labels is that they impose a much lower labeling burden. However, it is worth investigating how their quality can be improved with minimal effort. To this end, we also investigate approaches that can refine weak labels with additional information, such as counts and ontologies, with minimal additional computational effort. As an extreme case, we also investigate approaches where *no labels are available* for the data used for training. Finally, we aim to formulate a common formal framework that encompasses all of these solutions. Ultimately, the goal is to bridge the gap between the remarkable abilities of human auditory perception and the current state of computational audition.

In this section, we detail the structure of this thesis and highlight significant contributions made to its various components.

1.1 Organization of the Thesis

The chapters of this thesis are organized into seven sections:

- **Section 1 - Introduction:** This section provides a broad overview of the field, highlighting the importance of sound event detection, the challenges of relying on strong annotations, and the opportunities and obstacles presented by weak labels.
- **Section 2: NELS and DCASE** Discusses issues related to learning models from data at scale, using practical systems as experimental settings. It comprises the following chapters:
    1. *NELS :* Techniques developed to enable continuous learning of sound events. These are described in the context of a large-scale sound event classification system called the Never Ending Learner of Sound, or "NELS." The goal of NELS is to be able to assign audio annotations to audio recordings on the internet, continuously update itself from data, and infer sound classes from natural language.
    2. *DCASE :* Challenges unraveled and addressed in a large-scale competitive evaluation called Detection and Classification of Acoustic Scenes and Events, or "DCASE," on learning audio classifiers from weak labels and enhancing audio understanding systems.
- **Section 3:** Addresses the generic problem of learning with weak labels, and the specific problem of weak-label-based learning for sound event detection. It includes solutions for dealing with noise and uncertainty in weak labels, and extending weak labels with easily obtainable information. It comprises the following chapters:
    1. **Learning from weak labels:** Describes the formalisms we have developed for learning acoustic event detectors from weak labels, and presents empirical results showing that it is possible to achieve good acoustic event detection performance with models learned from weakly-labeled data.
    2. **Webly labeled data:** Describes our solution to learning from weak labels culled from sources such as the Internet. Such labels tend to be noisy and uncertain and pose many challenges during learning.
    3. **Semi-weak labels:** Describes our work on extending or augmenting weak labels with additional easily obtained information. We especially focus on the **counts** of events, i.e. the number of times an event has occurred in a recording. We refer to such labels as "semi-weak" labels and a solution for learning audio classifiers with semi-weakly labeled data.

- **Section 4:** Discusses ways of identifying and exploiting **structure** in the data. It comprises the following chapters:

    1. **The usefulness of samples:** It is known that not all training samples are equally important in learning a classifier. This is particularly true when training with weak labels. This chapter describes ways to identify the relative importance of samples, based on which we propose some effective sampling strategies.
    2. **Using Ontologies:** Ontologies provide useful information about how sound events are related to each other. In this chapter, we investigate ways of building ontological hierarchies of sounds using weakly labeled data and building classifiers based on those.

- **Section 5:** This section comprises a single chapter that describes our proposed unified formalism that incorporates solutions to the various problems we address in the context of training classifiers with imprecisely labeled data.

- **Section 6 - Conclusion and Future work:** This section contains a chapter that presents our conclusions – mainly insights drawn from our work so far. It also presents future directions.

- **Appendix:** Discusses the following topics 1) Unsupervised adaptation at test time - an approach critical for any learning task, 2) Learning without labels - An extreme case of weakly labeled data is unlabeled data and we present an approach to exploit the large corpora of this data with empirical evidence of improvements, 3) Strengthening of labels and open research problems, 4) Appendix for unified formalism with additional results.

Conclusion

In this thesis, a comprehensive exploration of the synergies between audio signal processing and machine learning has been undertaken, with a specific focus on the integration of advanced machine learning techniques, particularly deep learning, into the realm of audio analysis. The research presented here has traversed foundational concepts to sophisticated methodologies, culminating in a discussion of potential future directions that could shape the field.

The foundational aspects of audio signal processing and machine learning were first delineated, establishing the necessary context and framework for subsequent, more specialized discussions. This foundational knowledge is vital for understanding the

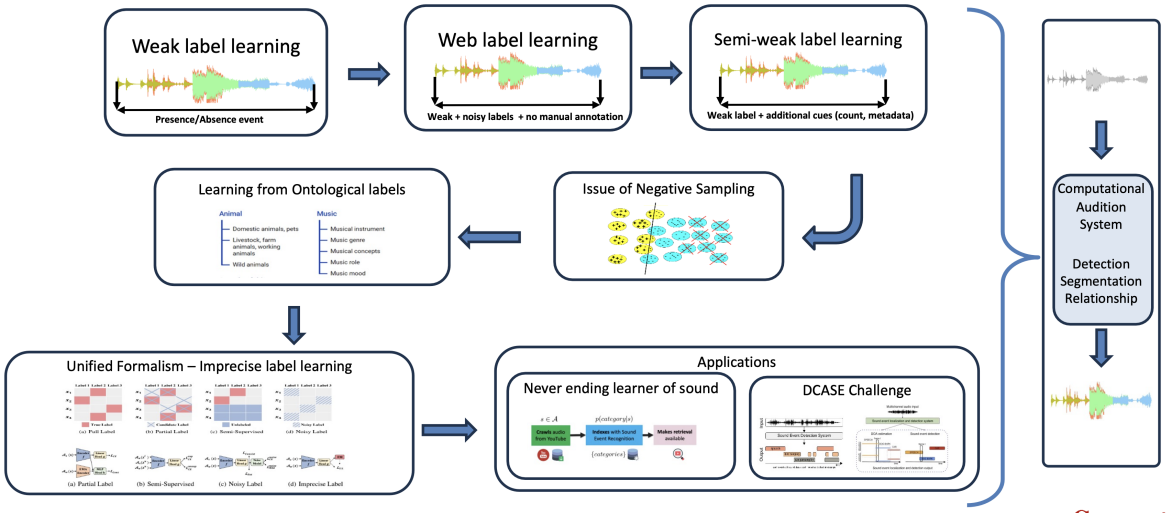


Figure 1.1: Overall Thesis Outline

complexities inherent in the analysis and interpretation of audio signals. The exploration of signal processing techniques and machine learning fundamentals lays the groundwork for the advanced concepts that follow.

In the realm of Sound Event Detection (SED), this thesis has provided an in-depth analysis of current methodologies, highlighting the challenges and innovations in this field. The discussion encompassed both traditional signal processing techniques and modern machine learning approaches, emphasizing the increasing importance of the latter in extracting nuanced patterns from complex audio data. The exploration of SED served as a precursor to the examination of weak label learning, a critical aspect of contemporary machine learning in audio signal processing.

The role of weak labels in machine learning was scrutinized, addressing both the opportunities and challenges they present. This discussion was extended to include specific algorithmic approaches for learning from weakly labeled data, with a particular emphasis on Convolutional Neural Network (CNN) architectures. The insights gained from these discussions underscore the importance of precise data annotation in the training of effective machine learning models and highlight ongoing efforts to optimize algorithms for weak label learning.

The thesis further delved into the practical applications and implications of these concepts through the analysis of two major projects in the field, NELS and DCASE. These case studies not only provided empirical evidence of the capabilities and limitations of existing techniques but also offered a perspective on the practical challenges faced in real-world implementations of audio signal processing systems.

Addressing the challenges of learning from noisy and web-sourced data, the thesis

explored contemporary strategies and methodologies, reflecting the increasing relevance of these data sources in the era of big data. This discussion highlighted the need for innovative approaches to handle the unique characteristics of such data.

Looking towards the future, the thesis proposed new directions for research and development in the field. The exploration of semi-weak label learning, ontology integration in audio analysis, and emerging trends in sound localization and unsupervised learning point toward a landscape ripe for innovation. These future directions not only address current limitations but also open new avenues for research, promising significant advancements in the efficiency and efficacy of audio signal processing systems.

In conclusion, this thesis contributes to the academic discourse in audio signal processing and machine learning by providing a nuanced understanding of the current state of the field and its potential trajectory. The research presented herein is characterized by its academic rigor, technical depth, and practical relevance, offering valuable insights and frameworks for both theoretical exploration and practical application. The findings and discussions pave the way for future innovations, encouraging continued research and development in this dynamic and evolving field. The hope is that this work will serve as a foundation for future studies and innovations, driving the field forward in the years to come. [1]

[1] Code for thesis available at https://github.com/cmu-mlsp/ankit-phd-thesis

Section II : NELS and DCASE

During the early stages of this dissertation, large-scale audio analysis systems were still relatively rare, so we set about creating two platforms to aid in our work:

1. NELS, a large-scale indexing platform that was intended to index audio on the web by content. Possibly the earliest of its type of platforms, NELS enabled a continuously expanding vocabulary of sound classes, as well as content-based retrieval of audio through natural-language text queries.

2. To deal with the paucity of data of the form we deal with, and to popularize the general area of research of dealing with such data, we designed and conducted DCASE challenges.

Both NELS and DCASE not only contributed significantly to the work in this thesis, but have also helped advance the field in general. Since its inception in 2016, NELS was reported in several research publications discussed in Section 2.7, and has won multiple awards. The DCASE challenges (and their offshoots) and the data remain popular. In the two chapters of this section, we describe NELS and our DCASE challenges.

2

Never Ending Learner of Sound

Videos and sounds available on the internet constitute the largest archive of sounds that we know of. They are continuously updated, and their volume on the internet increases by the minute, if not by the second. It is estimated that more than a million minutes of videos are being uploaded to the Internet each minute. This constitutes the vast majority of all consumer traffic [22]. The ability to recognize sounds in all these recordings is key to organizing, understanding, and exploiting rapidly growing audio and multimedia data.

However, most Internet recordings have unlabeled content, making it difficult to use them for automatic sound analysis, indexing, and retrieval. To enable these applications, it is important to develop methods that address the challenges involved in learning from such data, such as those that leverage the relation between sounds and language to assign labels, those that recognize and differentiate between diverse sound classes, and also methods of continuous, large-scale evaluation of the implemented techniques for sound analysis.

To work on these challenges and to consolidate the solutions on one platform, we developed a system that continuously learns from the data on the internet. It attempts to learn relations between sounds and language and improves sound recognition models over time.

We call this system the Never-Ending Learner of Sounds (NELS). It is available

online at **nels.cs.cmu.edu**.

2.1 Prior work on sound recognition and understanding

At the time of this work research on sound recognition, such as [23, 24, 25, 26], largely focused on curated data, with carefully collected and recorded audio. These generally used well-defined, task-oriented sets of sound classes. In addition, they include a limited set of classes and samples and have have rich descriptions of their content.

In contrast, the challenges of learning from weakly curated or uncurated web audio were, and indeed remain relatively under-explored. Similar challenges that use data collected in an unstructured manner, with an unbounded number of sound classes covering a wide range of topics, include an ever-growing number of classes and samples, and have insufficient, unavailable, or wrong descriptions, etc. are necessary to develop good solutions for learning from imprecisely labeled data. These requirements strongly motivate a system like NELS.

Never-ending learning architectures that learn many types of knowledge from years of diverse sources, using previously learned knowledge to improve subsequent learning with sufficient self-reflection to avoid learning stagnation, have been explored previously in the literature in other contexts, e.g. Never-Ending Language Learner [27] for text and Never-Ending Image Learner [28] for images.

In contrast, NELS represents the first such system for learning sounds. To enable a system such as NELS, we need (in the least) strategies that learn associations between sounds and language (metadata, ontologies, descriptive terms), continuously grow acoustic vocabularies, improve the robustness of sound recognizers, and evaluate the performance of sound recognizers in the absence of prior knowledge of the source or generation process.

The following sections address our work towards developing each of these strategies under NELS.

2.2 Learning associations between sounds and language

In the context of NELS, examples of associated knowledge are semantics related to objects, events, actions, places [29], cities [30, 31] or qualities [32, 33].

The following paragraphs describe the indexing and search (using indexed sounds) within the NELS framework. These use elements of associated knowledge such as those in the examples mentioned above.

The NELS framework includes data acquisition through a crawler that runs continuously (24/7), collecting audio and metadata from YouTube videos. From these, it

creates a content-based index based on a vocabulary of 605 sounds.

To initialize the process, NELS uses search queries corresponding to 605 sound event labels from four different datasets. The queries are used to collect the corresponding audio and metadata from YouTube videos. The video metadata is extracted using the Pafy API [34] and corresponds to 12 attributes, such as title, URL, and description. The crawled metadata are used to index the audio content. Although NELS can feed from different sound web archives, we selected YouTube as our first source due to its diversity of sounds and the available metadata associated with them.

The procedure for indexing further involves sound recognizers and their evaluations, combined with meta-data gathered from the automatic downloads. The sound recognizers are initially trained using data from various sources, including web audio. Their performance is evaluated using standard evaluation metrics such as the F1 score and precision for classifier detections, as well as evaluation of the segmented data was performed using human feedback. Combining crawled metadata, sound recognition predictions, and human feedback is done as follows:

We crawled over 2300 hours of audio corresponding to 4 million video segments of 2.3 seconds. The indexed audio content is available for search and retrieval using our engine on the user interface.

2.2.1 Relation between sounds and language

Language can describe audio content, be used to search for sounds, and help to define sound vocabularies [35]. However, the relationship between sounds and language is still an inchoate topic. To better understand the language usage for our indexing, we performed two studies.

First, sound recognition results [23, 36] evidenced how although two sound events: *quiet street* and *busy street* defined audio from streets, the qualifier implied differences in the acoustic content. These kinds of nuances can be described with Adjective-Noun Pairs (ANPs) and Verb-Noun Pairs (VNPS) [32]. Our team collected one thousand pair labels derived from different audio ontologies. The audio recordings containing the sound events and their pair labels were crawled from the collective archive *freesounds.org*. It was concluded that despite the subjectivity of the labels, there is a degree of consistency between sound events and both types of pairs.

Second, in [33], we wanted to identify text phrases that contain a notion of audibility and can be termed as a sound event. We noted that sound-descriptor phrases can often be disambiguated based on whether they can be prefixed by the words "sound of" without changing their meaning. Hence, by matching the combination "sound(s) of <Y>" where Y is any phrase of up to four words to identify candidate phrases, followed

by applying a rule-based classifier to eliminate noisy candidates, we obtained a list of over 100,000 sound labels. Further, we could also discover ontological relations by applying a classifier to features extracted from a dependency path between a manually listed set of acoustic scenes and the discovered sound labels. For example, forests may be associated with the sounds of *birds singing*, *breaking twigs*, *cooing*, and *falling water*.

2.3 Continuously growing acoustic vocabularies

An essential component of continuously growing sound vocabularies is to continuously acquire new data. We do this through continuous crawling based on newer queries generated by NELS by analyzing data obtained in response to the initial 605 classes.

In contrast to audio-only recordings in archives such as *freesounds.org*, collecting audio from videos poses several problems. YouTube contains a vast variety of videos, and a proper formulation of the search query is necessary to filter videos with higher chances of containing the desired sound event. Typing a query composed by a noun such as *air conditioner* will not necessarily fetch a video containing the corresponding sound event because the associated metadata often corresponds to the visual content. For example, it may contain visuals and descriptions of air conditioner inventory in an electronics shop.

As a solution, we modify the query to be a combination of keywords, for example, "air conditioner sound." Although the results were empirically improved in response to this, there is still no guarantee that the sound event will occur in the recording. On the other hand it may be present, but of very short duration, overlap with other sounds, or have a low relative volume.

We use the heuristic of discarding videos longer than ten minutes and shorter than two seconds because they are either likely to contain unrelated sounds or be too short to be useful.

2.4 Improving the robustness of sound recognizers

Extending the training dataset and improving the classifiers improves the robustness of sound recognizers. We initialize by using the data obtained from the initial 605 sound events (obtained from four annotated datasets) to train classifiers. We then use the trained classifiers to recognize newly crawled YouTube video segments that are unlabeled. These pseudo-labeled data are then used to improve the classifiers. The class predictions are also used to index the audio content.

Initially, we used the Convolutional Neural Networks (CNNs) classifier described in [37], trained on the following datasets. Using curated annotations in these datasets

for initialization helps us deal with mismatch conditions and improves robustness.

1. **ESC-50**: The ESC-50 database [37] has 50 classes from five broad categories: animals, natural soundscapes and water sounds, human non-speech sounds, interior/domestic sounds, and exterior sounds. The dataset consists of 2,000 audio segments with an average duration of 5 seconds each.

2. **US8k**: The US8K or UrbanSounds8K [26] database has 10 classes (such as gunshot, jackhammer, children playing). The dataset consists of 8,732 audio segments with an average duration of 3.5 seconds each.

3. **TUT16 (Task-3)**: The TUT 2016 (Task-3) [36] has 18 classes (such as car passing by, bird singing, door banging) from two major sound contexts, namely home context and residential area. The dataset comprises 954 audio segments with an average duration of 5 seconds each.

4. **AudioSet**: The AudioSet dataset [38] has 527 classes and 2.1 million audio segments with an average duration of 10 seconds each.

The recordings in the data sets above are segmented into 2.3 seconds and converted into 16-bit encoding, mono-channel, and 44.1 kHz sampling rate WAV files as in [37]. Then, we extract features comprising log Mel-spectrogram with 60 mel-bands with a window size of 1024 (23 ms) and a hop size is 512. These features are used to train multi-class classifiers using CNNs for each dataset.

Subsequently, audio from both the datasets and crawled video segments are preprocessed and classified. Ultimately, NELS is meant to be classifier agnostic and the bank of classifiers used will include multiple types of classifiers.

2.4.1 Continuous semi-supervised learning of sounds.

NELS should use existing curated sound datasets and non-curated web audio to improve its learning. Previously, semi-supervised self-training approaches have been used to improve sound event recognizers [39, 40]. In [40], we used an earlier version of NELS. Its classifiers were trained and tested using the US8K dataset consisting of about 8,000 labeled samples for 10 sounds. For re-training, we used 200,000 unlabeled YouTube video segments. Similar to the earlier work of [40], but with mismatched data, we achieved about 1.4% overall precision improvement. Regardless of the improvement, we reached a learning plateau. This could be due to mismatched conditions between training and self-training audio. The initial class bias is introduced by the hand-crafted dataset, and the use of ambiguous YouTube audio to self-train sound classes. Therefore,

additional investigation is required to glean insights from the continually expanding repository of web audio.

2.5 NELS Framework

In its current form, NELS is a framework that continuously (24/7) crawls audio and metadata from YouTube videos and creates a content-based index based on a vocabulary of 605 sounds. The sound recognizers were trained from a variety of sources, including web audio itself. NELS also evaluates the quality of the recognition through human feedback. The audio content is indexed combining the crawled metadata, sound recognition predictions and human feedback. The indexed audio content is available for search and retrieval using our engine on the website.

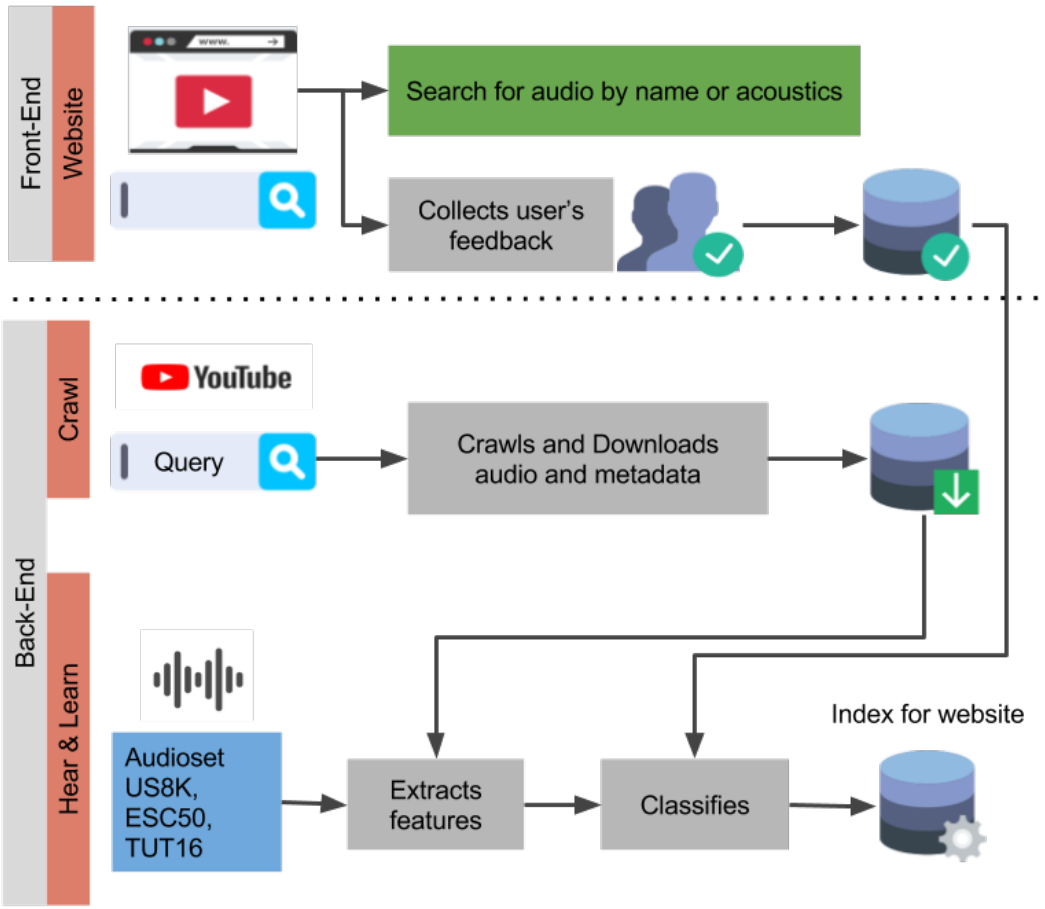


Figure 2.1: The framework serves as a continuous audio content indexer, a sound recognition evaluation, and a search engine for the indexed audio.

2.5.1 Crawl

In this module, a search query is used to crawl audio and metadata from YouTube videos. The query corresponds to 605 sound event labels from four different datasets.

The video metadata is extracted using the Pafy API[1] and corresponds to 12 attributes, such as title, URL, and description. The crawled metadata is used to index the audio content.

Although NELS can feed from different sound web archives, we selected YouTube as our first source due to its diversity of sounds and the available metadata associated with them. In contrast to audio-only recordings, collecting audio from videos poses several challenges. YouTube contains a massive amount of videos and a proper formulation of the search query is necessary to filter videos with higher chances of containing the desired sound event. Typing a query composed by a noun such as *air conditioner* will not necessarily fetch a video containing such sound event because the associated metadata often corresponds to the visual content; contrary to audio-only archives such as *freesounds.org*. Therefore, we modified the query to be a combination of keywords: "<sound event label> sound", for example, "air conditioner sound". Although the results empirically improved, the sound event was not always found to be occurring and even if it was, sometimes it was present within a short duration, overlapping with other sounds and with low volume. We discarded videos longer than ten minutes and shorter than two seconds because they were either likely to contain unrelated sounds or were too short to be processed.

2.5.2 Hear & Learn

In this module, we used 605 sound events from four annotated datasets to train classifiers and run them on the crawled YouTube video segments, which are unlabeled. The class predictions are also used to index the audio content.

The framework is being developed so that given a set of guidelines, new datasets could be added seamlessly. NELS should be able to take advantage of existing curated annotations, however dealing with mismatch conditions. The current four datasets are: ESC50, US8k, TUT16 (Task-3), and AudioSet. *ESC-50* [37] has 50 classes from five broad categories: animals, natural soundscapes and water sounds, human non-speech sounds, interior/domestic sounds, and exterior sounds. The dataset consists of 2,000 audio segments with an average duration of 5 seconds each. *The US8K or UrbanSounds8K* [26] has 10 classes like *gun shot, jackhammer, children playing*. The dataset consists of 8,732 audio segments with an average duration of 3.5 seconds each. *TUT 2016 (Task-3)* [36] has 18 classes like *car passing by, bird singing, door banging* from two major sound contexts namely home context and residential area. The dataset consists of 954 audio segments with an average duration of 5 seconds each. *AudioSet* [38] has 527 classes and 2.1 million audio segments with an average duration of 10 seconds

[1] https://pypi.python.org/pypi/pafy

each.

The audio from both the datasets and crawled video segments are preprocessed and classified. NELS is meant to be classifier agnostic. At the time of the original work, we followed a Convolutional Neural Networks (CNNs) classifier setup described in [37]. Recordings are segmented into 2.3 seconds and converted into 16-bit encoding, mono-channel, and 44.1 kHz sampling rate WAV files as in [37]. Then, we extracted features comprising log-scaled mel-spectrograms with 60 mel-bands with a window size of 1024 (23 ms) and a hop size is 512. Lastly, the features are used to train multi-class classifiers using CNNs for each of the datasets. The system has since been updated to use a more contemporary architecture [41].

2.6 Evaluation in the absence of prior knowledge

The evaluation of NELS is included in its user interface. This is shown in Fig. C.4.

Figure 2.2: NELS user interface Interface: The user interface takes a text query and recovers audio recordings that best match the text in the indexed database.

The user interface for NELS **nels.cs.cmu.edu** and currently serves two purposes: 1) to evaluate sound recognition using human feedback, which is also part of the audio indexing, and 2) to provide a search engine for indexed audio content.

In NELS, search is an integral part of the evaluation, and we describe this first.

2.6.1 Search

The user interface provides a search field to capture a term (text-query) for searching for sounds. The term is mapped to the closest of available sound classes. Our initial implementation performed the mapping using the tool *word2vec* [42] and a precomputed vocabulary of 400,000 words called *Glove* [43]. Word2Vec computes vector representations of vocabulary words and the text query. It then computes cosine similarity between the text query, the precomputed vocabulary, and our list of sounds. More recently we have switched to more contemporary encoders such as [41]. Since our list of sound classes does not necessarily match all the words of the precomputed vocabulary, we only consider words within a similarity threshold of 0.15 (otherwise no results may be retrieved). This is used to provide an additional functionality to the NELS interface – that of matching sounds. This allows the user to paste a YouTube video link on a second text field, inresponse to which NELS yields the dominant sound in the corresponding video.

As part of the human evaluation strategy, each displayed video segment resulting from the text-based search can be evaluated by the user with a two -button option: *Correct* or *Incorrect* (i.e., whether the human claims that the system's predicted class was present within the given segment or not). Figure 2.3 shows some examples.

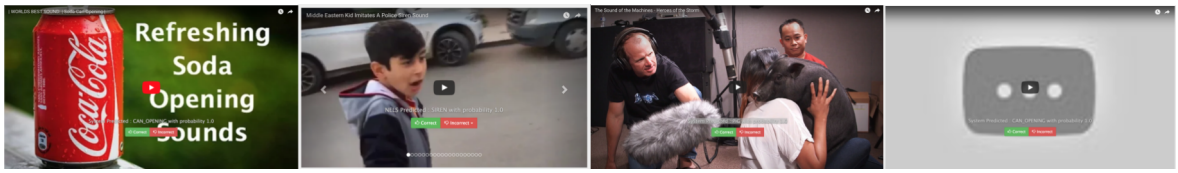


Figure 2.3: Examples of indexed video segments using NELS. One shows an example where the title and images are related for a *can-opening* sound. Two shows an example where a *siren wailing* sound and title are related, but not the visual sound source, which is a child rather than an electronic device. Three shows an example of *pig-oink* sound, which matches the visuals but not with the title and text metadata. Four shows the thumbnail of a video that was indexed but eventually deleted by the user.

2.6.2 Evaluation of the learning quality

NELS indexes audio content 24/7, but these segments are unlabeled or have weak or wrong labels. Therefore, finding methods to evaluate quality automatically at a large scale is essential. A solution is to include human intervention [44]. Hence, our user interface allows human feedback collection to assess the correctness of sound event

indexing. Nevertheless, human feedback may be slow or costly. Hence, combining it with other methods that estimate performance at a large scale is important.

In [45], we used an earlier version of NELS with a recognizer trained on 78 sound events from three different datasets. After, we crawled audio from YouTube videos using the sound event labels from the datasets as the search query. The query was a combination of keywords: "<*sound event label*> sound", for example, "air conditioner sound". Then, we evaluated the highest-40 recognized segments per sound class based on both types of reference (ground truth), human feedback, and search query. The search query is a summary of the video's metadata that describes the whole video, but it was interesting to know to what extent it holds at the video's segment level. The results showed how the performance trends for using both types of references are similar and relatively close with less than an absolute 10% difference in precision. This trend suggests that the query could be used as a lower bound of human inspection. In other words, it could serve as a preliminary reference for evaluating sound recognition. Further exploration of this and other associated metadata and multimedia cues could be used as alternative measurements.

2.7 Discussion

In summary, acoustic intelligence in machines requires at least three essential components (a) extracting sound-related knowledge from text (b) recognizing and detecting a large number of sound events, and (c) a learning system that links these two systems. A system like NELS is expected to do all of these same 24/7, continuously trying to discover new sound concepts, attempting to build a computational model for it by mining audio on the web, and then finally trying to infer some higher-level semantic knowledge and commonsense relations about sounds. We have addressed the individual components in this dissertation, and the next steps would be to tie the components together. Moreover, we will also have to incorporate the never-ending part where the processes are continuously running and updating the knowledge base.

The system consists primarily of a front end (nels.cs.cmu.edu), which shows information about the current state of the system. It shows the vocabulary of sound events that the system can currently detect and examples of YouTube videos where the system has detected the sound events. We also provide users with several ways to query the system. Users can search for a sound through a text query, and the system returns a set of YouTube videos where it has detected the presence of the queried sound event. Users can also directly upload an audio recording and the system can annotate the recording with sound events. The same can also be done for any YouTube video as well by providing the URL link of that video to the system. The system's back end primarily

consists of crawling YouTube for sounds and then learning from the crawled videos. The recognition system has been seeded by detectors built using various datasets such as Audioset, Urbansounds, and ESC-50. In the next phase of NELS, the goal would be to add semi-supervised approaches, allowing the system to learn from unlabeled audio data on the Web. Moreover, the system also needs to continuously improve itself as far as detecting sound events is concerned. To this end, human feedback might be important to ascertain that the system is actually improving its performance [45]. The next phase would also involve linking the natural language understanding from text with the detection process in audio/multimedia recordings. The two processes can help each other and be useful in establishing commonsense relationships for sounds.

2.8 Key Discoveries during building NELS

The territory of web audio during the building phase of NELS was relatively unexplored and had several challenges and opportunities. For instance, traditionally, sound recognition research has focused on curated datasets with meticulously recorded and labeled audio, carefully chosen sound classes, and limited sample sizes [23, 24, 25, 26]. However, the vast and ever-growing world of web audio presents a unique set of challenges. Unlike the structured datasets of the past, web audio is often:

- Unlabeled: Lack of consistent labels makes it difficult to train supervised learning algorithms.
- Unstructured: Audio content varies greatly in quality, duration, and complexity, with overlapping sounds and background noise.
- Unbounded: The number of sound classes is virtually infinite and new sounds are constantly emerging.

These challenges make it difficult to apply traditional sound recognition techniques to web audio. However, the sheer volume and diversity of web audio also present a unique opportunity. By tapping into this rich source of data, we can potentially build sound recognition systems with unprecedented capabilities, able to recognize and understand a vast and ever-evolving vocabulary of sounds.

3

Techniques for Detection of Acoustic Scenes and Events

Over the past decade, work toward developing better techniques for acoustic scene and event detection has been done under a community-wide computer audition challenge called the Detection and Classification of Acoustic Scenes and Events (DCASE) challenge. As part of this thesis, we participated in the definition and setup of this task. We also developed approaches in many subtasks.

DCASE is a research competition that is focused on developing algorithms for automatic audio event and scene recognition. The challenge is organized annually and typically related to a wide number of tasks for computational audition such as classifying the different types of sound events or identifying the location or context for the audio recording. The challenge is open to researchers from all over the world and is an important event in the field of audio event and scene recognition.

Since the outset of this thesis, we have been involved with the DCASE challenges, both as *participants* and as *organizers*. These contributions have shaped the course of this dissertation. The following is our contribution to the DCASE challenge over the years. We participated in the DCASE 2016 challenge for Acoustic Scene Classification and Sound Event Detection from Real-Life Recording, as detailed in the paper [46], and organized DCASE 2017 Task 4, titled "Large-scale weakly supervised sound event

detection for smart cars", focused on detecting sound events using weakly labeled training data [25]. We followed this up by co-organizing Task 4 in DCASE 2018 for large-scale weakly supervised sound event detection, where the task aimed to develop systems capable of detecting sound events using weakly labeled data, given only labels at the clip level without time boundaries and unlabeled data [47]. In the following year at DCASE 2019, Task 4 was focused on sound event detection (SED) in domestic environments using weakly labeled data and soundscape synthesis where the primary goal was to advance SED with limited annotations, utilize synthetic data for model training and improve generalization in domestic settings using weak labels [48]. In the next section, we briefly describe the DCASE task challenges over the years and then dive into the individual contributions as described above.

3.1 Tasks in DCASE Challenge

There are a total of six different tasks carried out over the years in DCASE challenges from 2016 to 2022. These tasks are as follows:-

- Task 1: Acoustic Scene Classification
- Task 2: Unsupervised Anomalous Sound Detection for Machine Condition Monitoring Applying Domain Generalization Techniques
- Task 3: Sound Event Localization and Detection Evaluated in Real Spatial Sound Scenes
- Task 4: Sound Event Detection in Domestic Environments
- Task 5: Few-shot Bioacoustic Event Detection
- Task 6: Automated Audio Captioning and Language-Based Audio Retrieval

3.1.1 Task 1: Acoustic Scene Classification

The objective of acoustic scene classification is to classify a test audio recording into one of the predefined classes, provided that it characterizes the environment in which it was recorded. Examples of acoustic scenes might include a busy street, a quiet office, or a crowded train station. [1]

The acoustic scene classification task comprises two different subtasks:

Subtask A: Acoustic Scene Classification with Multiple Devices: This subtask requires the classification of data from real and simulated devices. The goal is to develop

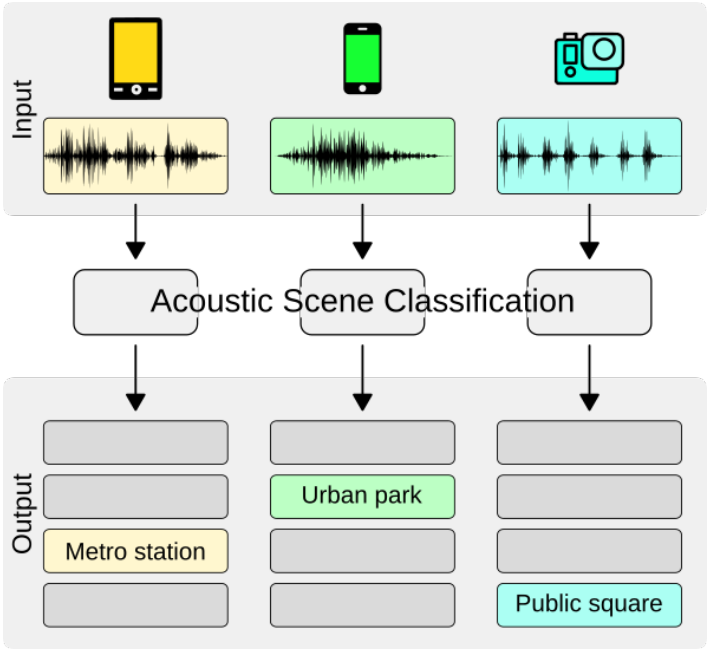


Figure 3.1: Overview of the Acoustic Scene Classification System [1]

techniques that generalize well across a large number of different devices. Data for this task comprise real and simulated audio recordings for mobile devices.

Subtask B: Low-Complexity Acoustic Scene Classification: This subtask requires the classification of data from a single device. The goal is to develop low-complexity solutions for acoustic scene classification. Constraints are imposed on the model size (number of parameters) complexity (multiply-accumulate operations count), and memory allowance. In addition to low complexity, the aim is to generalize across several different devices. For this purpose, the task uses audio data recorded or simulated on a variety of devices.

The following figure shows a brief overview of the progress made in the field from the year 2017 to 2022 in the DCASE challenge, where gradually over time the challenge data have increased from smaller-scale datasets to larger-scale datasets, with the additional task of low complexity acoustic scene classification being introduced in 2021 and 2022.

The key findings from this task during these years were that the best performing systems used convolutional neural networks with attention mechanisms and data augmentation to improve the classification score. Generalization across devices was improved using device-invariant features and domain adaptation methods. For the low-complexity acoustic scene classification, the participants achieved competitive results using efficient architectures and quantization. For the low-complexity solution with the post-training quantization of parameters, it was observed that the technique reduces the number of bits used to represent each weight or activation in the neural network, which

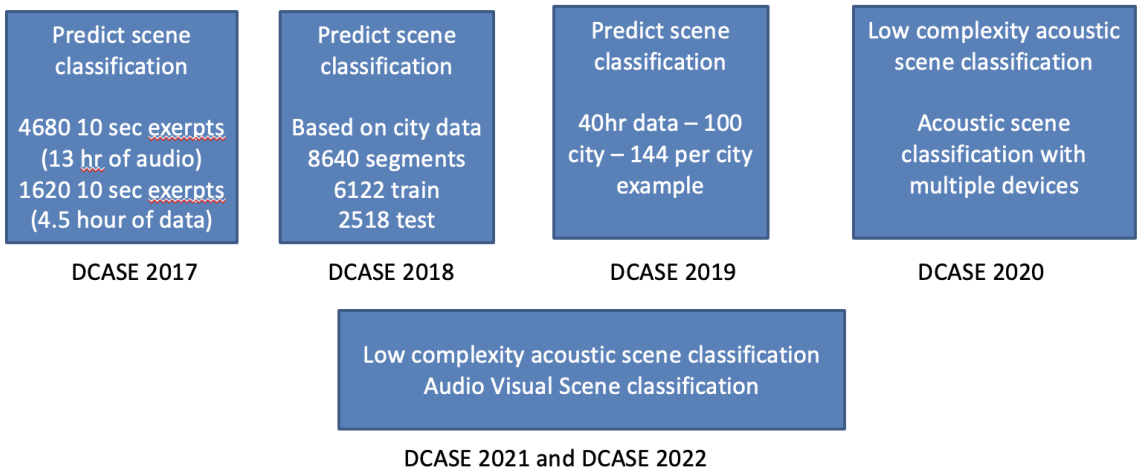


Figure 3.2: Progress in Acoustic Scene Classification at DCASE Challenge between 2017 to 2022

led to smaller models and faster inference. Separable depthwise convolutions, where one factorizes the standard convolution into depth-wise and point-wise convolution, led to an additional reduction of the parameters, and operations saw additional success on this task of acoustic scene classification.

3.1.2 Task 2: Unsupervised Anomalous Sound Detection for Machine Condition Monitoring Applying Domain Generalization Techniques

Anomalous sound detection (ASD) is the task of identifying whether the sound emitted from a source is normal or anomalous. This is useful in many situations, e.g. automatic detection of mechanical failure or detection of mechanical anomalies for monitoring the condition of machines. Figure 3.3 shows an overview of the anomaly detection system for this task.

The evaluation metric for DCASE task 2 is based on the area under the curve (AUC) and partial AUC (pAUC) scores. AUC score measures how well a system can distinguish between normal and anomalous sounds across all possible thresholds. pAUC score measures how well a system can distinguish between normal and anomalous sounds at low false alarm rates. The final score is calculated by averaging the AUC and pAUC scores on all types of machines and test sections.

For this task, around 1000 samples were provided as training data for each machine type/ID (i.e., each class). These data were expected to be insufficient to train a large-scale DNN.

The participant developed DNN-based strategies for outlier detection. These strate-

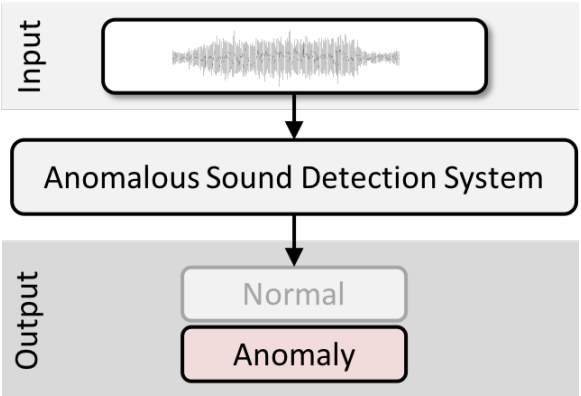


Figure 3.3: Overview of the Anomaly detection system

gies are similar to using unsupervised approaches to learn accurate decision boundaries between normal and anomaly classes. They designate the data for other classes as anomalies for the target class for better classification. This achieves high AUC scores for the different types of machines.

These approaches have the drawback that it is difficult to obtain good decision boundaries when the samples of anomalies are similar to the target sample. This leads to frequent false positives during classification. Also, there is a high variety of anomalies, and these models are not very robust. Solving this is still an open problem for the anomaly detection task.

3.1.3 Task 3: Sound Event Localization and Detection Evaluated in Real Spatial Sound Scenes

The goal of the sound event localization and detection task is to detect occurrences of sound events belonging to specific target classes, track their temporal activity, and estimate their directions of arrival or positions during it.

Given multichannel audio input, a sound event detection and localization (SELD) system outputs a temporal activation track for each of the target sound classes, along with one or more corresponding spatial trajectories when the track indicates activity. This results in a spatiotemporal characterization of the acoustic scene that can be used in a wide range of machine cognition tasks, such as inference on the type of environment, self-localization, navigation without visual input or with occluded targets, tracking of specific types of sound sources, smart-home applications, scene visualization systems, and acoustic monitoring, among others.

The best-performing systems for this task used convolutional recurrent neural networks (CRNNs) with various input features such as log Mel spectrogram, intensity vector, and generalized cross-correlation (GCC). The evaluation metrics for this task

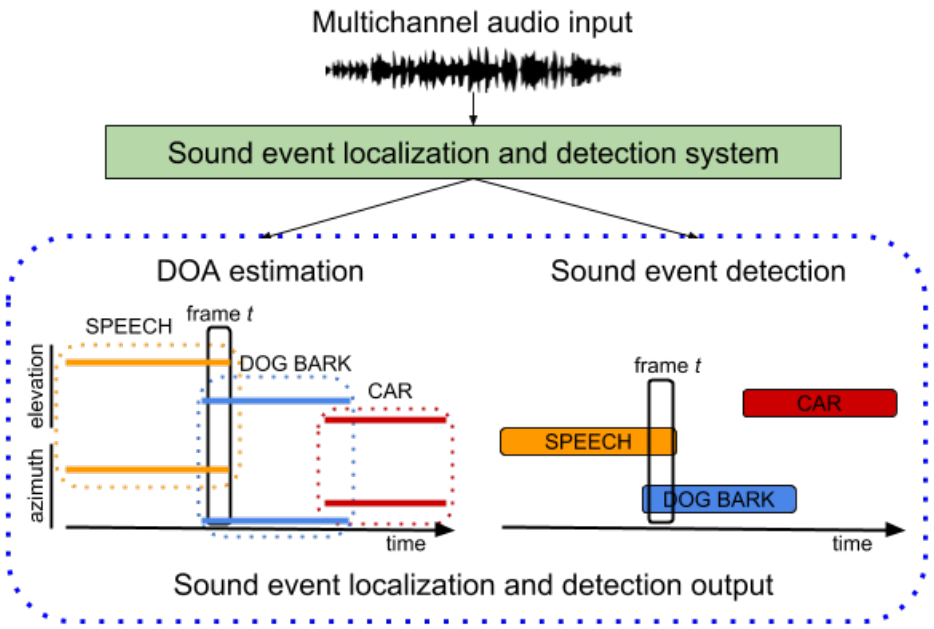


Figure 3.4: Overview of sound event localization and Detection system

were event-based F-score, polyphonic sound event detection score (PSDS), location-aware detection score (LDS), and direction of arrival error (DOA).

3.1.4 Task 4: Sound Event Detection in Domestic Environments

The goal of the task is to develop techniques for detecting and localizing sound events in audio recorded in domestic environments. The task uses heterogeneous data comprising weakly labeled or unlabeled real data and simulated data that is strongly labeled (with time stamps). The data provide information not only about the event class but also about event time localizations in multi-event recordings. Figure 3.5 shows an overview of the sound event detection system.

The participants were allowed the use of external data and pre-trained models in this task. The main scientific question addressed here pertains to what strategies work well when training a sound event detection system with a heterogeneous dataset, including the following: 1) a large amount of unbalanced and unlabeled training data, 2) a small weakly annotated set, and 3) a synthetic data set containing isolated sound events and backgrounds. Additionally, the task investigates the impact of using embeddings extracted from pre-trained models, and also whether there is any potential advantage in using external data.

The evaluation metric for this tasks was sound event detection (SED) and sound event separation (SES) performance. SED performance was measured by the segment-

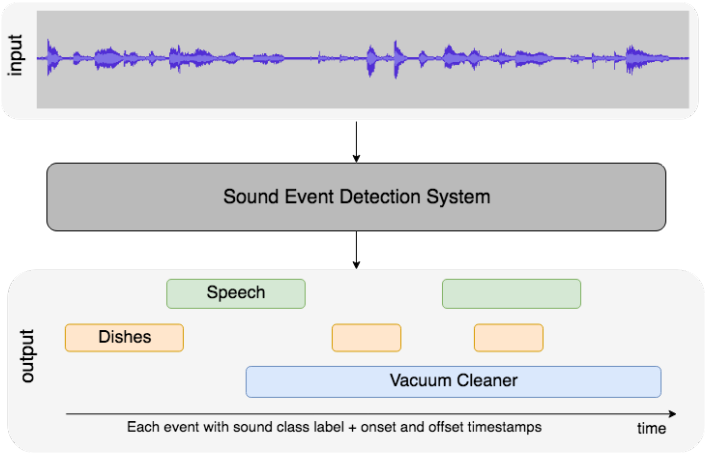


Figure 3.5: Overview of Sound event detection system

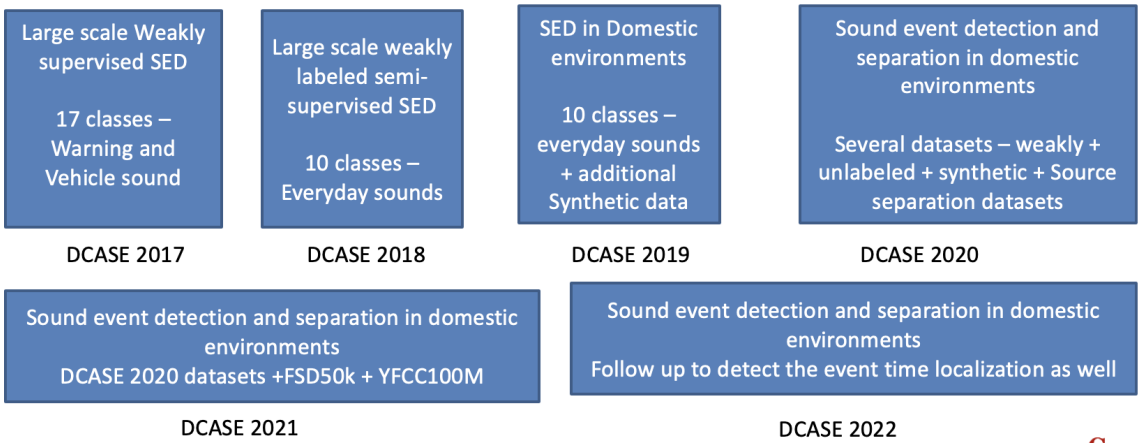


Figure 3.6: Progress in Weak Labels using DCASE Challenge

based F-1 score with a segment length of 1 second. SES performance was measured by improving the scale-invariant signal-to-distortion ratio (SI-SDRi) between the estimated source signals and the reference signals. The participants were evaluated on the final score, which was calculated as the average of the SED F1 score and the SES SI-SDRi score across all the test recordings.

$$\text{SED F1-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3.1}$$

$$\text{SES SI-SDRi} = \frac{1}{N}\sum_{n=1}^{N}(\text{SI-SDR}(s_n, \hat{s}_n) - \text{SI-SDR}(s_n, x_n)) \tag{3.2}$$

$$\text{Final score} = \frac{\text{SED F1-score} + \text{SES SI-SDRi}}{2} \tag{3.3}$$

where s_n is the reference source signal, $\hat{s}_n$ is the estimated source signal, x_n is the mixture signal, and N is the number of test clips.

The task shows that state-of-the-art systems are robust to noise and signal degradation, but still struggle with overlapping sound events.

3.1.5 Task 5: Few-shot Bioacoustic Event Detection

This task focuses on detecting sound events in a few-shot learning environment for animal (mammal and bird) vocalizations. The goal is to detect a novel sound class, given only a few examples, and the data comprise just five exemplar vocalizations (shots) of mammals or birds in field recordings.

[49] provide an overview of the task objectives, dataset, and baselines, as well as a comparison of 20 submitted systems based on different metrics. The best solutions included using an ensemble of prototypical neural networks with adaptive embedding functions to improve the F-score from 41.3% to 60.0% on the validation dataset [50], and using segment-level metric learning with negative data and transductive inference to achieve an F-score of 63.9% on the validation dataset [51].

The DCASE 2022 Challenge's Task 5 addressed the challenge of detecting novel animal vocalizations—specifically from mammals and birds—in field recordings, given only five exemplar sounds per class. This few-shot learning scenario is particularly relevant in bioacoustics, where annotated data is often scarce.

Nolasco et al. (2022) provided a comprehensive overview of this task, detailing its objectives, the dataset utilized, baseline performances, and a comparative analysis of 20 submitted systems. The top-performing approaches employed advanced few-shot learning techniques:

Prototypical Neural Networks with Adaptive Embedding Functions: Martinsson et al. (2022) implemented an ensemble of prototypical networks, each with adaptive embedding functions tailored to the specific characteristics of the target sound events. This method improved the F-score from 41.3% to 60.0% on the validation dataset.

Segment-Level Metric Learning with Negative Data and Transductive Inference: Liu et al. (2022) introduced a segment-level metric learning framework that incorporated both positive and negative examples during training. Additionally, they applied transductive inference to better adapt to the evaluation set, achieving an F-score of 63.9ARXIV

These results underscore the potential of few-shot learning methodologies in bioacoustic event detection, especially when dealing with limited annotated data. The significant improvements over baseline models highlight the effectiveness of these advanced techniques in capturing the nuances of animal vocalizations in complex acoustic

environments.

3.1.6 Task 6: Automated Audio Captioning and Language-Based Audio Retrieval

3.1.6.1 Automated Audio Captioning

Automated Audio Captioning (AAC) is the task of general audio content description using free text. It is an inter-modal translation task (not speech-to-text), where a system accepts as an input an audio signal and outputs the textual description (i.e. the caption) of that signal. AAC methods can model concepts (e.g. "muffled sound"), physical properties of objects and environment (e.g. "the sound of a big car", "people talking in a small and empty room"), and high-level knowledge ("a clock rings three times"). This modeling can be used in various applications, ranging from automatic content description to intelligent and content-oriented machine-to-machine interaction.

3.1.6.2 Language Based Audio Retrieval

This subtask is concerned with retrieving audio signals using their textual descriptions of the sound content (that is, audio captions). Human-written audio captions will be used as text queries. For each text query, the goal of this task is to retrieve 10 audio files from a given dataset and sort them based on their match with the query. Through this subtask, we aim to inspire further research into language-based audio retrieval with unconstrained textual descriptions.

3.2 Contribution to DCASE Challenge: Task 1 – Acoustic Scene Classification

Our approach was to treat Task 1 as a multi-class classification problem. For this, an effective method for characterizing acoustic scenes or events in audio segments is needed. One approach is to use bag-of-audio-words-based feature representations [52], which are usually built over low-level features such as MFCCs. Acoustic scenes, however, are more complex mixtures of different audio events and a more robust representation is required. To this end, we used Gaussian Mixture Models (GMMs) for feature representation. We describe this in detail below.

On the classification front, we used support vector machines (SVMs) as our primary classifier and in combination with other classifiers.

3.2.1 Feature Representations

Let D-dimensional MFCCs vectors for a recording be represented as $\vec{x}_t$, where $t = 1$ to T, T is the total number of MFCCs vectors for the recording. The major idea behind both

high-level feature representations is to capture the distribution of MFCC vectors of a recording. We will refer to these features as $\vec{\alpha}$ and $\vec{\beta}$ features and the sub-types will be represented using appropriate subscripts and superscripts.

The first step in obtaining high-level fixed dimensional feature representation for audio segments is to train a GMM on MFCC vectors of the training data. Let us represent this GMM by $\mathcal{G} = \{w_k, N(\vec{\mu}_k, \Sigma_k), k = 1\ to\ M\}$, where w_k, $\vec{\mu}_k$ and Σ_k are the mixture weight, mean and covariance parameters of the k^{th} Gaussian in $\mathcal{G}$. We will assume diagonal covariance matrices for all Gaussians and $\vec{\sigma}_k$ will represent the diagonal vector of Σ_k. Given the MFCCs vectors $\vec{x}_t$ of a recording, we computed the probabilistic assignment of $\vec{x}_t$ to the k^{th} Gaussian. These soft assignments are added over all t to obtain the total mass of MFCCs vectors belonging to the k^{th} Gaussian (Eq 3.4). Normalization by T is used to remove the effect of the duration of recordings.

$$Pr(k|\vec{x}_t) = \frac{w_k N(\vec{x}_t;\vec{\mu}_k,\Sigma_k)}{\sum_{j=1}^{M} w_j N(\vec{x}_t;\vec{\mu}_k,\Sigma_k)}, P(k) = \frac{1}{T}\sum_{i=1}^{T} Pr(k|\vec{x}_t) \tag{3.4}$$

The soft count histogram features referred to as $\vec{\alpha}$ is, $\vec{\alpha}^M = [P(1), ..P(k)..P(M)]^T$. $\vec{\alpha}^M$ is an M-dimensional feature representation for a given recording. It captures how the MFCC vectors of a recording are distributed across the Guassians in $\mathcal{G}$. $\vec{\alpha}^M$ is normalized to sum to 1 before using it for classifier training. The next feature ($\vec{\beta}$), also based on the GMM $\mathcal{G}$, tries to capture the actual distribution of the MFCC vectors of a recording. This is done by adapting the parameters of $\mathcal{G}$ to the MFCC vectors of the recording. We employ maximum *a posteriori* (MAP) estimation for the adaptation [53]. Parameter adaptation for k^{th} Gaussian follows the following steps. First, we compute,

$$n_k = \sum_{t=1}^{T} Pr(k|\vec{x}_t),\ \ E_k(\vec{x}) = \frac{1}{n_k}\sum_{t=1}^{T} Pr(k|\vec{x}_t)\vec{x}_t,\ \ E_k(\vec{x}^2) = \frac{1}{n_k}\sum_{t=1}^{T} Pr(k|\vec{x}_t)\vec{x}_t^2 \tag{3.5}$$

Finally, the updated mean and variances are obtained as

$$\hat{\vec{\mu}}_k = \frac{n_k}{n_k + r}E_k(\vec{x}) + \frac{r}{n_k + r}\vec{\mu}_k \tag{3.6}$$

$$\hat{\vec{\sigma}}_k = \frac{n_k}{n_k + r}E_k(\vec{x}^2) + \frac{r}{n_k + r}(\vec{\sigma}_k^2 + \vec{\mu}_k^2) - \hat{\vec{\mu}}_k^2 \tag{3.7}$$

The relevance factor r controls the effect of the original parameters on the new estimates. We obtain 2 different feature representation using the adapted means ($\hat{\vec{\mu}}_k$) and variances ($\hat{\vec{\sigma}}_k$). The first one denoted by $\vec{\beta}^M$ is an $M \times D$ dimensional feature obtained by concatenating the adapted means $\hat{\vec{\mu}}_k$ for all k, that is $\vec{\beta}^M = [\hat{\vec{\mu}}_1^T, ...\hat{\vec{\mu}}_K^T]^T$. In the second $\vec{\beta}$ features adapted $\hat{\vec{\sigma}}_k$ are concatenated along with $\hat{\vec{\mu}}_k$ to obtain a $2 \times M \times D$ dimensional

features. This form of $\vec{\beta}$ features are denoted by $\vec{\beta}_{\sigma}^{M}$.

3.2.2 Classification

Once the feature representation for audio segments has been obtained, Task 1 essentially becomes a multi-class classification problem. Our primary classifiers are SVMs where we explore a variety of kernels. For the $\vec{\beta}$ features, we use the Linear Kernel (LK) and RBF Kernel (RK). Finally, we have a classifier fusion step in which we combine the output of the different classifiers. We combined multiple classifiers by taking a prediction vote from each classifier, and the final predicted class is the one that gets the maximum vote. We call it *Fused Classifier* and observe that the fused classifier can provide significant improvement for several acoustic scenes.

3.2.3 Results

Our experimental setup with the folds structure is the same as the one provided by DCASE. We extracted 20-dimensional MFCC features using $30ms$ window and 50% overlap. MFCCs are augmented with their delta and acceleration features. For our final feature representation, we experimented with 4 different values of GMM component size M, $64, 128, 256$ and 512. The relevance factor r for $\vec{\beta}$ is set to 20. Table 3.1 shows the overall accuracy results for different cases. The accuracy for the MFCC-GMM *baseline* method provided in the challenge is 72.6%.

We can observe from Table 3.1 that $\vec{\alpha}$ features in general do not perform better than the baseline method for any SVM kernel. However, $\vec{\beta}$ features clearly outperformed the baseline method. In the best case, with $M = 128$ and $\vec{\beta}_{\sigma}^{M}$ our method outperformed the baseline by an absolute 5%.

Table 3.2 shows results for the fused classifiers. For the fusion step, we did not consider classifiers built over $\vec{\alpha}$ since these classifiers are inferior compared to those using $\vec{\beta}$ features. We can observe that our proposed method beats the baseline method by an absolute **6.3**%. Moreover, for scenes such as *Park, Train, Library* where the baseline method gives very poor results, we improved the accuracy by an absolute $\mathbf{16 - 30}$%. We also obtained superior overall accuracy on all folds which suggests that our proposed method is fairly robust. This is further supported by the fact that on the DCASE evaluation set, we achieved an overall accuracy of **85.9**%.

3.3 Contribution to DCASE Challenge: Task 3 – Sound Event Detection in Real-Life Recordings

In earlier work by Elizalde et. al [54, 55], the detection of sound events in scenes and long recordings has been treated as a multi-class classification problem. The classifier is trained with the event sound segments. During testing, the classifier outputs segment/frame-level predictions for all the classes.

Table 3.1: Task 1 Accuracy for different cases (Single Classifier)

	$\vec{\alpha}^M$		$\vec{\beta}^M$		$\vec{\beta}_\sigma^M$	
M	LK	RK	LK	RK	LK	RK
64	62.8	60.6	76.8	76.6	75.5	**76.7**
128	63.6	62.3	76.5	75.3	77.5	**77.5**
256	63.9	63.9	76.5	71.9	**76.6**	75.9
512	65.0	62.9	**76.4**	72.2	76.2	75.9

Table 3.2: Overall Task 1 Accuracy (Fused Classifier)

	Baseline					**Proposed**				
Scene	Fold 1	Fold 2	Fold 3	Fold 4	**Avg.**	Fold 1	Fold 2	Fold 3	Fold 4	**Avg.**
Beach	84.2	66.7	78.9	47.4	69.3	100	71.4	89.5	52.6	78.4
Bus	68.4	65.0	100	85.0	79.6	68.4	50.0	100	95.0	78.4
Cafe/Restaurant	66.7	94.7	71.4	100	83.2	88.9	63.2	76.2	95.0	80.8
Car	70.0	89.5	89.5	100	87.3	80.0	100	100	100	95.0
City Center	83.3	73.7	89.5	95.5	85.5	88.9	84.2	100	95.5	92.1
Forest Path	57.1	100	66.7	100	81.0	81.0	100	100	100	95.2
Grocery Store	52.6	81.0	89.5	36.8	65	89.5	81.0	94.7	84.2	87.3
Home	100	55.6	95.0	77.8	82.1	100	61.1	80.0	44.4	71.4
Library	47.6	38.9	15.0	100	50.4	47.6	33.3	85.0	100	66.5
Metro Station	84.2	94.4	100	100	94.7	94.7	94.4	100	100	97.3
Office	100	100	94.4	100	98.6	78.9	100	72.2	83.3	83.6
Park	10.0	5.6	0	40.0	13.9	65.0	33.3	50.0	30.0	44.6
Residential	78.9	47.6	100	84.2	77.7	84.2	42.9	94.7	57.9	69.9
Train	16.7	31.6	30.4	61.1	34.9	50.0	63.2	34.8	88.9	59.2
Tram	88.9	88.9	63.6	100	85.3	83.3	88.9	63.6	100	84.0
Overall	**67.2**	**68.9**	**72.3**	**81.9**	**72.6**	**80.0**	**71.1**	**82.7**	**81.8**	**78.9**

3.3.1 Features and Classifiers Optimization

We used a similar approach. As features, we first used conventional MFCCs with 13 coefficients, plus delta and double delta vectors computed from them, for a total of 39 dimensions.

For the classifiers, we considered Tpot [56], built on top of Scikit [57], which is a Python tool that automatically creates and optimizes machine learning pipelines. This toolbox (version 4) considers 12 classifiers such as Random Forest, SVMs, K-Neighbors, and Logistic Regression. The main Tpot parameter is "number of generations", which corresponds to the number of iterations run to tune the classifier; we set it to 15. Interestingly, Random Forest and Logistic Regression outperformed other models like SVMs.

For our experiments, we extracted the 18 sound events from the two scenes using the annotations, and then we extracted the features from these isolated sounds. For each experiment, the sound events' feature files were fed to Tpot in a randomly selected ratio of 75% training and 25% testing, with no overlap between train and test sets. We

Feature Type	**Accuracy**%
Home	56.4
Home + G	55.2
Home + G + P	55.7
Residential	53.3
Residential + G	57.8
Residential + G + P	56.7

Table 3.3: Sound-event classification accuracy using the DCASE set up with four-folds partitions. The inclusion of the [G]eneric class improved performance for both scenes, whereas the inclusion of the [P]erturbed audio improved only the Home performance.

kept the same partitions across our experiments for consistency. The performance was measured in terms of accuracy, and the performance of MFCCs was 67.7%.

3.3.2 Inclusion of Generic Sound Event Class

In the annotated data, not every segment of audio corresponds to a labeled sound. To handle unlabeled or out-of-vocabulary segments, we proposed a generic sound event class.

In the first experiment, we analyzed the impact of the generic class together with the 18 sounds in the multi-class classification setup described in Section 3.3.1 using MFCCs. The data for the generic class was chosen to be the unlabeled audio between the labeled segments.

There were two event classes (or scenes) in this task. The average number of sound events per scene was 60. We randomly selected 60 audio files from each scene for analysis.

The normalized confusion matrices (CMs) in Figures 3.7 and 3.8 show the performance. The performance with and without the generic class is 60.94% and 67.7% respectively. However, although the generic class shared the background acoustics with the other sound classes, it was not significantly confusible with them.

For the second set of experiments, the DCASE setup comprised separate audio scenes, each with and without the generic class. Additionally, in each case where the generic class was used, the generic class had events that were specific to the scene. A four-fold cross-validation experiment was conducted on the given data.

The results are shown in Table 3.3. We see that performance improves with the inclusion of the generic class, which reduces class ambiguity. In addition, the CMs (which are not included due to space limitations), had cleaner diagonals. The use of fewer sound classes reduces class ambiguity, and scene-specific optimization of the classifier using Tpot also helps.

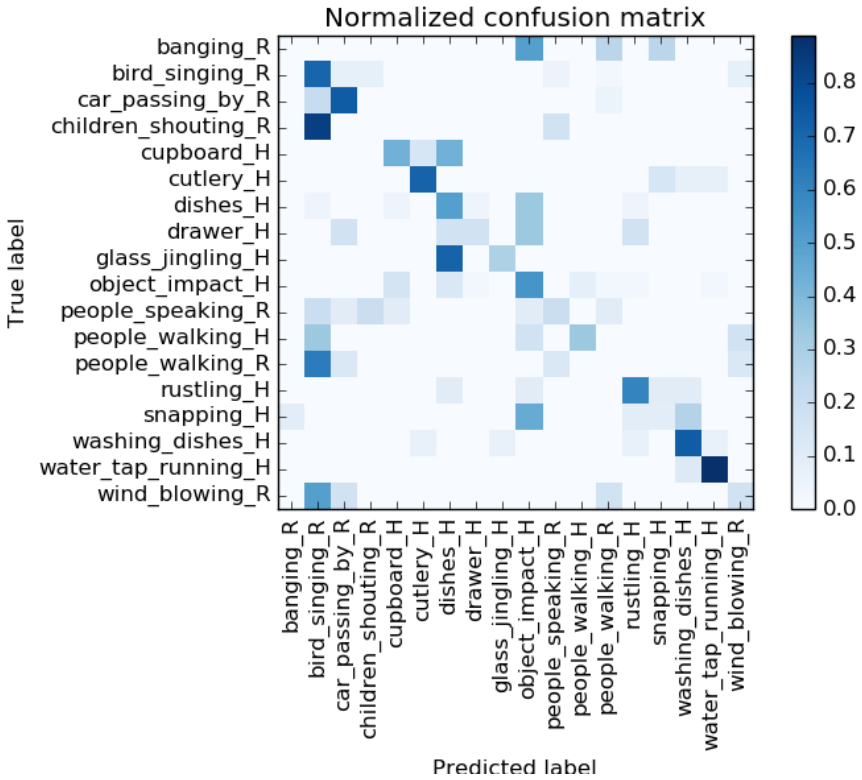


Figure 3.7: [H]ome and [R]esidential without the generic class. *Object impact* and *bird singing* capture most of ambiguities.

3.3.3 Generation of data through perturbation

The scarcity of labeled data for each event is a common issue. Annotations are costly, all sound events don't occur with the same frequency, and it is also hard to capture enough variations of the same sound to train robust models.

To address the problem of data scarcity, many data augmentation techniques have been proposed in the literature for different tasks. One example is [58] where the authors augment data by perturbing the audio signal in multiple ways and achieve improved speech separation performance. We empirically analyzed multiple combinations of time-based perturbations, by speeding up and slowing down the sound event samples to different levels. We found that speeding up the original signal by more than 30% resulted in unintelligible audio. Also, slowing down the signal by more than 100% didn't make sense. The range of rate changes that we explored included 13 different speed factors and the original signal at its normal rate.

As before, the experimental data included separate scenes from home (inside homes) and residential areas (outside homes but within residential areas). Four-fold cross-validation experiments were performed with that data, augmented with perturbed audio that comprised the original data and 13 different rate-changed versions of it. The test set was not augmented and remained intact. The results are shown in Table 3.3, where

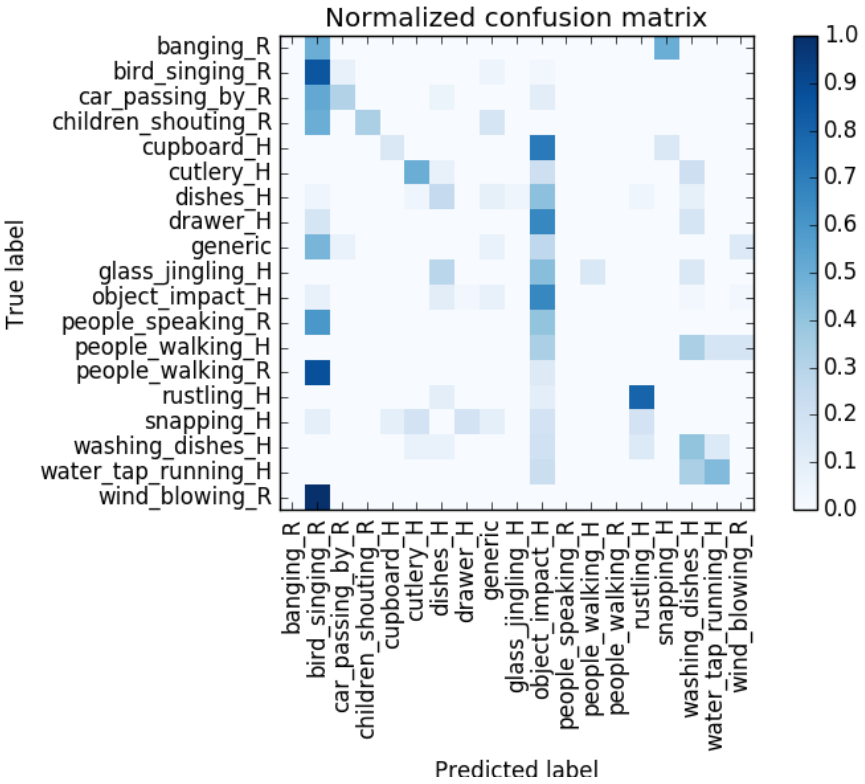


Figure 3.8: [H]ome and [R]esidential with the generic class. Although this class shared the background acoustics with the 18 sounds, it didn't cause major confusion.

the performance for *Home* improved, but not for *Residential*.

3.3.4 Sound event detection and submission systems

For Task 3 in DCASE we used the setup described above with data augmentation using rate-perturbation. Sound events were extracted from each scene using the given annotations. MFCC features were then extracted and a Tpot-optimized multi-class classifier was trained. For testing, we segmented the scene recordings from the test set into sequences of one-second-long segments. This duration was selected due to the metric schema of the DCASE evaluation, which considers one-second segments.

Three experiments were performed: without the generic class or perturbation, with generic class and perturbation (G+P), and with generic class but without perturbation (G). The results of the development-test set are shown in Table 3.4. The inclusion of the generic class and perturbation outperformed the baseline by a significant margin for both *Home* and *Residential* scenes. G and G+P on the evaluation set achieved SBER of 0.9613 for G and Fscore of 33.6% for G+P.

3.3.5 Conclusion from Acoustic scene classification

We used different approaches for both acoustic scene classification (Task 1) and sound event detection (Task 3) of the 2016 DCASE challenge. On both tasks, we were able

Acoustic Scene	SBER	F-score
Home	1.05	25.1
Home + G	0.91	23.7
Home + G + P	0.9	24.7
Residential	0.64	54.6
Residential + G	0.72	45.9
Residential + G + P	0.63	52.2

Table 3.4: Our Segment-based Error Rate, using [G]eneric and [P]erturbation, outperformed the baseline.

to obtain significant improvement over the baseline method. For Task 1 we observed that the $\vec{\beta}$ features performed much better than $\vec{\alpha}$ features. Although linear and RBF kernels with $\vec{\beta}$ features can outperform the baseline by a considerable margin on their own, we make note of the fact that a multiple classifier system can give further improvements. For Task 3, we tested different classifiers and significantly improved the baseline. Moreover, we explored a way of handling out-of-vocabulary sound segments with the generic class and the inclusion of perturbed audio to add robustness.

3.4 Contribution to DCASE Task 4: Sound event detection using weak labels

Task 4 evaluated systems for the large-scale detection of sound events using weakly labeled audio recordings. The audio comes from YouTube video excerpts related to the topic of transportation and warnings. The topic was chosen due to its industry relevance and the underuse of audio in this context. The results will help define new grounds for large-scale sound event detection and show the benefit of audio for self-driving cars, smart cities, and related areas.

The task consisted of detecting sound events within 10-second clips and it was divided into two subtasks:

- Subtask A: Without timestamps (same as audio tagging, Fig 4)
- Subtask B: With timestamps (similar to Task 3, Fig 3)

The task employed a subset of AudioSet[38]. AudioSet consists of an ontology of 632 sound event classes and a collection of 2 million human-labeled 10-second sound clips drawn from YouTube videos. The ontology is specified as a hierarchical graph of event categories, covering a wide range of human and animal sounds, musical instruments and genres, and common everyday environmental sounds. To collect the dataset, Google worked with human annotators who listened, analyzed, and verified the sounds they heard within the YouTube 10-second clips. To facilitate faster accumulation of examples

for all classes, Google relied on available YouTube metadata and content-based search to nominate candidate video segments that were likely to contain the target sound. Note that AudioSet does not come with precise time boundaries for each sound class within the 10-second clips and thus annotations are considered weak labels. Also, one clip may correspond to more than one sound event class. Task 4 relied on a subset of 17 sound events divided into two categories: Warning and vehicle sounds.

- **Warning sounds**: Train horn, Air horn Truck horn, Car alarm, Reversing beeps, Ambulance (siren), Police car (siren), Fire engine fire truck (siren), Civil defense siren, Screaming.
- **Vehicle sounds**: Bicycle, Skateboard, Car, Car passing by, Bus, Truck, Motorcycle, Train.

For both subtasks, the data was divided into two main partitions: development and evaluation. The development data was divided into training and testing. Training had 51,172 clips, which are class-unbalanced, and had at least 30 clips per sound event. The test had 488 clips, with at least 30 clips per class. A 10-second clip may have corresponded to more than one sound event class. The evaluation set had 1,103 clips, with at least 60 clips per sound event. The sets had weak labels denoting the presence of a given sound event within the audio but with no timestamp annotations. For testing and evaluation, strong labels (timestamp annotations) were provided to evaluate performance on Subtask B. The task rules did not allow the use of external data, such as other datasets. Similarly, it was not allowed to use other elements.

We followed the success of our work in the subsequent DCASE challenges and the following describes our work on DCASE Task 4 in 2019. Task 4 involves training systems for large-scale detection of sound events using a combination of weakly labeled data, i.e., training labels without time boundaries, and strongly-labeled synthesized data. For this we introduced the Domestic Environment Sound Event Detection (DESED) dataset, mixing a part of prior year's dataset with an additional synthetic, strongly labeled dataset.

The context for this section's work is DCASE 2018 Task 4 [59], where we investigate how large-scale SED systems can exploit a small set of weakly annotated data, a larger set of unlabeled data and an additional training set of synthetic soundscapes with strong labels.

Given these data, the goal of this task is to train SED models that output event detections with time boundaries (i.e. strong predictions) in domestic environments.

Class	Unique events	Dev set	
		Clips	Events
Alarm/bell/ringing	190	392	755
Blender	98	436	540
Cat	88	274	547
Dishes	109	444	814
Dog	136	319	516
Electric shaver/toothbrush	56	221	230
Frying	64	130	137
Running water	68	143	157
Speech	128	1272	2132
Vacuum cleaner	74	196	204
Total	1011	2045	6032

Table 3.5: Class-wise statistics for the synthetic development subset.

3.4.1 Task description

This task is the follow-up to the DCASE 2018 Task 4 [59] and focuses on the same 10 classes of sound events. Systems are expected to produce strongly labeled output (i.e. detect sound events with a start time, end time, and sound class label), but are provided with weakly labeled data (i.e. sound recordings with only the presence/absence of a sound included in the labels without any timing information) for training. Multiple events can be present in each audio recording, including overlapping events. As in the previous iteration of this task, the challenge entails exploiting a large amount of unbalanced and unlabeled training data together with a small weakly annotated training set to improve system performance.

The advantage of having an additional training set with strongly annotated synthetic soundscapes in the 2019 challenge was that this allowed us to explore scientific questions around the informativeness of real (but weakly labeled) data versus strongly labeled synthetic data, to explore whether the two data sources are complementary or not, and to find out how to best leverage these datasets to optimize system performance.

3.4.2 DESED development dataset

The DESED development dataset is composed of 10-sec audio clips recorded in a domestic environment or synthesized to simulate a domestic environment. The real recordings are taken from AudioSet [16]. The development data set is divided into two subsets: (i) A training subset composed of real recordings similar to DCASE 2018 task 4 [60] and synthetic soundscapes generated using Scaper (see also Table 3.5). (ii) A validation subset composed of real recordings with strongly labeled data, which is the

combination of the validation and evaluation sets from DCASE 2018 Task 4.

3.4.2.1 Synthetic soundscape generation procedure

The subset of synthetic soundscapes comprises 10-second audio clips generated with Scaper [44], a Python library for soundscape synthesis and augmentation. Scaper operates by taking a set of foreground sounds and a set of background sounds and automatically sequencing them into random soundscapes sampled from a user-specified distribution, controlling the number and type of sound events, their duration, signal-to-noise ratio, and several other key characteristics. The foreground events are obtained from the Freesound Dataset (FSD) [61, 62]. Each sound event clip was verified by a human to ensure that the sound quality and event-to-background ratio were sufficient to be used as an isolated sound event. We also controlled for whether the sound event onset and offset were present in the clip. Each selected clip was then segmented when needed to remove silences before and after the sound event and between sound events when the file contained multiple occurrences of the sound event class. The number of unique isolated sound events per class used to generate the subset of synthetic soundscapes is presented in Table 3.5. We also list the number of clips containing each sound class and the number of events per class.

The background textures are obtained from the SINS dataset (activity class "other") [63]. This particular activity class was selected because it contains a low amount of sound events from our 10 target foreground sound event classes. However, there is no guarantee that these sound event classes are completely absent from the background clips. A total of 2060 unique background clips are used to generate the synthetic subset.

Scaper scripts are designed such that the distribution of sound events per class, the number of sound events per clip (depending on the class), and the sound event class co-occurrence are similar to that of the validation set, which is composed of real recordings. The synthetic soundscapes are annotated with strong labels that are automatically generated by Scaper [44].

3.4.3 DESED evaluation dataset

The evaluation set is composed of two subsets: a subset with real recordings and a subset with synthetic soundscapes.

3.4.3.1 Real recordings

The first subset contains 1,013 audio clips and is used for ranking purposes. It comprises audio clips extracted from 692 YouTube and 321 Vimeo videos under creative commons

Class	Unique events	Synth set 1	
		Clips	Events
Alarm/bell/ringing	63	101	184
Blender	27	84	95
Cat	26	113	197
Dishes	34	161	293
Dog	43	124	217
Electric shaver/toothbrush	17	113	117
Frying	17	52	52
Running water	20	67	73
Speech	47	471	803
Vacuum cleaner	20	92	93
Total	314	1378	2124

Table 3.6: Class-wise statistics for the synthetic evaluation subsets

licenses. Each clip is annotated by a human and annotations are verified by a second annotator.

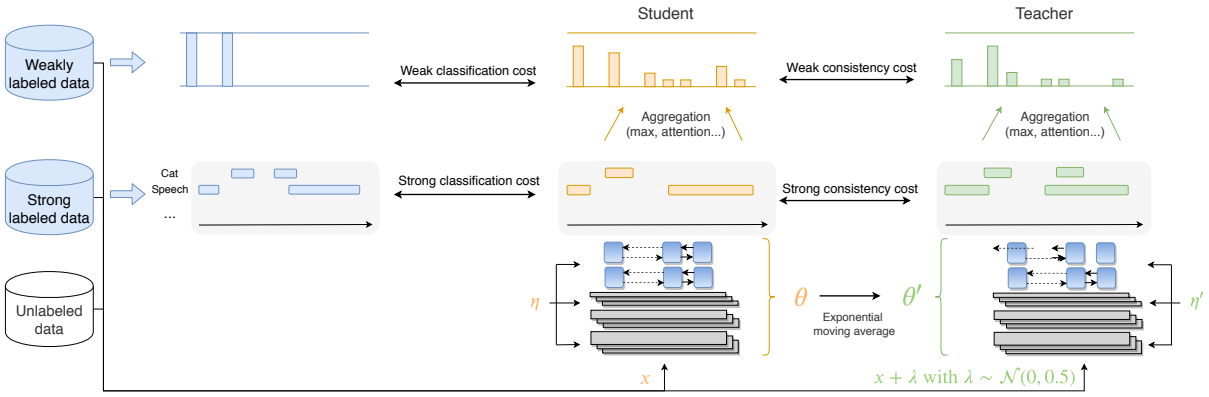


Figure 3.9: Mean-teacher model. η and η' represent noise applied to the different models (in this case dropout).

3.4.3.2 Synthetic soundscapes

The second subset is comprised of synthetic soundscapes generated with Scaper[1]. This subset is used for analysis purposes and its design is motivated by the analysis of last year's results [60]. In particular, most submissions from the previous year performed poorly in terms of event segmentation. One of the goals of this subset is to facilitate studies on the extent to which including strongly labeled data in the training set helps

[1]The JAMS [64] annotation files corresponding to these soundscapes can be accessed from DCASE website: http://dcase.community/.

improve and refine the segmentation output. The foreground events are obtained from the FSD [61, 62]. The selection process was the same as described for the development dataset. Background sounds are extracted from YouTube videos under a Creative Common license and from the Freesound subset of the MUSAN dataset [65]. The synthetic subset is further divided into several subsets (described below) for a total of 12,139 audio clips synthesized from 314 isolated events. The distribution of isolated sound events per class is presented in Table 3.6.

3.4.3.3 Varying foreground-to-background SNR

A subset (denoted Synthetic set 1) of 754 soundscapes is generated with a sound event distribution similar to that of the training set. Four versions of this subset are generated varying the value of the foreground events' SNR with respect to the background: 0 dB, 6 dB, 15 dB, and 30 dB.

3.4.3.4 Audio degradation

Six alternative versions of the previous subset (with SNR = 0dB) are generated introducing artificial degradation with the Audio Degradation Toolbox [66]. The following degradations are used (with default parameters): "smartPhonePlayback," "smartPhoneRecording," "unit_applyClippingAlternative," "unit_applyLowpassFilter," "unit_applyHighpassFilter" and
"unit_applyDynamicRangeCompression".

3.4.3.5 Varying onset time

A subset of 750 soundscapes is generated with uniform sound event onset distribution and only one event per soundscape. The SNR parameter is set to 0 dB. Three variants of this subset are generated with the same isolated events, only shifted in time. In the first version, all sound events have an onset located between 250 ms and 750 ms, in the second version the sound event onsets are located between 4.75 s and 5.25 s, and in the last version the sound event onsets are located between 9.25 s and 9.75 s.

3.4.3.6 Long sound events vs. short sound events

A subset with 522 soundscapes is generated where the background is selected from one of the five long sound event classes (Blender, Electric shaver/toothbrush, Frying, Running water, and Vacuum cleaner). The foreground sound events are selected from the five short sound event classes (Alarm/bell/ringing, Cat, Dishes, Dog, and Speech).

Three variants of this subset are generated with similar sound event scripts and varying values of the sound event SNR parameter (0 dB, 15 dB, and 30 dB).

3.4.4 Baseline

We designed the baseline system[2] inspired by the winning system from the DCASE 2018 Task 4 by Lu [67]. It uses a mean-teacher model which is a combination of two models: a student model and a teacher model (both have the same architecture). Our implementation of the mean-teacher model is based on the work of Tarvainen and Valpola [68]. The student model is the final model used at inference time, while the teacher model is aimed at helping the student model during training and its weights are an exponential moving average of the student model's weights. A depiction of the baseline model is provided in Figure 3.9.

The models are a combination of a convolutional neural network (CNN) and a recurrent neural network (RNN) followed by an aggregation layer (in our case an attention layer). The RNN output gives strong predictions (the weights of this model are denoted θ_s) while the output of the aggregation layer gives weak predictions (the weights of this model are denoted θ).

The student model is trained on synthetic and weakly labeled data. The loss (binary cross-entropy) is computed at the frame level for the strongly labeled synthetic data and at the clip level for the weakly labeled data. The teacher model is not trained, rather, its weights are a moving average of the student model (at each epoch). During training, the teacher model receives the same input as the student model but with added Gaussian noise, and helps train the student model via a consistency loss (mean-squared error) for both strong (frame-level) and weak predictions. Every batch contains a combination of unlabeled, weakly and strongly labeled samples.

This results in four loss components: two for classification (weak and strong) and two for consistency (weak and strong), which are combined as follows:

$$\begin{aligned} L(\theta) =& L_{class_w}(\theta) + \sigma(\lambda) L_{cons_w}(\theta) \\ &+ L_{class_s}(\theta_s) + \sigma(\lambda) L_{cons_s}(\theta_s) \end{aligned} \tag{3.8}$$

3.4.5 Submission evaluation

DCASE 2019 Task 4 obtained 57 submissions from 18 different teams involving 60 researchers overall.

[2]Open source code available at: https://github.com/cmu-mlsp/ankit-phd-thesis/DCASE2019_task4/tree/public/baseline

Rank	System	Classifier	Real recordings Event-based **Eval**	Youtube	Vimeo	Valid	Segment-based Eval	Synthetic Event-based Set 1
1	**Lin, ICT**	CNN	**42.7%**	47.7%	29.4%	45.3%	64.8%	47.6%
2	**Delphin-Poulat, OL**	CRNN	**42.1%**	45.8%	33.3%	43.6%	71.4%	59.8%
3	**Shi, FRDC**	CRNN	**42.0%**	46.1%	31.5%	42.5%	69.8%	53.2%
4	**Cances, IRIT**	CRNN	**39.7%**	43.0%	30.9%	39.9%	64.7%	50.8%
5	**Yan, USTC**	CRNN	**36.2%**	38.8%	28.7%	42.6%	65.2%	41.8%
6	**Lim, ETRI**	CRNN, Ensemble	**34.4%**	38.6%	23.7%	40.9%	66.4%	42.5%
7	**Kiyokawa, NEC**	ResNet, SENet	**32.4%**	36.2%	23.8%	36.1%	65.3%	42.3%
8	**Chan, NU**	NMF, CNN	**31.0%**	34.7%	21.6%	30.4%	58.2%	46.7%
9	**Zhang, UESTC**	CNN,ResNet,RNN	**30.8%**	34.5%	21.1%	35.6%	60.9%	49.2%
10	**Kothinti, JHU**	CRNN, RBM, CRBM, PCA	**30.7%**	33.2%	23.8%	34.6%	53.1%	35.6%
11	**Wang B., NWPU**	CNN, RNN, ensemble	**27.8%**	30.1%	21.7%	31.9%	61.6%	32.9%
12	**Lee, KNU**	CNN	**26.7%**	28.1%	22.9%	31.6%	50.2%	33.0%
	Baseline 2019	CRNN	**25.8%**	29.0%	18.1%	23.7%	53.7%	40.6%
13	**Agnone, PDL**	CRNN	**25.0%**	27.1%	20.0%	59.6%	60.4%	46.7%
14	**Rakowski, SRPOL**	CNN	**24.2%**	26.2%	19.2%	24.3%	63.4%	29.7%
15	**Kong, SURREY**	CNN	**22.3%**	24.1%	17.0%	21.3%	59.4%	23.6%
16	**Mishima, NEC**	ResNet	**19.8%**	21.8%	15.0%	24.7%	58.7%	33.0%
17	**Wang D., NUDT**	CRNN	**17.5%**	19.2%	13.3%	22.4%	63.0%	14.0%
18	**Yang, YSU**	CMRANN-MT	**6.7%**	7.6%	4.6%	19.4%	26.3%	7.5%

Table 3.7: F1-score performance on the evaluation sets

3.4.5.1 Evaluation metrics

Submissions were evaluated according to an event-based F1-score with a 200 ms collar on the onsets and a collar on the offsets that is greater of 200 ms and 20% of the sound event's length. The overall F1 score is the unweighted average of the class-wise F1 scores (macro-average). In addition, we provide the segment-based F1-score on 1 s segments as a secondary measure. The metrics are computed using the sed_eval library [69].

3.4.5.2 System performance

The official team ranking (best system from each team) along with some characteristics of the submitted systems is presented in Table 3.7. Submissions are ranked according to the event-based F1 score computed over the real recordings in the evaluation set. For a more detailed comparison, we also provide the event-based F1-score on the YouTube and Vimeo subsets and the segment-based F1-score over all real recordings. The event-based F1-score on the validation set is reported for the sake of comparison with last year's results (75% of the 2019 validation set is comprised of the 2018 evaluation set). Performance on synthetic recordings is not taken into account in the ranking, but the event-based F1-score on Synthetic set 1 (0 dB) is presented here as well. The baseline for DCASE 2018 would obtain 22.2% F1-score on the evaluation set.

Twelve teams outperform the baseline with the best systems [70, 71, 72] outperforming the baseline by 16% points and the best system from 2018 by over 10 % points. While the ranking on the YouTube subset is similar to the official ranking, there rankings based on the Vimeo and synthetic subsets are notably different. Performance on the

Vimeo set is in general considerably lower than on the YouTube set and Synthethic set 1. The fact that no data from Vimeo was used during training (unlike data from YouTube and synthetic data) suggests that the submitted systems struggle to generalize to an entirely unseen set of recording conditions.

All three top-performing teams used a semi-supervised mean-teacher model [68]. Lin et al. [70] focused on the importance of semi-supervised learning with a guided learning setup [73] and on how synthetic data can help when used together with a sufficient amount of real data. Delphin-Poulat et al. [71] focused on data augmentation and Shi [72] focused on a specific type of data augmentation where both audio files and their labels are mixed. Cances et al. [74] proposed a multi-task learning setup where audio tagging (producing weak predictions) and the sound event localization in time (strong predictions) are treated as two separate subtasks [75]. The latter was also the least complex of the top-performing systems.

Most of the top-performing systems also demonstrate the importance of employing class-dependent post-processing [70, 71, 74], which improves performance significantly compared to e.g. using a fixed median filtering approach. This highlights the benefits of applying dedicated segmentation post-processing [74, 76].

3.4.5.3 Conclusion from DCASE Task 4 Analysis

For Task 4, we observe that the best submissions outperform the prior winning submission by over 10 % points, representing a notable advancement. Performance tends to show that the proposed systems in general did not overfit the soundscape data set and still performed well on the real data set. However, the performance from all systems degrades in the unseen Vimeo subset, which indicates a lack of ability to generalize. The progress is encouraging, but the remaining limitations indicate possible directions for follow-up to this task.

Despite this challenge, we studied the robustness of the systems to noise and signal degradation, which is known to impact model generalization. Our analysis is based on the results of task 4. A comparative analysis of the performance of state-of-the-art sound event detection systems shows that while overall systems exhibit significant improvements compared to previous work, they still suffer from biases that could prevent them from generalizing to real-world scenarios.

3.4.6 Data augmentation: Additional experiments

In Task 4, the synthetic soundscapes evaluation dataset included multiple subsets comprised of the same set of synthetic soundscapes, each with a different type of data degradation applied via the audio degradation toolbox [66] or with for different

foreground-to-background signal-to-noise ratio (FBSNR), that is the ratio between the loudness of the sound event (foreground) and the loudness of the background noise. We also considered other subsets where the same sound event would be localized at different time instants within a sound clip and subsets using the long sound event classes as background noise. All these scenarios are realistic and represent conditions where the submitted systems are likely to fail. However, gathering enough real data that would cover each of these aspects is hardly feasible. Not to mention that in some cases (e.g., varying FBSNR) it would require recording new data to ensure that only the tested parameter is changing between experiments. Synthetic soundscapes then offer a flexible and realistic alternative to performance preliminary tests before considering further investigation on real data if need be.

In the section below we present the results obtained by the participants of DCASE 2019 task 4 on the evaluation composed of synthetic soundscapes. This performance was neither presented nor analyzed within the challenge. We propose an analysis of the robustness of the ten top-performing approaches (on the evaluation set composed of synthetic soundscapes) to degradation of the recording quality and to varying FBSNR and a study of the robustness of the segmentation process implemented in the submitted systems.

3.4.6.1 Varying foreground-to-background SNR

A subset of 754 soundscapes was generated with Scaper scripts. The scripts are designed such that the distribution of sound events per class, the number of sound events per clip (depending on the class), and the sound event class co-occurrence are similar to that of the validation set which is composed of real recordings. The foreground event signal-to-noise ratio (SNR) parameter was uniformly drawn between 6 dB and 30 dB. Four versions of this subset were generated varying the value of the background SNR parameter:

- 0 dB (the FBSNR is between 6 dB and 30 dB);
- 6 dB (the FBSNR is between 0 dB and 24 dB);
- 15 dB (the FBSNR is between -9 dB and 15 dB);
- 30 dB (the FBSNR is between -24 dB and 0 dB).

These subsets are referred to as **fbsnr_30dB**, **fbsnr_**24**dB**, **fbsnr_15dB** and **fbsnr_0dB**, respectively. These subsets are designed to study the impact of the SNR on the SED system's performance. Related results are discussed in Section 3.4.9.1.

3.4.6.2 Audio degradation

To study the robustness of the SED to audio degradation, six alternative versions of the subset **fbsnr_30dB** were generated introducing artificial degradation with the Audio Degradation Toolbox [66]. The signal degradations are generated to simulate degradation faced in real environments. The following degradations are used (with default parameters) : "smartPhonePlayback," "smartPhoneRecording," "unit_applyClippingAlternative," "unit_applyDynamicRangeCompression," "unit_applyLowpassFilter" and "unit_applyHighpassFilter." We refer to these versions as **phone_play**, **phone_record**, **clipping**, **compression**, **lowpass** and **highpass**, respectively. Performance results for these are discussed in Section 3.4.9.

3.4.6.3 Varying onset time

A subset of 750 soundscapes was generated with uniform sound event onset distribution and only one event per soundscape. The parameters were set such the FBSNR was between 6 dB and 24 dB. Three variants of this subset were generated with the same isolated events, only shifted in time. In the first version, all sound events had an onset located between 250 ms and 750 ms, in the second version the sound event onsets were located between 4.75 s and 5.25 s, and in the last version the sound event onsets were located between 9.25 s and 9.75 s. We will refer to these subsets as **500ms**, **5500ms** and **9500ms**, respectively. These are designed to study the sensitivity of the SED segmentation to sound event location in time. In particular, we wanted to control if SED systems were learning a bias in terms of time localization depending on the event length (e.g., long sound events would most often start at the beginning of the sound clip). Related results are discussed in Section 3.5.

3.4.6.4 Long sound events vs. short sound events

A subset with 522 soundscapes was generated where the background was selected from one of the five long sound event classes (Blender, Electric shaver/toothbrush, Frying, Running water, and Vacuum cleaner). The foreground sound events were selected from the five short sound event classes (Alarm/bell/ringing, Cat, Dishes, Dog, and Speech). Three variants of this subset were generated with the same sound event scripts, and varying values of the background SNR parameter: 0, 15, and 30 dB in the first, second, and third subsets respectively. These subsets are referred to as **ls_0dB**, **ls_15dB** and **ls_30_dB**, respectively. This experiment is designed to study the impact of a sound event being in the background or the foreground on SED performance [77]. Related results are discussed in Section 3.4.9.1.

3.4.7 Evaluation metrics

Submissions were evaluated with event-based measures for which the system output was compared to the reference labels event by event [69]. The correspondence between sound event boundaries was estimated with a 200ms tolerance collar on onsets and a tolerance collar on offsets that are a maximum of 200ms and 20% of the duration of the sound event. When sound event classes are taken into account, the overall F1 score is the unweighted average of the class-wise F1 scores (macro-average). The metrics are computed using the sed_eval library [69].

3.4.8 Robustness to noise and degradations

In this section, we focus on the impact of signal degradation on the SED and the FBSNR performance. Each participant was allowed to submit up to four different systems. Only the F1-score for the top-performing system (on **fbsnr_24dB**) for each participant is presented here. We limit the analysis to the 10 top-performing systems. Of the top 10 performing systems, 9 were based on CNN and 7 included recurrent layers, the most common input features were log-mel energies and the most common post-processing approach was median filtering. For further details about each system see the submission reports [78, 74, 79, 71, 80, 81, 70, 72, 82, 83].

3.4.9 Simulated degradations

System	Event-based F1-score						
	fbsnr_24dB	**phone_play**	**phone_record**	**clipping**	**compression**	**highpass**	**lowpass**
Agnone, PDL [78]	39.1%	15.4%	9.2%	14.6%	29.6%	8.5%	0.9%
Cances, IRIT [74]	47.1%	25.7%	35.8%	42.6%	44.3%	19.2%	1.2%
Chan, NU [79]	41.2%	25.9%	17.5%	22.8%	33.4%	19.3%	1.2%
Delphin-Poulat, OL [71]	**53.6%**	32.9%	23.7%	29.5%	48.2%	**23.3%**	**4.8%**
Kiyokawa, NEC [80]	36.8%	33.9%	21.9%	35.6%	40.2%	22.1%	4.2%
Lim, ETRI [81]	38.9%	26.9%	30.3%	39.7%	48.1%	15.4%	0.7%
Lin, ICT [70]	43.7%	22.4%	9.3%	19.8%	35.3%	17.6%	0.5%
Shi, FRDC [72]	46.4%	**35.0%**	**36.4%**	**48.3%**	**54.1%**	17.4%	4.0%
Yan, USTC [82]	36.5%	22.1%	21.5%	18.3%	32.7%	16.6%	1.0%
Zhang, UESTC [83]	43.7%	21.8%	15.3%	24.6%	41.4%	14.1%	1.7%
Average score (all participants)	33.9%	22.0 %	16.4%	21.6%	31.4%	15.8%	1.7%

Table 3.8: F1-score performance on the degraded synthetic soundscapes

The F-1 score obtained on the degraded subsets is presented in Table 3.8. The performance on **fbsnr_24dB** subset is presented here for comparison.

Some systems seemingly over-fitted the synthetic soundscapes subset of the training set and as a result, their performance decreased for most of the degradations. Otherwise, the trend was similar for most of the systems. The submitted systems were found to be robust to smartphone-related degradations and compression. This can be related to the fact that they had been trained on audio data extracted from YouTube, which had

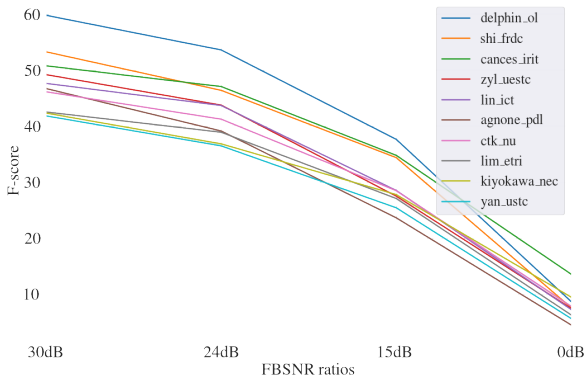


Figure 3.10: SED performance depending on the FBSNR.

probably been recorded using smartphones. On the other hand, all systems seem to be very sensitive to low-pass and high-pass filtering which tends to indicate that systems are not robust to changes in the frequency range of the input representation between training and test.

3.4.9.1 Foreground-to-background Signal-to-noise ratio

In Figure 3.10, we present the F1-score performance for the 10 top-performing systems mentioned above under varying FBSNR (see Section 3.4.6.1). The trend for all systems is similar, so no submission stands out in terms of robustness to noise. Interestingly, on **fbsnr_15dB** where FBSNR should be distributed almost evenly around 0 dB, F1-score performance is still acceptable for most systems and remain in the range of what was obtained on real recording clips [84]. Unsurprisingly, on **fbsnr_0dB**, the FBSNR is always negative and the performance for all systems collapses.

We then propose to analyze the systems' performance when the background is actually one event from the long sound event classes and the foreground sound events are selected from the short sound event classes (see Section 3.4.6.4). In Figure 3.11, we present the F1-score performance for the 3 top performing systems (on this particular task) together with the performance averaged over all systems.

In all cases, when the FBSNR is low, all systems consistently obtain better performance on long sound event classes. On the other hand, when the FBSNR is high, all systems obtain better performance on short sound event classes. When the FBSNR is 0dB, most systems perform similarly on short sound event classes and long sound event classes. This shows that the bias toward long event classes observed in DCASE 2018 [59, 60] is less important in this (2019) case. The trend is confirmed by the performance on short or long sound event classes that are within the same range in the most favorable cases (0 dB FBSNR for the long sound event classes, 30 dB for the

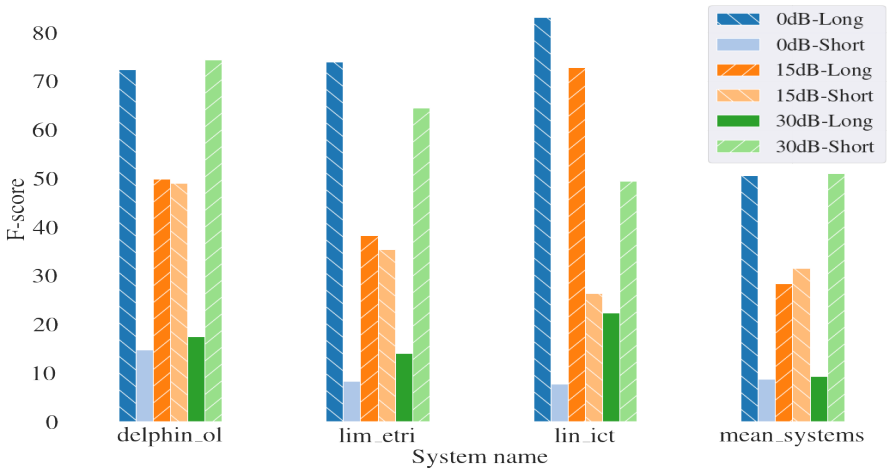


Figure 3.11: SED performance and FBSNR for soundscape composed of a long event and multiple short events.

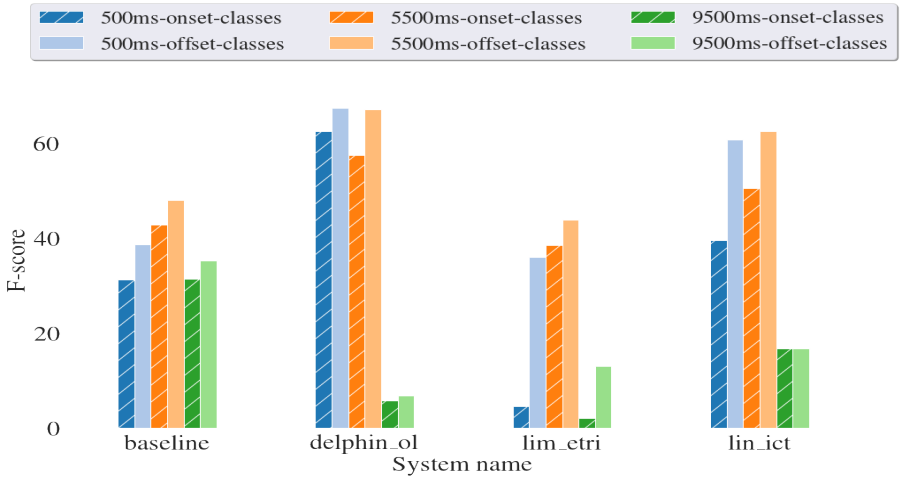


Figure 3.12: Segmentation (event localization) performance for long sound event classes.

short sound event classes). However, the system submitted by Lin et al. [70] does not follow this trend and mostly performs better on long events. This could be due to the guided learning methods that biases the mean teacher model. The teacher labels are converted to 0/1 predictions instead of probabilities which may increase the number of with a positive label and introduce a bias towards long events.

3.5 Segmentation

We now focus on system performances for segmentation, i.e., sound event localization time (regardless of the sound event class). Event segmentation refers to finding both the onset and offset time for each sound event. We especially consider the scenario in

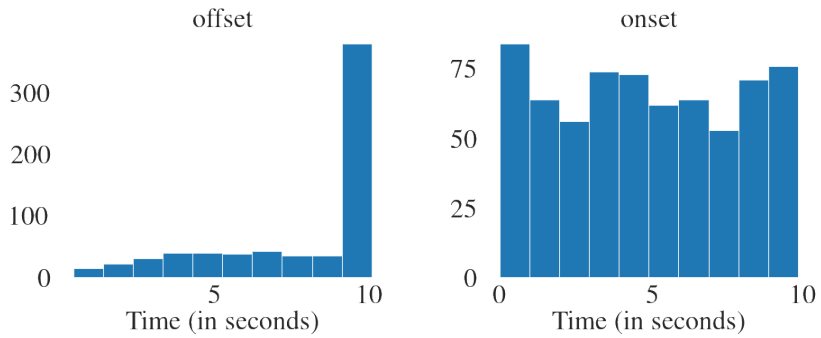


Figure 3.13: Long sound event classes: Time distribution of the onsets and the offsets in the synthetic soundscapes subset of the DESED training set.

Section 3.4.6.3, in which three versions of a sound clip are generated with the same background and the same sound event starting either at the beginning, in the middle, or at the end of the sound clip. The F1-score performance is presented for the 3 top-performing systems (on this particular task) together with the performance of the baseline system [84].

For short sound event classes, the F1-score performance is similar wherever the sound event is located within the segment.

For the long sound event classes, performances are shown in Figure 3.12. This shows the F1-score for segmentation, onset, and offset detection for long sound event classes (but regardless of the sound event class). We see that the position of the long sound event in the clip affects the performance, which becomes worse when the sound event is near the end of the clip.

One reason could be that in the training set, long sound events start and end largely at the start of the clips. However, Figure 3.13 shows that long and short sound events start at similar times. Long sound events also often end near the end of the clip. Thus, if there was any bias from training, it should have helped to find better endings for long sound events towards the end of the clips. This is true – all systems can find good endings for long sound events that start in the middle of the clip. But in this case, most of these sound events also end near the end of the clip (see Figure 3.14). An alternative explanation is that the submitted systems are simply not able to detect a long sound event class toward the end of the sound clip. For example, median filtering with variable length, which is used in most of the submissions (more than 0.5 s for long sound event classes in some cases [70, 71]), would make it unlikely to detect a long sound event class at the end of the sound clip. This hypothesis is confirmed by the fact that the baseline that uses fixed-length median filtering as post-processing performs similarly regardless of where the sound event is located.

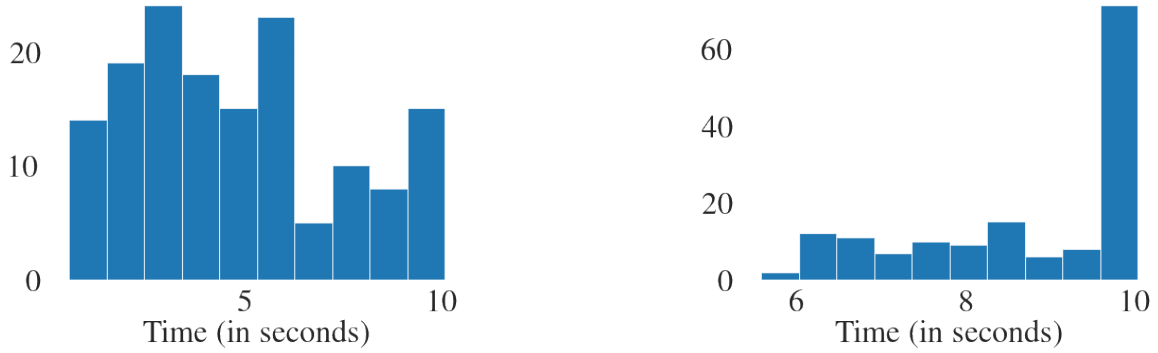


Figure 3.14: Time distribution of the offsets for long event classes in the subsets **500ms** (left) and **5500ms** (right) of DESED Evaluation set.

In summary, we tested all DCASE 2019 task 4 submissions on synthetic soundscapes. The analysis revealed that using internet video sound clips to train SED makes the systems handle smartphone sound quality better. But despite the improvement relative to the DCASE 2018 task 4, SED systems still depend on biases (especially for segmentation) that would limit their use in real situations. As a mitigation strategy, in DCASE 2019 we added a real recording from an unknown source (Vimeo) to the evaluation set.

A possible solution for the segmentation problem would be to make the evaluation set include the extreme cases discussed above and use separate sound events (not soundscapes) so that more diverse training examples could be created. This could reduce some of the bias in system design and learning for this task.

3.6 Key insights

The above chapter outlines advancements in the Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge, focusing on various tasks related to automatic audio event and scene recognition. The chapter provides a detailed exploration of six primary tasks, including acoustic scene classification, sound event detection, anomalous sound detection, and more.

Key tasks as part of DCASE include:

- Acoustic Scene Classification: Progress in this field has been made, with state-of-the-art systems leveraging convolutional neural networks (CNNs), attention mechanisms, and data augmentation to improve generalization across devices. Low-complexity solutions were also explored, with techniques such as depth-wise separable convolutions enabling more efficient models.

- Anomalous Sound Detection: This task highlighted the use of deep learning-based outlier detection techniques, with impressive results in detecting machine anomalies. However, challenges remain in dealing with false positives and highly variable anomalies.

- Sound Event Localization and Detection (SELD): Here, top-performing models employed convolutional recurrent neural networks (CRNNs) to achieve accurate spatiotemporal characterization of sound events. The systems performed robustly in real-world environments, but further work is needed on handling domain shifts and overlapping events.

- Sound Event Detection in Domestic Environments: Combining weakly labeled, unlabeled, and synthetic data, this task demonstrated how data fusion and domain adaptation techniques can be used to detect sound events in noisy domestic environments.

- Few-shot Bioacoustic Event Detection: This task showcased the effective use of few-shot learning to detect novel animal vocalizations with only a handful of examples.

- Automated Audio Captioning and Retrieval: Advances in generating free-text descriptions for audio and retrieving relevant audio clips based on text queries were explored, pointing to future applications in multimedia retrieval and intelligent machine interaction.

The chapter effectively illustrates how cutting-edge machine learning techniques, especially deep learning architectures like CNNs and CRNNs, have driven advances in acoustic scene classification and event detection. It emphasizes the importance of model generalization, domain adaptation, and handling real-world challenges such as device variability and overlapping sound events. Despite the substantial progress, open challenges like false positives, overlapping events, and segmentation accuracy indicate that further research is needed.
[3]

[3]Code is available at https://github.com/ankitshah009/\/Task-4-Large-scale-weakly-supervised-sound-event-detection-for-smart-cars

Section III: Training models with Weak labels

Training models with weak labels is a crucial aspect of audio classification and sound event detection, especially when access to precisely labeled data is limited. Chapters 4, 5, and 6 lay the foundation for understanding the methodologies and challenges associated with learning from weakly labeled data.

Chapter 4 explores the foundational concepts of learning from weak labels, emphasizing the importance of weak labels in audio classification tasks. It introduces various strategies to mitigate the challenges posed by weak labels, such as handling label noise, label corruption, and label density. The chapter further discusses the effectiveness of Convolutional Neural Networks (CNNs) designed specifically for weakly labeled audio data, presenting experimental designs and results that highlight the impact of label quality and density on model performance.

Chapter 5 addresses the complexities of learning from noisy and web-sourced data. It presents the WeblyNet system, a novel approach for leveraging weak labels obtained from the internet, which are inherently noisy and imprecise. The chapter details the challenges of using such data and proposes a co-training method that significantly improves model performance compared to directly using weak labels. This method relies on using two complementary networks to refine the learning process, demonstrating the potential of webly labeled data to supplement traditional training datasets.

Chapter 6 extends the discussion to semi-weak labels, which incorporate additional cues such as event counts and proportions. The chapter introduces a two-stage framework for semi-weak label learning, where the first stage estimates class counts, and the second stage assigns instance labels based on these counts. This method improves the granularity of weak labels with minimal additional annotation, leading to improved classifier performance.

Together, these chapters provide a comprehensive view of the techniques and strategies developed to train models effectively using weak, noisy, and semi-weak labels.

They establish the theoretical and practical foundations for advancing sound event detection in scenarios where precise labeling is not feasible, paving the way for more robust and scalable audio classification systems.

4

Learning from weak labels

Traditional training methods for training sound-event classifiers require *strongly* labeled data – audio recordings where each instance of the target sound event is marked, along with its temporal boundaries. Such data being hard to obtain, we must *weaken* the requirements of labeling. In this context, the weakest form of labeling is one where only the *presence or absence* of a sound event in a recording is marked, with no further indication of the number of occurrences of the recording or their locations in the recording. Approaches that can learn classifiers from this lattermost class of label are what we refer to as "weak label" approaches.

The development of weak labeling approaches for audio event detection (AED) has opened up the possibility of large-scale AEDs. However, a deeper understanding of how weak labeling affects the learning of sound events is still missing from the literature.

In this work, using a CNN-based approach for weakly supervised training of audio events, we study the effect of factors that naturally arise in weakly supervised learning of sound events on the generalization of the models. Specifically, we empirically analyze how two important factors, namely, *label density* and *label corruption*, affect the learning process in weakly supervised training of audio events.

4.1 Background - Importance of Weak labels

The problem of audio event detection (AED) deals with automatically analyzing the *audio content* of multimedia data to identify different sound events. The development of

large-scale AED techniques has been slow due to the lack of availability of large datasets and the difficulty in annotating data for sound events. Many datasets [85, 86, 87, 88] released in the last decade were small-scale both in terms of amount of training data and vocabulary of sound events. Supervised learning methods that rely on *strongly labeled* exemplars for training or time-stamped information to extract event exemplars are not expected to be a scalable approach to AED, as large amounts of strongly labeled data are very hard to come by. In fact, large-scale AED has been possible primarily due to techniques that learn from weakly labeled data [89]. Following [89, 20], several works on weakly supervised learning have been proposed, and this is now the primary mechanism to train audio classifiers.

It should be noted that all of these works use manually created weakly labeled datasets such as Audioset [16], which is a weakly labeled sound event dataset that, to the best of our knowledge, is currently the largest dataset for sounds. As a reminder, a primary motivation behind our work on weak labeling is to scale AED by exploiting the web data without manual labeling effort, e.g., by using metadata associated with web videos to automatically tag them with sound event labels. Consumer-generated web videos are recorded under unstructured conditions; occurrences of sound events could be corrupted by noise/other events, leading to huge intra-class variations, making the problem more challenging. This signal noise is, however, not the focus of this work.

In this chapter, we take a closer look at AED using weakly labeled data by analyzing two factors: *label density* and *label noise*, both of which are generally expected to induce noise in labels during the learning process.

Label density measures what percentage of a weakly labeled recording actually contains the labeled event. *Label noise* refers to incorrectly labeling audio events. It can occur due to several reasons. In the case of manual labeling, sound events are sometimes hard to interpret,, leading to incorrect labeling. However, label noise becomes a bigger problem when we attempt to work with labels obtained through automated methods, and it is essential to understand how this might affect the learning process. So far, there is little work on understanding label noise for sound events, although there is some literature on this related to vision tasks [90][91][92].

In [93], the authors have used *label noise* in weakly supervised learning and have proposed a unified learning framework that learns simultaneously from strongly and weakly labeled data. They proposed a graph-based semi-supervised learning method, which can be hard to scale for large amounts of audio data. In[94], the authors propose methods for transient sounds that can be thought of as being related to label density to a certain extent.

In the sections below, we quantify the effect of label noise and label density and

study how these affect learning from weak labels. We first describe a weak label learning approach that achieves state-of-the-art results on Audioset. Later, we also describe how we build and analyze a weakly labeled dataset in a fully automated fashion to aid AED.

4.2 CNN for Weakly labeled audio

In our work, we use the CNN-based AED model based on the one described in [95]. The CNNs in the model are designed for $X \in R^{n \times m}$ input, where n (row) is the time dimension and m (column) is some multidimensional feature vector. n depends on the duration of the recording, and m on the type of feature representation used to represent the data.

We use two different input features in our work: logMel spectrograms and the embeddings obtained from Google for the Audioset dataset [96].

The logMel spectrogram of an audio recording is represented by $X \in R^{n \times m}$. In this case, n is the number of log Mel frames, and m is the number of Mel filters used in the representation. In this work, we $m = 128$ Mel-filters. We use the term "frame" to refer to the i^{th} 128-dimensional time frame, and "segment" to represent a small segment of the whole audio recording. In terms of logMel features, a segment is a collection of k continuous frames ($k \times 128$ matrix). For example, $k = 128$ represents a 1.5 second segment of the audio recording.

The Google embeddings for Audioset comprise a 128-dimensional feature vector for each 1-second segment of the audio recording. This leads to a $R \in n \times 128$ dimensional representation for any audio, where n is the duration of the audio recording in seconds.

Given a collection of weakly labeled recordings (X_i, y_i), the goal is to design a network architecture that can learn from this weakly labeled dataset. We would expect the network to factor in the fact that the labels are *weak*.

Our network is inspired by [95]. We refer to our weakly labeled network as WalNet. The WalNet architecture is shown in Figure 4.5 for both logMel and Google-embedding-based input features. The networks for both features consist of convolutional blocks, which first predict the posteriors for each class at the segment level. These segment-level outputs are mapped by a mapping function to produce recording-level outputs. Our mapping function is an $average()$ function. Thus, the recording level output for any class is the average of the segment-level outputs. Using the recording level outputs, we compute the loss with reference to the target labels and train the network using backpropagation.

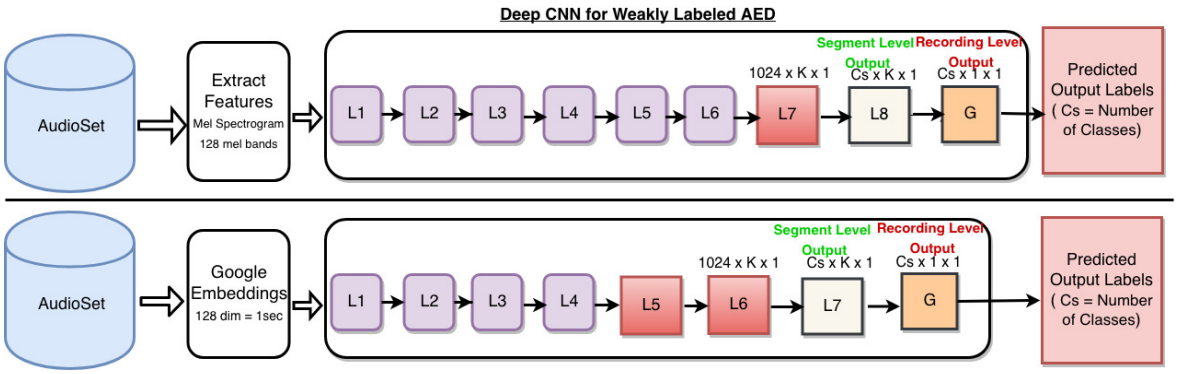


Figure 4.1: WalNet CNNs for Audio Events. The upper network is the CNN for logMel features. The network shown in the lower portion is for Google embedding features. The network provides an output at two levels of granularity - segment level output and recording level output

4.2.1 Characteristics of WalNet

Our network design embodies the different aspects of weakly labeled learning. It is designed to predict posteriors at the recording level by first predicting class posteriors at the segment level. The principle applied here is that we scan through the recording and predict outputs at every small segment. Once we know what happened at the segment level, we can use it to assign what will occur at the recording level. Thus, the network does not make any assumptions about labeling within the recordings.

Moreover, the network is fully convolutional, allowing it to handle variable length recordings. The network design controls the size of the segments over which posteriors are predicted. The network design can also control the amount by segments that are shifted.

In the specific case of LogMel WLAT in Fig 4.5, the segment size is 128 frames ($\sim$ 1.5 seconds), and segments are shifted by 64 frames ($\sim$ 0.75 seconds). Hence, consecutive segments overlap by 50%. The number of segments obtained after layer L8 depends on the duration of the audio recording. For example, an audio recording consisting of 864 logmel frames ($\mathcal{R} \in R^{864\times128}$) will produce $K = 12$ segments at L8. These segment-level outputs also be used for the temporal localization of sound events. Hence, the network can localize events despite learning from weakly labeled data.

4.3 Label Noise, Label Density, and Corresponding Experimental Designs

4.3.1 Labels Density

Label density indicates how much of a given audio recording contains the tagged event. We formally define label density (LD) of a recording R, with respect to an audio event

e as

$$LD_R^e = \frac{\mathrm{D}}{\mathrm{R}} \tag{4.1}$$

Where D is the duration (any time unit) for which event e is present in recording R, and L is the length of R. Correspondingly, label density noise (LDN) is $LDN_R^e = 1 - LD_R^e$. Label density noise is a measure of the *weakness* of the labels of a recording with respect to a given event. For example, an audio recording where a sound event is present for one-fourth of the duration of the recording is a weaker representation of the event than an audio recording where the event is present for three-quarters of the duration.

Understanding how label density affects the learning process in a weakly supervised learning framework is important. However, designing an evaluation strategy to measure the impact of label density on the generalization capabilities of trained models is not straightforward. Any such method would require one to measure the label density of all recordings with respect to each event. This would require us to manually obtain the duration for which each given event is present in the audio recording. Clearly, this cannot be done for a large-scale dataset such as *Audioset*. Hence, we consider an alternative view on label density, allowing us to study its impact on weakly labeled AED empirically. This alternate view is based on the intuition that if we mine weakly labeled audio from the web, the expected density of labels for a given audio event will be lower for long-duration audio recordings. In other words, on average, long-duration audio recordings will have higher "label density noise" than shorter audio recordings.

Taking this alternate view, we designed our experiment as follows:

Audioset has weak labels for YouTube audio recordings. On this dataset, weak labeling was manually done on audio recordings that were of exactly 10 seconds duration These 10-second audio segments were actually obtained from longer YouTube videos.

Let R_i be the i^{th} recording, YT_{id}^i be its source YouTube recording, S^i and E^i be the start and end times of R_i in YT_{id}^i, and L^i the list of events marked present in R_i.

Considering that the R_i are predominantly 10 seconds long, a high label density is anticipated for these recordings. By extracting segments ranging from $S^i - 10$ seconds to $E^i + 10$ seconds, and $S^i - 25$ seconds to $E^i + 25$ seconds from each YT_{id}^i, we generate recordings that exhibit successively lower label densities and concurrently, higher label noise. The weakly labelled sets thus obtained are denoted as *Audioset-At-30* and *Audioset-At-60* for the former and latter cases respectively. This procedure is performed for each tuple $(YT_{id}^i, S^i, E^i, L^i)$.

WalNet, our proposed model, is trained on these three sets to examine the effect of label density on its performance. Given the design of our network, which accommodates audio recordings of variable lengths, modifications in the length of the recordings do not constitute an obstacle.

4.3.2 Corrupted Labels and Noise

Moreover, we perform the corruption so that the number of recordings marked to contain the recording remains consistent with respect to the original labels. We perform an analysis of the different values of r. The set of corrupted labels for each r will be released for future works.

Labels that are incorrect or corrupted present a significant challenge. This issue persists even in meticulously labeled datasets such as Audioset, where the authors have acknowledged that the labels are imperfect, with numerous instances of false positive labels[1].

The quality of labels may significantly deteriorate when we endeavor to mine audio directly from the web and automate the weak labeling process, employing metadata associated with the audio (videos) to assign labels. The process could potentially result in both false positive and false negative label assignments. This could lead to marking an event as present in the recording when it is not, or inversely, failing to identify an event and incorrectly marking the recording as devoid of the given sound event. We conduct an analysis of this form of label noise by intentionally corrupting the labels in Audioset.

We regard the labels from AudioSet as flawless or having 0% corruption, given that Audioset is manually labeled, which may be assumed to be the optimal achievable quality. We subsequently incrementally introduce label noise by manually corrupting the assigned labels in Audioset. In specific terms, for a subset of the labeled tuples $(YT^i_{id}, S^i, E^i, L^i)$, we corrupt L^i by modifying the events in L^i. This corruption process is executed in a stratified manner, such that approximately r% labels for each event are corrupted. This r% encompasses both false positive and false negative label noises. Additionally, we orchestrate the corruption such that the quantity of recordings marked as containing the event remains consistent with the original labels. We conduct an analysis for different values of r.

4.3.3 Weakly Labeled Audio In the Wild

In order to deepen our understanding of weakly labeled learning for sound events, particularly in the context of automated labeling and web data, we directly procure audio recordings from YouTube associated with a specific sound event and use these recordings to train our model.

We employ a straightforward yet efficient filtering method to acquire recordings for a specific sound event. We retrieve videos by using search queries of the form "<sound event name> sound". The addition of the word "sound" significantly enhances the

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

retrieval of relevant videos on YouTube. We consider the top 50 retrieved videos (each under 4 minutes in duration) for each event and label these as containing the event. If a video is retrieved for multiple events, it is labeled accordingly for each event. We refer to this training set as *YouTube-wild*. In *YouTube-wild*, all forms of label noise can occur, and label density can range from 0 to 1. Since the retrieval process is imperfect, labeling all top 50 videos containing the event results in false positive labeling. This implies that we assign a positive label to the recording even when the event is not present. In the process, false negative labels are also naturally introduced. For example, a video V retrieved for an event e_1 might also contain another event e_2, but unless V is retrieved for e_2 as well, we are unaware of this and mark e_2 as not present in V.

The *YouTube-wild* dataset was created for a subset of 40 sound events from Audioset and comprises approximately 1,900 recordings. Like Audioset, *YouTube-wild* is multi-labeled. The selection of sound events for *YouTube-wild* was based on several factors. Please see the appendix of this chapter for more details.

4.4 Experiments and Results

4.4.1 Dataset

The AudioSet dataset [16] comes pre-divided into three subsets: *balanced training*, *Eval* and *Unbalanced*. We utilize the *balanced training* set as our training dataset for our experiments. This balanced training set contains approximately 21,500 audio recordings. The actual balanced trainingthe set available from Audioset is slightly larger due to a few videos being unavailable for download. Most of the recordings are 10 seconds in duration, though a few are less than 10 seconds. The total training data amounts to around 60 hours. For our experiments, we use the *Eval* set from Audioset as our test set, which consists of 19,789 recordings. Both the training and test sets contain at least 59 examples per class. From the *Unbalanced* set of Audioset, we extract a collection of 12,676 recordings to use as a validation set in our experiments. This validation set contains at least 30 examples per class. Audioset is a multi-label dataset with an average of 2.7 labels per recording.

4.4.2 Acoustic Features, Implementation and Evaluation Metrics

As mentioned earlier, we use Logmel spectrograms and Google embeddings as our features. For logmel spectrograms, all audio recordings are sampled at a 44.1 KHz sampling rate. A window of $\sim$ 23 ms (1024 points) and window hop size of $\sim$ 11.5 ms is used. The number of mel bands is set to 128. Hence, each frame of $X \in R^{k \times 128}$ corresponds to the logmel representation of a 32 ms window. WAL-Net is designed to produce outputs for a 128-frame segment moving by 64 frames. This corresponds to a

Table 4.1: Mean Average Precision on Audioset - 10-second recordings

Model	**AP**
ConvNet (mean pooling)	20.3
ResNet (mean pooling)	21.8
ResNet-ATT [Xu et al., 2017a]	22.0
ResNet-SPDA [Zhang et al., 2016]	21.9
M&mnet [Chou et al., 2018]	22.6
WAL Net	**22.9**

segment size of $\sim$ 1.5 seconds and a hop size of $\sim$ 0.75 seconds.

For Google embeddings, we followed the procedure described in [96] and further outlined here. [2]

Hyper-parameters are tuned using the validation set. Adam optimization [97] is used to train the networks and model selection (across different epochs), and parameter tuning is done using the validation set. We used average precision (AP) [98] and area under ROC curves for each event (AUC) [99] as evaluation metrics. The mean average precision (MAP) and mean area under ROC curves (MAUC) over all events are used as overall evaluation metrics.

4.4.3 Audioset Performance

Table 4.1 presents the Mean Average Precision (MAP) on the evaluation set across all 527 sound events. To allow for a comparison with the current state-of-the-art, we also include the reported figures from [100, 94]. As of the time of this work, WalNet is capable of achieving state-of-the-art performance by using the balanced training set from Audioset. An ensemble of M&mnet, as presented in [94], demonstrates slightly superior performance with a MAP of 23.2. We anticipate that an ensemble of WalNet models could yield comparable improvements.

4.4.4 Analysis of Label Density

The experimental setup for this portion adheres to the procedure outlined in Section 4.3.1. For the Audioset training set, we created two additional sets, designated *Audioset-At-30* and *Audioset-At-60*. These two sets, which contain approximately 180 and 360 hours of audio data respectively, have higher label density than the original Audioset. We train WalNet using the original Audioset, Audioset-At-30, and Audioset-At-60. The validation and evaluation sets remain identical to those in the original Audioset. The original Audioset provides label density at 10-second intervals, so we are analyzing the performance of models trained on datasets with weak labels available at 10-, 30-, and 60-second intervals.

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

Table 4.2: Effect of label density on performance using AudioSet

Training Set	**Feature**	**MAP**	**MAUC**
Audioset	Log Mel Spec	21.3	0.923
Audioset-At-30	Log Mel Spec	21.8	0.928
Audioset-At-60	Log Mel Spec	21.6	0.923
Audioset	Google Embedding [96]	22.9	0.919
Audioset-At-30	Google Embedding	22.4	0.905
Audioset-At-60	Google Embedding	22.4	0.908

Table 4.2 illustrates the impact of changes in label density on MAP and Mean Area Under the Curve (MAUC) values. For logmel spectrograms, there is a minor performance improvement when transitioning from Audioset to Audioset-At-30. However, a slight decline in performance is observed when further transitioning to Audioset-At-60. Overall, it could be contended that WalNet's performance is not significantly affected by label density.

This intuitively suggests that WalNet exhibits robustness to increases in label density, a desirable trait in any weak label learning framework. The enhancement observed with increasing audio lengths may be due to the corresponding increase in total training data. However, it can be inferred that extended-duration recordings, assuming they are available in large quantities, can be effectively utilized for training an architecture like WalNet, given the minimal impact of label density.

4.4.5 Analysis of Labels Corruption

The analysis of label noise, specifically in the context of corrupted labels, adheres to the procedure detailed in 4.3.2. We introduce artificially induced corruption in labels at 9 different rates, denoted by r. Consequently, we obtain 9 distinct training sets. A WAL-Net model is trained on each of these training sets, utilizing Google embeddings as input. The validation and evaluation sets are maintained as before. Fig. 4.2 illustrates MAP values for different rates of corruption, r. The case where $r = 0$ corresponds to the original training set from Audioset.

The model demonstrates reasonable robustness to additional noise up until $r = 15$. A 15% noise in labels results in only about a 5.9% relative decrease in performance. This showcases WAL-Net's resilience to label noise up to a certain extent. However, at extremely high corruption levels, specifically $r = 40$ and $r = 50$, a significant decrease in performance is observed as expected. The MAP at these corruption rates are $17.40(-24\%)$ and $16.81(-26.4\%)$ respectively.

From this, it can be concluded that corrupted labels play a significant role, and depending on the extent of label noise, the performance degradation for a state-of-the-art

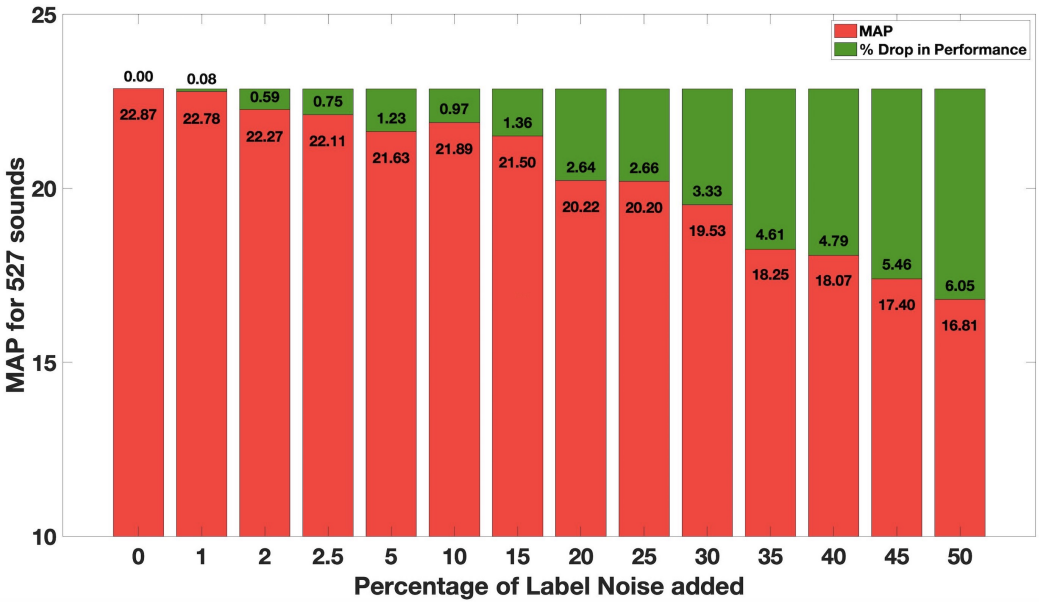


Figure 4.2: Effect of corruption of labels on performance. The X-axis represents corruption level r in %. The Y-axis shows MAP (Mean Average Precision) and % reduction in MAP compared to no corruption.

Table 4.3: Weakly Labeled Audio in the Wild. Comparison with manually labeled Audioset

Training Set	**MAP**	**MAUC**
Audioset-40	54.3	0.938
YouTube-Wild	38.0	0.872

method can range from moderate to severe. Although a deep learning model like WalNet demonstrates a degree of robustness, label noise remains a significant challenge for future works on Acoustic Event Detection (AED) utilizing weak labels, especially those relying on YouTube data. Such work should account for the likelihood of corrupted labels and model accordingly. As previously noted, even a manually labeled dataset like AudioSet is not immune to label noise, and it can be anticipated that any other large-scale weakly labeled dataset will encounter the same issue. Therefore, training algorithms should be designed considering the presence of label noise in datasets.

4.4.6 Weakly Labeled Audio in the Wild

We now present the performance results for YouTube-wild. For comparison purposes, we trained WalNet on the subset of Audioset training recordings that correspond to the 40 events included in the YouTube-wild vocabulary. This subset is referred to as Audioset-40 and comprises approximately 4,650 recordings. The objective is to draw a comparison between YouTube-wild and a manually labeled set. The list of events

and the YouTube-wild dataset will be made publicly available on our GitHub page. Evaluation is once again conducted on the recordings from the Eval set of Audioset, with results presented for Google embedding features.

We note a significant performance disparity between Audioset-40 and YouTube-wild. Specifically, compared to the manually labeled and scrutinized Audioset-40, YouTube-wild lags behind by an absolute 16.3% or 30% relative in terms of MAP. This outcome underscores the importance of careful examination of how weak labels function for sound event identification and the need to develop more robust learning methodologies.

4.5 Conclusion

In this chapter, our objective was to delve into the challenges associated with large-scale AED using weakly labeled data, thereby providing a foundation for future research in this field. We introduced a CNN-based framework designed to learn from weakly labeled audio recordings and demonstrated its state-of-the-art performance on AudioSet. Subsequently, we outlined various ways in which label noise might appear in weakly labeled data. We proposed experimental designs aimed at examining the effects of *label density* and *labels corruption* on performance. As we highlighted in this study, label density is an inherent aspect of the weak label learning paradigm, but its effects on performance are minimal. Conversely, label noise can have a moderate to severe impact on performance.

In addition, we explored the potential of directly mining weakly labeled data from the web and compared it with AudioSet, which consists of manually labeled short audio recordings. Our findings indicate a significant need for further work in this area, given the considerable performance gap.

4.6 Appendix

4.6.1 CNN Architecture

Let's consider each recording and its weak labels as pairs $(\mathcal{R}_i, Y_i)$, where $\mathcal{R}_i$ is the i^{th} recording and Y_i represents its corresponding label. The label vector, Y_i, is a C-dimensional vector with C denoting the total number of classes. Each index j of this vector is assigned a value of 1 if the C_j^{th} sound event is present, otherwise 0. Our goal is to train a network using this weakly labeled dataset, acknowledging that the labels are *weak*. This means that even though we have a label for the entire recording, the labeled event may not necessarily persist throughout the recording, which complicates loss calculation and network training.

Let's represent the segments of the recording $\mathcal{R}_i$ as $S_k^{\mathcal{R}_i}$, where $k = 1, 2, ...K$. Here,

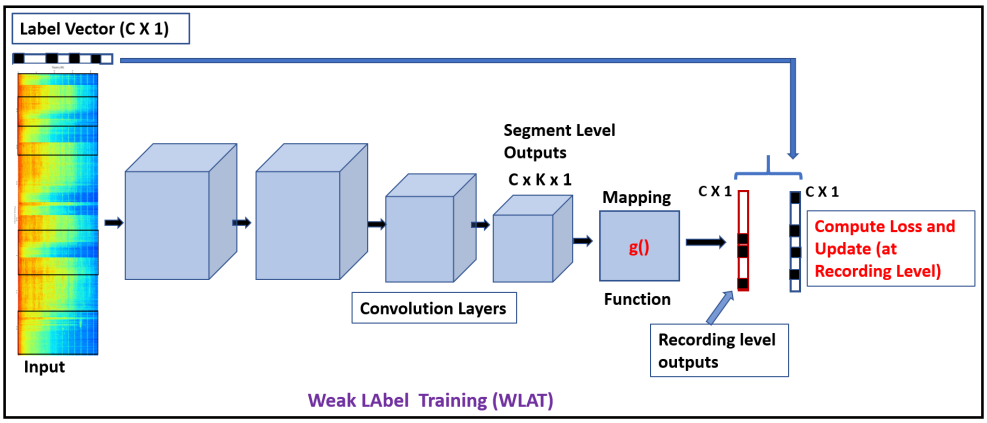


Figure 4.3: General Schema of Weak Label Training. The network is designed to produce segment level outputs first and these segment level outputs are then mapped by the mapping function $g()$ to obtain recording level output. The loss computation and updates can then be done using these segment level outputs.

K indicates the total number of possible segments for the recording $\mathcal{R}_i$ considering a certain segment and hop size. We'll assume that $f(S_k^{\mathcal{R}_i})$, where $k = 1, 2, ...K$, are the network outputs for the recording segments. Each $f(S_k^{\mathcal{R}_i})$ is a C-dimensional vector. Our method of weak label training chiefly involves mapping these segment-level outputs to the recording-level, thereby generating a single vector that represents the class-wise output for the entire recording. Once we have the recording-level outputs for all classes, we can compute the loss function with the recording-level label Y_i and train the network. This concept is formalized in Eq. 4.2.

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

In Eq. 4.2, the function $g()$ is used to map the segment-level predictions ($f(S_i^{\mathcal{R}})$) to the recording-level prediction. The loss ($\mathcal{L}$ loss function) is then computed by comparing this recording-level prediction with the recording-level label, Y_i. The mapping function $g()$ can be any function that inspects segment-level predictions and uses this information to generate an output for the entire recording. This formulation encapsulates the notion that *weak labels*, which are labels for the entire recordings, originate from the presence or absence of events at lower levels, such as at the segment level. The purpose of the mapping function is to analyze these segment-level outputs and generate the recording-level output based on them. The general structure for Weakly LAbeled Training (WLAT, pronounced "dub-lat") is depicted in Figure 4.3.

4.6.2 Multi-Label Training for WAL-Net

An audio recording can have multiple classes present in it. The weakly labeled *Audioset*, on an average consists of 2.7 labels per recording. Hence, the training procedure needs to consider presence of multiple labels in the recording. The output of WLAT gives class specific posteriors for any given input. The binary cross entropy loss with respect to each class is then given by Eq. 4.3.

$$l(y_c, p_c) = -y_c * log(p_c) - (1 - y_c) * log(1 - p_c) \tag{4.3}$$

y_c is the target output for c^{th} class, 1 if c^{th} event is marked to be present and 0 otherwise. p_c = WLAT(X) is the network output for the c^{th} class. The training loss is the mean of losses over all classes

$$L(X, y) = \frac{1}{C} * \Sigma_{c=1}^{C_s} \; l(y_c, p_c). \tag{4.4}$$

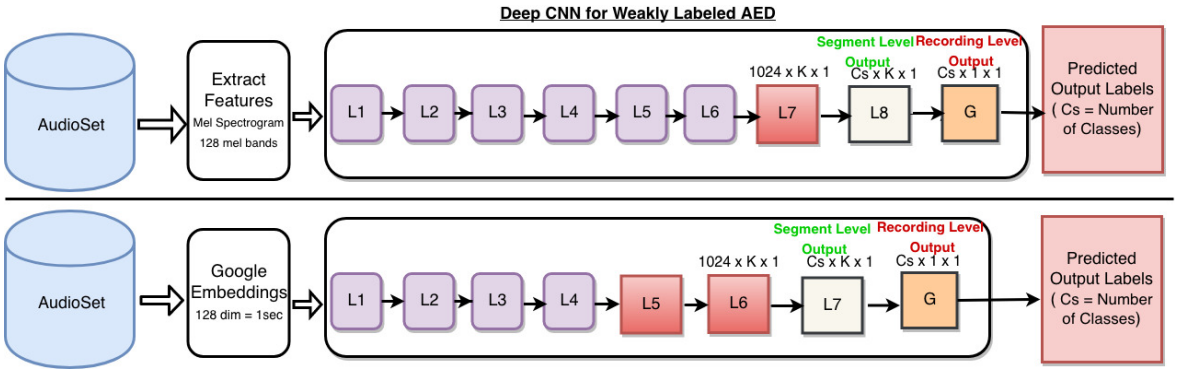


Figure 4.4: WALNet for Audio Events. The upper network is CNN for logmel features. The network shown in the lower portion is for Google embedding features.

WALNet CNN for Logmel: As shown in the top half of Figure 4.5, we have a CNN network for logmel features. Any given audio recording is represented by $\mathcal{R} \in R^{m \times 128}$, considering we use 128 mel-bands in logmel representations and m stands for the number of logmel frames for $\mathcal{R}$.

The blocks of layers from L1 to L5 are composed of two convolutional layers, followed by a max pooling layer. Prior to the non-linear activation function, each convolutional layer has batch normalization. The ReLU function ($max(0, x)$) is used in all layers from L1 to L6, and 3×3 convolutional filters are used throughout. The stride and padding values are consistently set to 1. The number of filters used in different layers is as follows: *L1: 32, L2:64, L3:128, L4:128, L5:256, L6:512* . Note that L1 to L5 consist of two convolutional layers followed by max pooling, while L6 only has one convolutional layer followed by a max pooling layer. Both convolutional layers in these blocks use the

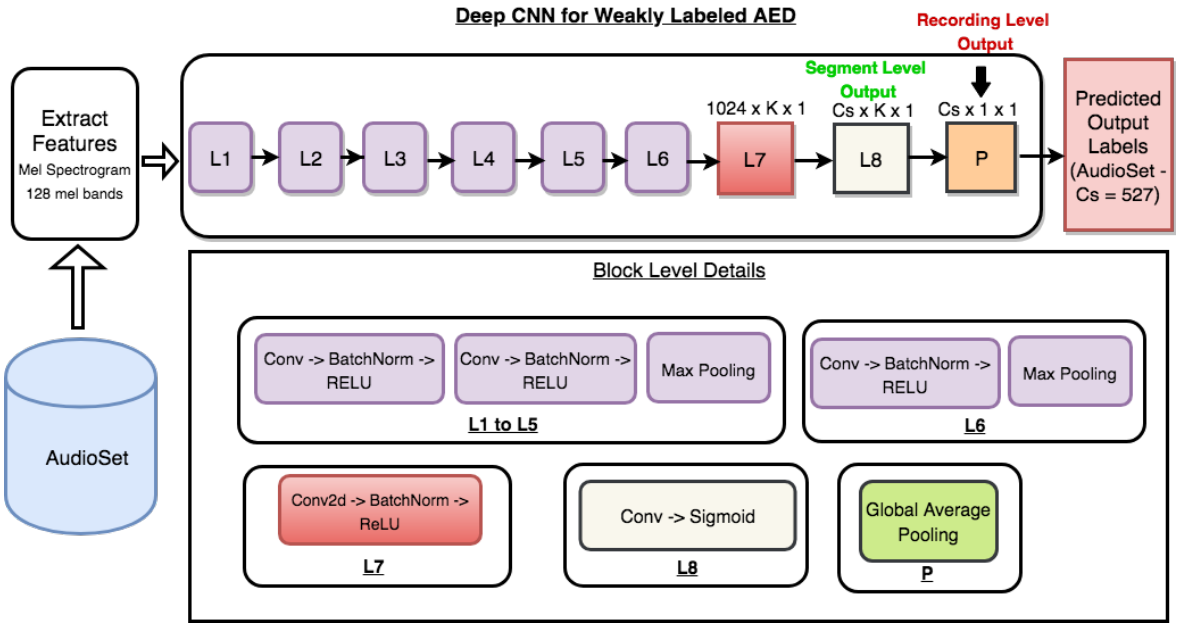


Figure 4.5: WALNet CNN architecture block level details for log mel features.

same number of filters. The max pooling operation is applied over a 2×2 window, and logmel inputs are treated as single channel inputs.

Layer L7 is a convolutional layer with 1024 filters of size 2×2 and uses ReLU activation. The stride is set to 1 and no padding is used. Layer L8 produces segment level output and is a convolutional layer consisting of C_s filters of size 1×1, where C_s is the number of classes in the dataset. This layer uses a sigmoid non-linear activation function. The mapping function $g()$ averages the segment level outputs for each class.

WALNet for Google Embeddings: Google provides a VGG-like network to compute segment level embedding for any recording. This network, trained on a substantial amount of data (over 5 million hours) through strong label assumptions, is expected to yield discriminative representations for audio recordings. The embeddings are 128-dimensional quantized vectors, with the details of the embeddings available in [101][3]. Each 1-second of audio recording is assigned an embedding, so a 10-second audio recording would result in a 10×128 matrix representation. Thus, each audio recording is represented by $\mathcal{R} \in R^{N \times 128}$, with N being the number of embedding vectors for the recording.

The lower half of Figure 4.5 displays the WLAT CNN architecture for these embedding features. The blocks of layers from L1 to L4 are composed of two convolutional layers, followed by a max pooling layer. Each convolutional layer uses batch normal-

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

ization followed by ReLU activation. The filter or kernel size, stride, and padding are again 3×3, 1, and 1 respectively. The number of filters used in these blocks is as follows: *L1: 64, L2:128, L3:256, L4:512* . Both convolutional layers in these blocks use the same number of filters. The max pooling operation is applied over a 1×2 window, which moves with the same stride. Layer L5 is a convolutional layer with 1024 filters of size 1×8, and ReLU activation is used again. The stride is set to 1 and no padding is used. Layer L6 is another convolutional layer, this time with 1024 filters of size 1×1. Both of these layers apply batch normalization before the non-linear activations. Layer L7 is the segment level output layer, and is a convolutional layer with C filters of size 1×1 and uses a sigmoid non-linearity. The mapping function $g()$ averages the outputs again.

4.7 Datasets

4.7.1 Audioset

AudioSet is classified as a *large-scale weakly labelled* dataset. It includes weak labels that have been manually annotated. Each entry in the data set consists of a recording of 10 seconds or less, and these weak labels signify the existence of events within these short clips. This dataset is multi-label, with each recording having, on average, 2.7 labels. It comes pre-partitioned into three sets. We utilize the *Balanced* set as our training data. At the time of our acquisition, this set comprised approximately 21,500 recordings, which is about 60 hours of audio data. The actual balanced training set offered by AudioSet is slightly larger, but some videos were unavailable for download at the time of our retrieval. The *Eval* set of AudioSet is employed as our test or evaluation set, which contains about 20,000 recordings in total. We use a subset of around 13,000 recordings from the *unbalanced* set as our validation set, ensuring at least 30 examples for each class.

The AudioSet training data can be represented by tuples in the form of $(R^i, YT^i_{id}, S^i, E^i, L^i)$, where R^i is the i^{th} recording in AudioSet and YT^i_{id} is the corresponding YouTube id of R^i. S^i indicates the start time of R^i in YT^i_{id}, while E^i denotes the end time. Hence, R^i is a segment of the video YT^i_{id} on YouTube, starting at S^i and concluding at E^i. The recordings R^i are weakly labeled (manually), and L^i symbolizes the set of audio events found in R^i. Consequently, this labeling implies that the set of events in L^i are present in YT^i_{id} between S^i and E^i. For a deeper understanding, we refer readers to the label sets provided by AudioSet[4].

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

Table 4.4: 10 Events with highest relative drop in performance

Events	**AudioSet-10**	**AudioSet-10 at 45 % label corruption**	**AP Drop (% Drop)**
Glass	0.080, (0.925)	0.005, (0.476)	0.074, (93.35)
Neigh, whinny	0.165, (0.945)	0.012, (0.692)	0.152, (92.59)
Owl	0.144, (0.929)	0.011, (0.564)	0.134, (92.51)
Knock	0.110, (0.947)	0.008, (0.529)	0.102, (92.43)
Tick-tock	0.109, (0.949)	0.009, (0.735)	0.100, (91.95)
Mouse	0.052, (0.934)	0.004, (0.697)	0.0478, (90.94)
Single lens reflex camera	0.052, (0.935)	0.005, (0.621)	0.047, (90.44)
Breaking	0.142, (0.973)	0.017 (0.782)	0.125, (88.06)
Tap	0.07, (0.)	0.008, (0.833)	0.068, (87.82)
Burping,eructation	0.090, (0.956)	0.014, (0.771)	0.076, (84.13)
Mean	**0.101, (0.947)**	**0.014, (0.670)**	**0.087, (86.14)**

4.7.2 YouTube-Wild

Vocabulary Selection: Numerous audio events in *AudioSet* represent a vast range of meanings. The automated labelling process outlined above can yield insignificant results for these events. Therefore, we chose to work with a smaller subset of events, specifically, those with a more definitive meaning - that is, those that, when retrieved, produce a relatively meaningful and pertinent set of audio recordings. The selection process also considers the total number of examples available for the event in AudioSet. We select events with a higher count of examples in AudioSet to analyze how *YouTube-Wild* compares to *AudioSet*, which is manually labeled. This choice is driven by the desire to better understand where *YouTube-Wild* stands in comparison to AudioSet. *YouTube-Wild* is a dataset that is collected and labelled without manual intervention and comprises lengthy audio recordings, unlike AudioSet, which is manually labelled, and the weak labels apply to relatively short 10-second recordings. Both datasets originate from YouTube, hence, the level of background noise is expected to be similar. The most notable differences, however, are likely to stem from the two forms of "label noise".

In this chapter, we have elucidated deep learning techniques for detecting audio events utilizing weakly labeled data. Our methodologies primarily depend on convolutional neural networks, subsequently integrating the weak label constraints into the learning procedure. The fundamental concept entails adopting a bottom-up strategy, where the recording-level prediction is conducted through segment-level predictions. Within the Weakly Labeled Audio Tagging (WALNeT) framework, the network is configured to produce segment-level outputs directly, followed by a mapping function that converts these segment-level outputs into recording-level outputs. The comprehensive framework is universally generic and can be implemented with various network architectures. We explored multiple mapping functions, some of which introduce attention-like behavior into the framework. Other types of mapping function can also be developed.

From the viewpoint of weakly labeled audio event detection, our methodology encapsulates several attractive qualities. It can manage recordings of differing lengths, and the network design governs the segment sizes over which segment-level outputs are created, eliminating any need for additional preprocessing steps. Our method can achieve state-of-the-art performance. Particularly compared to the Segment-Level Audio Tagging (SLAT) method, our WLAT approach performs superiorly and is also less computationally demanding, resulting in enhanced training and inference times.

Additionally, we delved into the intricacies of weak label learning. We specifically highlighted two factors likely to be encountered when learning from weakly labeled data. Label density noise is a problem intrinsic to the nature of weak labels, and corrupted labels are also anticipated to frequently occur. Considering that other deep learning techniques for weakly labeled datasets have been proposed [102], it would be intriguing to discern which methods are more robust against these label noises. It is crucial that future research endeavors address these issues when learning from weakly labeled data. [5]

[5] Code available at: https://github.com/cmu-mlsp/ankit-phd-thesis/tree/master/WALNet-Weak_Label_Analysis

5

Learning from noisy and web data

Although weakly labeled data provide a significant simplification of labeling requirements, compared to strongly labeled data, they still require significant manual effort in assigning the weak labels themselves. In this work, we introduce *webly labeled* learning for sound events, which aims to remove human supervision altogether from the learning process. We first develop a method of obtaining labeled audio data from the web (*albeit noisy*), in which no manual labeling is involved. We then describe methods to learn from these webly labeled audio recordings efficiently. In our proposed system, *WeblyNet*, two deep neural networks *co-teach* each other to robustly learn from webly labeled data, leading to around **17%** relative improvement over the baseline method. The method also involves transfer learning to obtain efficient representations.

5.1 Motivation: Why deal with noisy and web data

As discussed previously, the long-standing problem in audio event detection (AED) has been the availability of labeled data. Labeling sound events in an audio stream requires marking their beginnings and ends. Annotating audio recordings with times of occurrences of events is a laborious and resource-intensive task. Weakly-supervised learning for sound events [30] addressed this issue by showing that it is possible to train audio event detectors using *weakly labeled* data: audio recordings that are tagged only with the presence or absence of the events as opposed to the time stamp annotations in

strongly labeled audio data.

Being able to work with weak labels is, however, only half the story. Even weak labeling, when done manually, becomes challenging on a large scale; tagging a large number of audio recordings for a large number of sound classes is non-trivial. Datasets along the lines of AudioSet [101] are not easy to create and require considerable resources. However, a significant advantage of weakly labeled learning is that it opens up the possibility of learning from the data on the web *without employing manual annotation*, thereby allowing large-scale learning without laborious human supervision.

The web provides us with a rich resource from which weakly-labeled data could be easily derived. It removes the resource-intensive process of creating the training data manually and opens up the possibility of completely automated training. However, this brings up a new problem – the weak labels associated with these recordings, having been automatically obtained through some means, are likely to be noisy. The challenge now extends to being able to learn from *weakly* and *noisily* labeled web data. We call such data ***webly labeled***.

Several works have been on webly supervised learning of visual objects and concepts [103, 104, 105]. However, learning sound events from webly labeled data has received little to no attention. The main prior work here is [93], where webly labeled data have been employed; however, to counter the noise in the labels, strongly labeled data are used to provide additional supervision. Needless to say, the strong labels that act as supporting data are manually obtained.

This chapter discusses an approach to utilize the web data to learn sound events. Primarily, the goal is to eliminate human supervision altogether from the learning process by proposing webly supervised learning of sounds. Webly labeled data by default are weakly labeled and hence our proposed methods are designed for weakly labeled audio recordings. Our motivation then is to introduce a learning scheme that can effectively counter additional challenges of webly labeled data. We first present the challenges of webly labeled learning of sounds and then an outline of the proposed system in next section.

5.1.1 Challenges in Webly Labeled Learning

Webly labeled learning involves several challenges. The first one is obtaining the webly labeled data itself. The challenge of obtaining quality exemplars from the web has been well documented in several computer vision works [103, 106, 107, 105]. This applies to sound events as well and is, in fact, harder due to the complex and intricate ways we describe sounds [108]. Often, sound-related terms are not mentioned in videos, and hence, text-based retrieval can lead to a much inferior collection of exemplars.

Nevertheless, a collection of exemplars for sound events obtained through automated methods will contain incorrect exemplars or label noise. This is a major challenge from the learning perspective. Learning from noisy labels has been a known problem [109], and in recent years, effectively training deep learning models from noisy labels has also started to get attention. However, it remains an open problem. Even here, most of the work on learning from noisy labels has been in the domain of computer vision.

The next challenge is the presence of signal noise. Manual annotations keep in check the overall amount of signal noise in the data. Even if the source of data is the web (e.g. AudioSet), manual labeling ensures that the sound is at least audible to most human subjects. However, for webly labeled data, the signal noise is "unchecked," and the sound event, even if present, might be heavily masked by other sounds or noise. Signal noise in the webly labeled data is challenging to quantify and remains an open research topic for future works.

In this work, we develop an entire framework to deal with such webly labeled data. We begin by collecting a webly labeled dataset using a video search engine as the source. We then propose a deep learning-based system for effectively learning from this webly labeled data. Our primary idea is that two neural networks can co-teach each other to robustly learn from webly labeled data. The two networks use two different views of the data due to which they have different learning capabilities. Since the labels are noisy, we argue that one cannot rely only on the loss with respect to the labels to train the networks. Instead, the agreement between the networks can be used to address this problem. Hence, we introduce a method to factor the agreement between the networks in the learning process. Our system also includes transfer learning to obtain robust feature representations.

5.2 Webly Labeled Learning of Sounds

5.2.1 Webly Labeled Training Data

Obtaining training audio recordings is the first step in the learning process and is a considerable hard open problem on its own. The most popular approach in webly supervised systems in vision has been text-query-based retrieval from search engines [106, 107, 103]. Our approach is along similar lines where we use text queries to retrieve potential exemplars from a video search engine, YouTube.

We must first select a "vocabulary" of sounds – terms used to describe sounds. For this work, we use a subset of 40 sound events from AudioSet, chosen based on several factors. These include preciseness in event names and definitions, the quality of metadata-based retrieval of videos from YouTube, the retention of sound hierarchies,

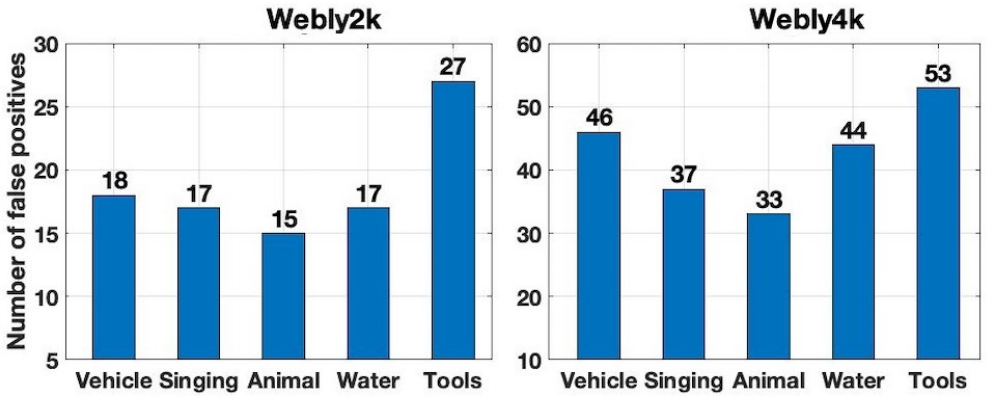


Figure 5.1: Top 5 sound classes False positive rate False positive False Positive for the 5 sound classes with the highest FP.

and the number of exemplars in AudioSet (larger is better). [1]

5.2.1.1 Obtaining Webly Labeled Data

Using only the sound name itself as a text query on YouTube leads to extremely noisy retrieval. [110] argued that humans often use the phrase "sound of" in texts before referring to a sound. Based on this intuition, we augment the search query with the phrase "sound of". This leads to a dramatic improvement in the retrieval of relevant examples. For example, using *"sound of dog"* instead of *"dog"* improves the relevant results (sound event actually present in the recordings) by more than 60% in the top 50 retrieved videos. Hence, we use the phrase "sound of <sound-class>" as the search query for retrieving example recordings of each class.

We formed two datasets using the above strategy. The first one, referred to as ***Webly-2k***, uses the top 50 retrieved videos for each class and has around 1,900 audio recordings. The second one, ***Webly-4k***, uses the top 100 retrieved videos for each class and contains around 3,800 recordings. Note that some recordings are retrieved for multiple classes; hence, the datasets are multi-labeled, similar to AudioSet. Only recordings under 4 minutes duration are considered.

5.2.1.2 Analysis of the Dataset

The average duration of the *Webly-2k* set is around 111 seconds, resulting in a total of around 60 hours of data. Webly-4k is around 108 hours of audio with an average recording duration of 101 seconds. As mentioned before, label noise is expected in these datasets. To analyze this, we manually verified the positive exemplars of each class and

[1] For more details visit github page mentioned in Section 5.3.

estimated the number of false positives (FP) for each class. As may be expected, the larger *Webly-4k* contains far more noisy labels than *Webly-2k*.

Figure 5.1 shows the FP counts for 5 classes with the highest false positives. Note that for these classes, 30-50% of the examples are wrongly labeled to contain the sound when it is actually not present. However, FP values can also be low for some classes, e.g., *Piano* and *Crowd*. Estimating false negatives requires one to manually check all of the recordings for all classes, which makes the task considerably tricky. Even the AudioSet dataset has not been assessed for false negatives (FN), and we also keep FN estimation out of the scope of this paper.

5.2.2 Our Approach: WeblyNet

The manual verification in the previous section was done only for analysis; the actual goal is to learn from noisily-labeled Webly-4k (or -2k) directly. Robustly training neural networks with noisy labels remains a very hard problem [111]. Several methods have been proposed, especially in the vision domain [112, 103, 104, 113, 111, 109]. These methods include bootstrapping, self-training, pseudo-labeling, and curriculum learning, to mention a few. Another set of approaches tries to estimate a noise transition matrix to estimate the distribution of noise in labels [112]. However, estimating the noise transition matrix is not an easy problem as it is dependent on the representation and input features. Conventionally, ensemble learning has also been helpful in handling noisy labels [109].

In supervised learning, neural networks are trained on some divergence measure between the output produced by the network and the ground truth label. As the network is trained, the noise in the labels will lead to wrong updates in parameters, which can affect the generalization capabilities of the network [114]. Some recent approaches have used the idea of having two networks working together to address this problem [115, 116]. [115] gives a "when to update rule" where networks are updated when they disagree. In [116], networks co-teach each other by sampling instances for training within a minibatch.

Our approach is fundamentally based on the idea of training multiple networks together, where the agreements (or disagreements) between the networks are used for improved learning. The method incorporates ideas from co-training and multi-view learning. Multi-view learning methods (e.g., co-training [117][118]) are primarily semi-supervised learning methods where learners are trained on different views of the data. The goal is to maximize their agreement on the unlabeled data. Our proposed method exploits this central idea of the agreement between classifiers to address the challenges of webly labeled data. The intuition is as follows: Two (or more) independent classifiers

Algorithm 1 WeblyNet system

Input: Networks $\mathcal{N}_1$ to $\mathcal{N}_K$, Representation R_1 to R_K of audio recordings for different networks and labels Y of the recordings, learning rates η_1 to η_K, divergence weights α_1 to $\alpha_{K(K-1)/2}$, number of epochs n_{epochs}

Output: Jointly trained networks

for $n = 1, 2, ...n_{epochs}$ **do**
  for $k = 1, 2, ...K$ **do**
    Compute loss, $\mathcal{L}_k(\mathcal{N}_k, Y)$ w.r.t label Y for network $\mathcal{N}_k$
  end for
  for $k = 1, 2, ...K(K-1)/2$ **do**
    Compute divergence $D(\mathcal{N}_i(R_i), \mathcal{N}_j(R_j))$ between each pair of networks
    Weigh each divergence by its corresponding hyperparameter α
  end for
  Combine all loss terms $\mathcal{L}()$ and divergence terms $D()$
  Update networks based on the combined loss.
end for

operating on noisily labeled data are likely to agree with the provided label when it is correct. When the given label is *incorrect*, the classifiers will unlikely concur with it. However, they are likely to agree with *one another* if both independently identify the correct label. Hence, the networks can inform each other of the errors they are making and help in filtering out those that are coming from noisy labels, thereby improving the overall robustness of the networks.

In contrast to prior work such as [116], our proposed method explicitly ties in the co-teaching of the networks by having a disagreement measure in the loss term. Moreover, in our method, the two networks operate on different views of the data and hence have different learning abilities. As a result, they will not fall into a degenerate situation where both networks essentially end up learning the same thing. This allows us to combine the classifiers' outputs during the prediction phase, which further improves the performance. We refer to our overall system as WeblyNet.

Furthermore, our method is easily extended to more than two networks. The central idea remains the same: a divergence measure captures the disagreement measure between two given network pairs. Given K networks in the system, $K(K-1)/2$ pairs of disagreements can be measured. These disagreement measures along with losses with respect to the available ground truths are then used to update the network parameters. Each divergence measure can be appropriately weighed by a scalar α to reflect the weight given to that particular pair of networks. Algorithm 3 outlines this procedure.

In this work, we work with only two networks. Figure 5.2 shows an overview of the proposed method. The two networks, "Network 1" and "Network 2," take two

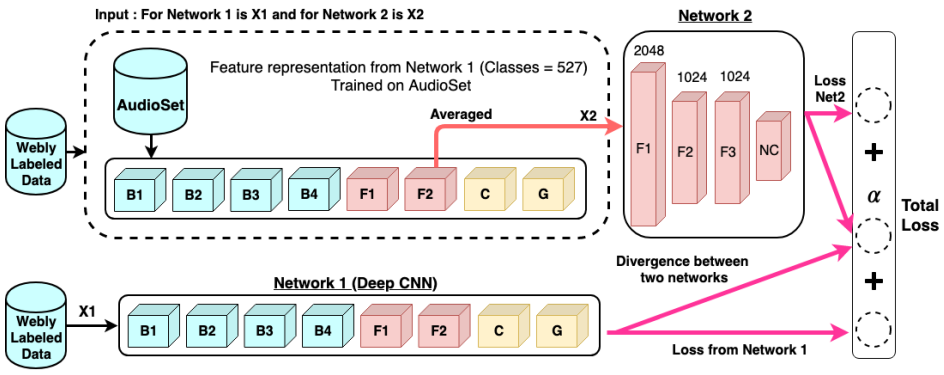


Figure 5.2: WeblyNet System: Network 1 (N1) is a deep CNN with first view of data as input. Network 2 (N2) takes in the second view of data obtained through transfer learning. The networks are trained together to co-teach each other.

different views of the data as input. The networks are trained jointly by combining their individual loss functions and a third divergence term, which explicitly measures the agreement between the two networks. The individual losses provide supervision from given labels, and the mutual agreement provides supervision when the labels are noisy.

5.2.2.1 Two Views of the Data

Our primary representation of audio recordings are embeddings provided by Google [96]. The embeddings are 128-dimensional quantized vectors for each 1 second of audio. Hence, an audio recording $\mathcal{R}$, in this first view, is represented by a feature matrix $X_1 \in \mathcal{R}^{NX128}$, where N depends on the duration of the audio. The temporal structure of the audio is maintained by stacking the embedding sequentially in X_1. The first network $\mathcal{N}_1$ is trained on these features.

Several methods exist in the literature for generating multiple views from a single view [118]. In this work, we propose to use various non-linear transforms through a neural network to generate the second view (X_2) of the data. To this end, we use a network trained on the first view X_1 to obtain the second view of the data. This network is trained on another large-scale sound events dataset (different from the webly labeled set we work with). One motivation is that, given the noisy nature of webly labeled data, a network trained on a large-scale dataset such as AudioSet can provide robust feature representations (as in transfer learning approaches [119]).

We first train a network ($\mathcal{N}_1$ with C = 527) on the AudioSet dataset and then use this trained model to obtain feature representations for our webly labeled data. More specifically, the F2 layer is used to obtain 1024-dimensional representations for the audio recordings by averaging the outputs across all 1-second segments. This representation

learning through knowledge transfer, as shown empirically later, is significantly useful in webly labeled data where a higher level of signal noise and intra-class variation is expected.

5.2.2.2 Network Architectures: $\mathcal{N}_1$ and $\mathcal{N}_2$

$\mathcal{N}_1$ is trained on the first (X_1) audio representations. It is a deep CNN. The layer blocks from B1 to B4 consists of two convolutional layers followed by a max-pooling layer. The number of filters in both convolutional layers of these blocks are, { B1:64, B2:128, B3:256, B4:256 }. The convolutional filters are of size 3×3 in all cases, and the convolution operation is done with a stride of 1. A padding of 1 is also applied to inputs of all convolutional layers. The max-pooling in these blocks are done using a window of size 1×2, moving by the same amount. Layer F1 and F2 are again convolutional layers with 1024 filters of size 1×8 and 1024 filters of size 1×1, respectively. All convolutional layers from B1 to F2 include batch-normalization [120] and ReLU ($max(0, x)$) activations. The layer represented as C is the segment-level output layer. It consists of C filters of size 1×1, where C is the number of classes in the dataset. This layer has a sigmoid activation function. The segment-level outputs are pooled through a mapping function in the layer marked as G to produce the recording-level output. We use the average function to perform this mapping.

The network architecture $\mathcal{N}_1$ achieves state-of-the-art results on AudioSet when trained with C = 527 (*i.e.* all 527 classes in AudioSet). Hence, we believe it is a good base network for our webly labeled learning.

The network $\mathcal{N}_2$ (with X_2 as inputs) consists of 3 fully connected hidden layers with 2048, 1024 and 1024 neurons respectively. The output layer contains a C number of neurons. A dropout of 0.4 is applied after the first and second hidden layers. ReLU activation is used in all hidden layers, and sigmoid is used in the output layer.

5.2.2.3 Training WeblyNet

Given the multi-label nature of datasets, we first compute the loss with respect to each class. The output layer of both $\mathcal{N}_1$ and $\mathcal{N}_2$ gives posterior outputs for each class. We use the binary cross-entropy loss, defined with respect to c^{th} class as, $l(y_c, p_c) = -y_c * log(p_c) - (1 - y_c) * log(1 - p_c)$. Here, y_c and $p_c = \mathcal{N}(X)$ are the target and the network output for c^{th} class, respectively. The overall loss function with respect to the target is the mean of losses over all classes, as shown in Eq 5.1

$$\mathcal{L}(\mathcal{N}(X), y) = \frac{1}{C} \sum_{c=1}^{C} l(y_c, p_c) \tag{5.1}$$

In the WeblyNet system, $\mathcal{N}_1$ and $\mathcal{N}_2$ co-teach each other through the following loss function

$$\begin{aligned}\mathcal{L}(X_1, X_2, y) = \mathcal{L}(\mathcal{N}_1(X_1), y) + \mathcal{L}(\mathcal{N}_2(X_2), y) + \\ \alpha \cdot D(\mathcal{N}_1(X_1), \mathcal{N}_2(X_2))\end{aligned} \tag{5.2}$$

The first two terms in Eq5.2 are losses for the two networks with respect to the target.

$D(\mathcal{N}_1(X_1), \mathcal{N}_2(X_2))$ is the divergence measure between the outputs of the two networks. The divergence of "opinion" between the networks provides additional information beyond the losses with respect to target labels and helps reduce the impact of noisy labels on the learning process. The α term in Eq 5.2 is a hyperparameter and controls the weight given to the divergence measure in the total loss. This can be set through a grid search and validation.

The divergence, $D(\mathcal{N}_1(X_1), \mathcal{N}_2(X_2))$, between the networks can be measured through a variety of functions. We found that the generalized KL-divergence worked best [121]. Note that the outputs from the two networks do not sum to 1. The generalized KL divergence is defined as $D_{KL}(\mathbf{x}||\mathbf{y}) = \sum_{i=1}^{d} x_i log(\frac{x_i}{y_i}) - \sum_{i=1}^{d} x_i + \sum_{i=1}^{d} y_i$. $D_{KL}(\mathbf{x}||\mathbf{y})$ is non-symmetric and is not a distance measure. We use $D(\mathbf{x}, \mathbf{y}) = D_{KL}(\mathbf{x}||\mathbf{y}) + D_{KL}(\mathbf{y}||\mathbf{x})$ to measure the divergence between the outputs of the two networks. This measure is symmetric with respect to $\mathbf{x}$ and $\mathbf{y}$. If $\mathbf{o}^{\mathcal{N}_1}$ and $\mathbf{o}^{\mathcal{N}_2}$ are outputs from $\mathcal{N}_1$ and $\mathcal{N}_2$ respectively then $D(\mathcal{N}_1(X_1), \mathcal{N}_2(X_2))$ is

$$D(\mathcal{N}_1(X_1), \mathcal{N}_2(X_2)) = \sum_{i=1}^{C} (\mathbf{o}_i^{\mathcal{N}_1} - \mathbf{o}_i^{\mathcal{N}_2}) log \frac{\mathbf{o}_i^{\mathcal{N}_1}}{\mathbf{o}_i^{\mathcal{N}_2}} \tag{5.3}$$

During the inference stage, the prediction from WeblyNet is the average of the outputs from the networks $\mathcal{N}_1$ and $\mathcal{N}_2$.

5.3 Experiments and Results

WeblyNet is trained on the Webly-4k and Webly-2k training sets. Audio recordings are represented through X_1 and X_2 views (Sec. 5.2.2). To the best of our knowledge, no prior work has done an extensive study of webly supervised learning for sound events. Previous work, [93], is computationally not scalable to over 100 hours of data we use in this work. Moreover, it relies on strongly labeled data in the learning process, which is not available in our case.

All recordings from the *Eval set* of AudioSet are used as the test set. This set contains around 4500 recordings corresponding to our set of 40 sound events. A subset of recordings from the *unbalanced set* of AudioSet is used for validation. Furthermore, to compare our webly supervised learning with a manually labeled set, we also create the

Method	**MAP**	**Method**	**MAP**
WLAT [119]	21.3	ResNet-SPDA [122]	21.9
ResNet-Attention [123]	22.0	ResNet-mean pooling [94]	21.8
M&mmnet-MS [94]	22.6	**Ours $\mathcal{N}_1$**	**22.9**

Table 5.1: MAP of $\mathcal{N}_1$ compared with state-of-the-art on whole AudioSet (527 Sound Events, Training: Balanced Set, Test: Eval Set)

Methods	MAP
$\mathcal{N}_1$-Self (Baseline) (4k)	38.7
$\mathcal{N}_2$-Self (4k)	41.4
WeblyNet (4k)	**45.3**

Methods	MAP
$\mathcal{N}_1$-Self (Baseline) (2k)	38.0
$\mathcal{N}_2$-Self (2k)	41.2
WeblyNet (2k)	**44.0**

Methods	MAP	Methods	MAP
$\mathcal{N}_1$-Self (Baseline)	38.7	$\mathcal{N}_1$ (Co-trained)	43.6
$\mathcal{N}_2$-Self	41.4	$\mathcal{N}_2$ (Co-trained)	43.5
$\mathcal{N}_1$-Self and $\mathcal{N}_2$-Self (Averaged)	43.5	WeblyNet	**45.3**
$\mathcal{N}_2$ [113]	42.6		

Table 5.2: Upper Tables: Comparison of systems on Webly-4k (L) and Webly-2k (R). Lower: $\mathcal{N}_1$ and $\mathcal{N}_2$ co-teach each other in WeblyNet, leading to improvement in their individual performances over training them separately. Results are shown on Webly-4k. See Sec. 5.3.2

AudioSet-40 training set. *AudioSet-40* is obtained from the *balanced set* of the AudioSet by taking all recordings corresponding to the 40 sound events in our vocabulary with over 4,600 recordings.

All experiments are done in the PyTorch toolkit. Hyperparameters are tuned using the validation set. The network is trained through Adam optimization [97]. Similar to other works [101, 119], we use Average Precision (AP) as the performance metric for each class and then the Mean Average Precision (MAP) of all classes as the metric for comparison.

5.3.1 Full AudioSet performance

We begin by assessing the performance of $\mathcal{N}_1$ (with C=527) on AudioSet. The primary motivation behind this analysis is to show that the architecture of $\mathcal{N}_1$ is capable of obtaining state-of-the-art results on a standard well-known weakly labeled dataset. Table 5.1 shows comparison with state-of-the-art. Our $\mathcal{N}_1$ can achieve state-of-art-performance on AudioSet (as of the time of this work). The performance of $\mathcal{N}_1$ on AudioSet shows that it can serve as a good base architecture for our WeblyNet system. Hence, it also serves as the baseline method for comparison.

5.3.2 Evaluation of Webly Supervised Learning

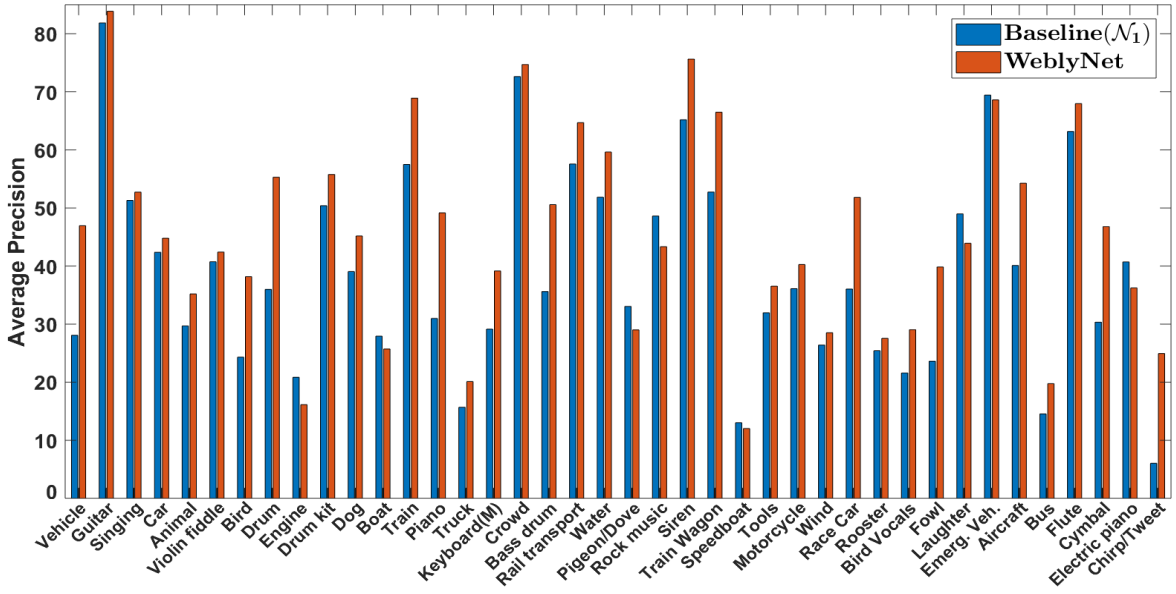


Figure 5.3: (Average Precision) AP for sound events on Webly-4k training. Comparison of baseline ($\mathcal{N}_1$-Self) and WeblyNet System

We first train $\mathcal{N}_1$ alone on the webly labeled dataset, and this performance ($\mathcal{N}_1$-Self) is taken as the baseline. We also train $\mathcal{N}_2$ alone on X_2 features ($\mathcal{N}_2$-Self) to assess the significance of the second view obtained through knowledge transfer in the webly supervised setting. To compare our method with a noisy label learning method, we apply the well-known approach described in [113] for training $\mathcal{N}_2$.

The two tables in the upper row of Table-5.2 show results for different systems on Webly-4k and Webly-2k training sets. We observe that WeblyNet leads to an absolute improvement of **6.6%** (**17%** relative) over the baseline method on the Webly-4k training set. Moreover, the X_2 representations from pre-trained AudioSet lead to considerable improvement over the baseline performance, around 7% and 8.5% relative improvements on the Webly-4k and Webly-2k training sets, respectively.

The lower table in Table-5.2 shows how our proposed system in which both networks co-teach each other leads to an improvement in the performance of the individual networks. First, we see that a simple combination of the two networks ($\mathcal{N}_1$-Self and $\mathcal{N}_2$-Self (Averaged) also leads to improved results. Once the WeblyNet system has been trained, we consider the output from individual networks $\mathcal{N}_1$ (or $\mathcal{N}_2$) as the output of the system. These are referred to as $\mathcal{N}_1$ (Co-trained) and $\mathcal{N}_2$ (Co-trained) respectively in Table 5.2. We can observe that the performances of both networks are improved by a considerable amount, over 12.7% for $\mathcal{N}_1$ (38.7 to 43.6) and over 5% for $\mathcal{N}_2$ (41.4 to 43.5). The overall WeblyNet system leads to 45.3 MAP.

Method	MAP	Method	MAP	Method	MAP
AudioSet-40 $\mathcal{N}_1$-Self (Clean data)	54.3	Webly-4k $\mathcal{N}_1$-Self	38.7	Webly-4k WeblyNet	**45.3**

Table 5.3: Webly labeled training vs. manual labeling (AudioSet-40)

Sounds	Webly-4k		Webly-2k	
	$\mathcal{N}_1$-Self	WeblyNet	$\mathcal{N}_1$-Self	WeblyNet
Vehicle	28.1	46.9	34.8	36.4
Singing	51.3	52.7	47.8	50.5
Animal	29.7	35.2	29.0	31.1
Water	51.8	59.6	50.4	51.9
Tools	31.9	36.5	40.7	40.6
Avg.	**38.6**	**46.2**	**40.5**	**42.1**

Table 5.4: APs for five classes with high label noise in webly sets.

Moreover, the noisy label learning approach from [113] leads to an improvement in $\mathcal{N}_2$ (to 42.6), which is 1% absolute less than the improvement in $\mathcal{N}_2$ obtained with our co-training approach (43.6). This shows that "learning together with agreement" is more effective than bootstrapping in [113]. Moreover, our overall system is 2.7% absolute (6.4% relative) better than that obtained through [113].

5.3.2.1 Comparison with manual labeling.

Table 5.3 shows the comparison of $\mathcal{N}_1$ trained on AudioSet-40 (which is manually labeled) with systems trained on the Webly-4k set. Note that the test set is the same for all cases, only the training set is changing. A considerable difference of 15.6%, exists between $\mathcal{N}_1$ trained on AudioSet-40 and that trained on Webly-4k. WeblyNet improves the webly supervised learning by reducing this gap to 9.0%. Such differences in performances between human-supervised data and non-human-supervised webly data have been discussed in computer vision as well. Often, human supervision is hard to beat even by using even orders of magnitude more data [103].

5.3.2.2 Class specific results.

Figure 5.3 shows the comparison of class-specific result comparison between baseline ($\mathcal{N}_1$-Self) and WeblyNet. WeblyNet improves over the baseline $\mathcal{N}_1$-Self for most classes (32 out of 40). In addition, we analyze the performance of five classes with high label noise (Fig. 5.1). Table 5.4 shows the AP for these sounds. For all of these events, the improvements are considerable, e.g., around 67% and 19% relative improvements

for Vehicle and Animal sounds, respectively. Interestingly, $\mathcal{N}_1$'s overall performance decreases for these 5 events as we increase the size of the dataset, from Webly-2k to Webly-4k. A considerable performance drop is seen for Vehicle and Tools sounds whereas only small improvements are seen for Animal and Water sounds. Deep learning methods are expected to improve as we increase the amount of training dataset. However, it is clear that the larger Webly-4k contains too many noisy labels, adversely affecting the performance in certain cases. The proposed WeblyNet system can address this problem. On average, WeblyNet gives 7. 6% absolute (20% relative) over the baseline method when trained on the Webly-4k set.

5.3.2.3 Effect of divergence measure.

To ensure that divergence plays a role in improving the system, we performed a sanity check experiment with $\alpha = 0$ in Eq. 5.2. The WeblyNet system, in this case, produces a MAP of 43.5, the same as the simple combination of the individually trained networks. This is expected as the networks are not tied together anymore, and one network will have no impact on the learning of the other.

To analyze the role of the divergence measure in the system's improvement, we remove it from the loss function in Eq. 5.2 (that is, $\alpha = 0$) and train the system using losses with respect to the targets only. This shows that the two networks actually co-teach each other by measuring the divergence between their outputs. Hence, the (dis)agreement between the networks can be used to train neural networks better.

5.4 Conclusions

Human supervision comes at a considerable cost, and hence we need to build methods which rely on human supervision to the least possible extent. We have presented webly-supervised learning of sound events as the solution. We presented a technique for mining web data and then a robust deep learning method to learn from webly labeled data. We showed that our proposed method in which networks co-teach each other leads to a considerable improvement in performance while learning from challenging webly labeled data. The method is extendable to more than two networks, although additional experiments have not shown significant gains from adding more networks. We also need better methods to mine the web for sound events. This can involve clever natural processing techniques to associate metadata with different sound events and then assigning labels based on that. [2]

[2]Code available at: https://github.com/cmu-mlsp/ankit-phd-thesis/tree/master/webly-labeled-sounds

6

Semi weak label learning

6.1 Exploiting Additional Cues with weak labels

Strong labels represent one end of the labeling spectrum, where every instance of an audio event must be fully tagged, including with its temporal boundaries. At an abstract level, it is helpful to think of data such as images or sound recordings as *bags* of candidate instances (e.g., candidate regions of the image or candidate sections of the recording), where every instance is labeled. Such labels are labor-intensive to obtain.

Weak labels represent, in some sense, the other end of the labeling spectrum, where labels are still provided, but only at the presence or absence of an event in a recording. Using the bag analogy, they only indicate if a bag of instances contains a particular class without indicating the specific instances or even the number of instances within the bag from that class. In terms of effort these require, perhaps, the lowest level of effort that still generates a label. The reduced labeling effort comes at a cost – there remains a considerable gap between the performances obtained from models trained with strongly labeled data and those trained from weakly labeled data. This leads to the question: is there an intermediate labeling strategy that requires only a relatively low increase in labeling effort over weak labels that can reduce this gap? This is the problem addressed in this chapter.

We introduce a middle ground between the two settings of full and weak supervision. In many settings, it is possible to annotate *count* information, which specifies the

number of occurrences of individual classes in a bag, for not much extra effort over simple weak labeling that merely tags their presence or absence. For instance, it is often fairly straightforward to annotate *how many* dogs there are in an image, if the annotation of exact bounding boxes is not required. Similarly, it is reasonable to expect that it is not significant extra effort to indicate how many (e.g.) gunshots were heard in an audio recording, if it is not required that their exact locations be tagged as well.

We refer to such labels as "semi-weak" labels, and the problem of learning from such data as as learning with semi-weak supervision, or learning from counts. We show that classifiers trained using semi-weak labels can classify test instances with much greater accuracy than those trained with merely weak labels.

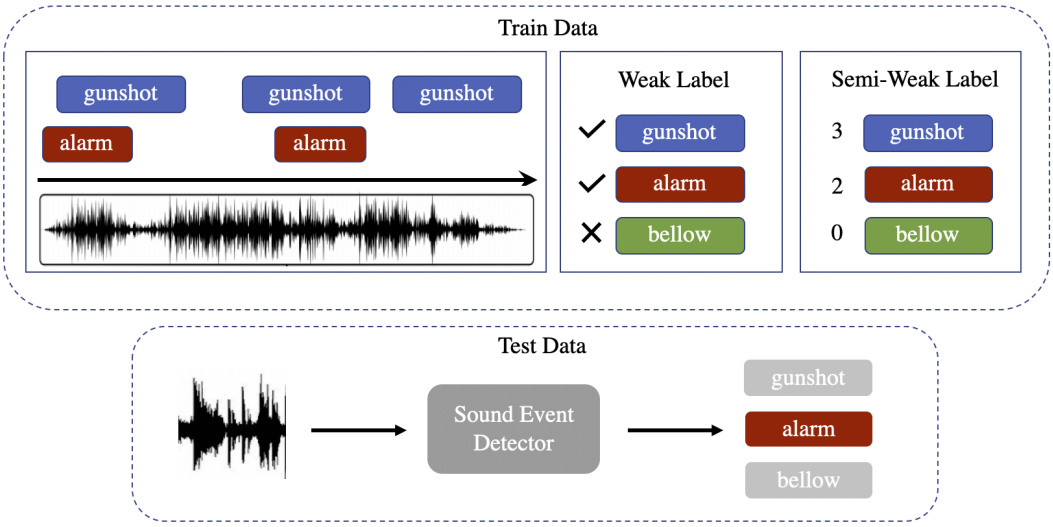


Figure 6.1: An example of semi-weak labels for audio event detection. Semi-weak labels give count information, while weak labels only provide information about the presence or absence of classes.

Our proposed solution for learning from semi-weak labels is to pose the training as a constraint satisfaction problem. The constraint to satisfy is that the sum of the counts predicted per category in each bag must equal the size of the bag. Thus, we first learn to predict whether the instance is present in the bag or not similar to the weak-label-learning setup, and then obtain an expected count of each class in a bag. Then, given the bag size and conditional expected count for each class, we propose to solve an optimization problem to translate the real valued expected count to a non-negative integer count. Finally, given the predicted count as "ground truth" and the instance-level logits, we try to find the best assignment of labels such that the counting requirement is satisfied as well as the likelihood of the bag is maximized.

6.2 Related Work

We first review some of the existing literature on prior approaches to using somewhat-less-weak labels than weak labels. These have generally considered counts and proportions in a bag.

6.2.1 Multiple Instance Learning

Learning from bag-level labels is often referred to as *multi-instance learning* (MIL). Carbonneau et al. [124] provides an extensive summary detailing most aspects of multiple instance learning. Dietterich [125] first introduced multiple instance learning where it was applied to predict drug activity. Since then, several algorithms have been introduced to address MIL. To deal with label noise, a natural solution is to count the number of positive instances in a bag and apply a threshold to tag a bag as positive. This was summarized in [126] as the threshold-based assumption for multi-instance learning. The threshold-based assumption, which defined that a bag is positive if and only if the number of positive instances is between a range, was the first time that count information was brought into multiple instance learning. Since then, several efforts[127, 128] have been made to predict at the bag level using count-based assumptions. [126] extended the assumption and proposed an SVM-based algorithm to predict the bag label. One common problem with those methods was scalability; also, those methods do not generalize to multi-class classification.

Multiple Instance Regression: MIL regression consists of assigning a real value to a bag. Compared to MIL classification, MIL regression has attracted much less attention in the literature. For MIL regression, one line of research has the assumption that some primary instance contributes largely to the bag label. This motivated sparsity-based approaches that assign sparse weights to instances and use regularization methods such as L1 and L2 regularizers [129, 130, 131]. However, most of these methods work only for small-scale data and focus on the accuracy of the predicted results rather than attempting to identify the primary instances that contribute to the bag label. Subramanian et al. [132] addressed the identification of primary instances by using a Dirichlet process to group the instances and find clusters. However, this method assumes that the largest cluster defines the label; this method also does not work for multi-class machine learning problems. Also, those methods do not work properly in our semi-weakly supervised setting, as they fail to incorporate the natural property of counts, which is a non-negative integer value.

Instance-Level Prediction: Another relevant mainstream of MIL research is instance-level prediction. Maron[133] is perhaps the best-known framework for instance-

level prediction for MIL. Following this framework, many research ideas have been proposed. The basic ideas of those frameworks are to label the instance dynamically or statically according to the bag label. Training instance-level classifiers is nontrivial as strong labels are unavailable. Recently, many methods have proposed to perform bag-level prediction and "hope" that bag-level accuracy could propagate to instance-level accuracy [125, 134, 135]. However, as discussed in [136, 137], this method is suboptimal. Empirical studies have been conducted in [137] that better bag-level prediction does not promise better instance-level prediction. Consequently, the most successful instance-level prediction models, i.e., mi-SVM and SI-SVM [134], discard the bag information as much as possible, and treat each instance individually.

6.2.2 Learning from Proportions

A related approach, previously proposed in the literature, is that of learning from *proportions* [138, 139, 140], where the proportion of instances in a bag that belong to any class is indicated. When the precise number of instances in individual bags is known, knowing proportions is identical to knowing instance counts of classes in the bag. But in other settings, such as in the image or sound examples mentioned above, it is much simpler and possibly far more natural to annotate the *counts* of occurrences of a class than the proportions. For example, in an audio recording that must be labeled for the occurrence of, say, dog barks, it is more intuitive and natural to state that a dog barks three times in the recording than to say (*e.g.*) that 20% of the recording comprised the sound of dogs barking. In fact, the main signature of sound events such as dog barked is their onset, which is generally distinct, while it is generally hard to spot precisely when the sound *terminates*. As such, while it is perfectly reasonable to state that the recording includes three barks, it may be infeasible, or even meaningless, to specify what fraction of the recording comprised dog barks as shown in Figure C.4.

Quadrianto [141] proposed the MeanMap model which assumes that the data follows an exponential distribution and is conditionally independent of the bags. Fan [142] and Patrini[143] further refined the loss function of MeanMap and made it applicable for multi-class classification. Yu [144] proposed another line of research and used SVM models to iteratively estimate the instance-level classifier. However, this method suffers from scalability issues when it is extended to a multi-class setting. More recently, a DL-based method was proposed by [138, 145, 146]. The commonality for the DL-based methods for the LLP problem is that they try to use bag-level supervision to perform instance-level estimation such that the estimated distribution is as close as possible to the bag label. Interestingly, [138] introduces another loss function that directly minimizes the prediction results at the instance level. They introduced the Optimal

Transport algorithm to make the loss function computationally tractable.

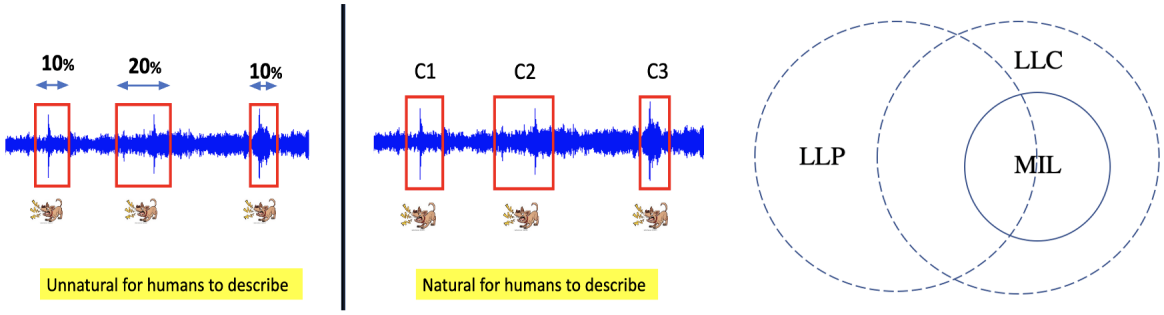


Figure 6.2: Left: We illustrate scenario which is natural for human to annotate between counts and proportions. Right: The relation between learning from proportion (LLP), learning from count (LLC) and Learning from Weak labels (MIL). The intersection part assumes that the bag size is known, and the count is exact. MIL problem is a special case of LLC where the count is equal to 1.

6.3 Problem Statement

Figure 6.1 shows an example setting of a problem in the detection of audio events. Generally, we consider a supervised multi-class classification problem, where we define the instance-level data $x \in \mathcal{X}$ and label $y \in \mathcal{Y}$, where $\mathcal{X} \subset \mathbf{R}^{H \times W \times C}$ and $\mathcal{Y} = \{0, \ldots, K\}$. Let K be the number of classes, N_B be the number of instances in a bag, and N be the number of bags. For each bag $b_i \in \mathbf{B} = \{\mathcal{X}\}$, $y_i^B \in \mathbf{Y}^B$ is the label of the bag level. More specifically, $y_{i,k}^B$ is the count of class k in bag i. Formally, given the bag size, we define the space of labels as

$$\mathbf{Y}^B = \{c \in \mathbb{Z}_+^K : \sum_{i=1}^{K} c_i = N_B\} \tag{6.1}$$

Given a series of bags, $S = [(b_0, y_0^B), (b_1, y_1^B), \ldots, (b_N, y_N^B)] \subseteq \mathbf{B} \times \mathbf{Y}^B$, our goal is to learn a predictor $f : \mathcal{X} \to \mathbb{R}^K$ that predicts the probability distribution of an instance belonging to different classes, i.e. $f_\theta(x_i) = \{p(y_i = 0|x_i), p(y_i = 1|x_i), \ldots, p(y_i = K|x_i)\}$, where x_i is a feature vector for an instance and θ is the parameters of the network.

6.3.1 A General Loss Function

In this problem, we focus on instance-level prediction and solve it by viewing the problem as a risk minimization task. More specifically, given a regression loss function $L_{reg} : \mathcal{B} \times \mathbf{Y}^B \to \mathbf{R}_+$ defined as

$$R_N(f) := \frac{1}{N} \sum_{i=1}^{N} L_{reg}(f(x_i), y_i^B) \tag{6.2}$$

One intuitive approach is the define a divergence function that quantitatively quantify the distance between the predicted bag labels and the ground-truth label. As such, let's define $\forall b_i \in \mathcal{B}$, the predicted count as $\tilde{c_k} = \sum_{j=1}^{n_i} f(x_j)$, and we want to minimize the divergence between the predicted counts and the actual counts

$$L_{reg}(f(x_i), y_i^B) \coloneqq \frac{1}{n_i} div(\tilde{c_k}, y_i^B)) \tag{6.3}$$

Usually, learning from counts could be formulated as a regression problem. However, for interclass separation of the features at the bag-level and to tackle the problem as multi-label bag classification, we used binary cross-entropy to recognize different classes existing in a bag. So let $\mathbb{1}(\cdot)$ denotes the indicator function, taking values 1 or 0 depending on whether its argument is true or not, we define another loss term as follows:

$$F_{max} = max_{i \in 0,1,...,B_N}[f(x_0), f(x_1), ...f(x_{B_N})] \tag{6.4}$$

Additionally, we require the loss function to be robust to the sparsity of the bag and the generalizability of the loss function is critical in the learning process. Thus, we introduce one L1 regularization term that makes the model perform better when the bag is sparse or "pure". Here, we define sparseness based on the bag level counts of each class. Our proposed loss function with the regularization term is defined as follows

$$R_N(f) = L_{reg} + \alpha L_{cls} + \beta \sum_{x_i \in b_i} \frac{1}{N_B} \|f(x_i)\|_{L1} \tag{6.5}$$

6.4 Preliminary: DLLP Approach

DLLP [145] approach is a DNN-based model for learning from proportion. Mathematically, given a bag $b_1, b_2, \cdot, b_N$, suppose that $f_\theta(b_{ij}) = p_\theta(\mathbf{y} \mid \mathbf{b}_{ij})$ is the vector outputs of the DNN for the j-th instance in b_i. Let $\bigoplus$ is an element-wise addition operator. Then the posterior bag-level class proportion could be estimated as

$$\overline{\mathbf{p}}_i = \frac{1}{N_B} \bigoplus_{j=1}^{N_B} p_\theta \left(\mathbf{y} \mid \mathbf{b}_{ij}\right) \tag{6.6}$$

The final objective, i.e., L_{dllp}, is made of a distance measurement. Let $\mathbf{c}_i^k$ be the counts of class k in bag i, $\mathbf{p}_i^k$ be the counts normalized by bag size, and $\overline{\mathbf{p}}_i^k$ be the k-th element in the vector $\overline{\mathbf{p}}_i$ (the estimated proportion of class k), then the loss is defined as $L_{dllp} = KL(\mathbf{p}_i, \overline{\mathbf{p}}_i)$. KL refers to the commonly used Kullback–Leibler divergence for measuring the distance between two distributions.

6.5 Proposed Methods: Two-Stage Framework

In this section, we introduce a two-stage model for learning from counts. It is extended from DLLP [145].

6.5.1 Stage-1: Estimating Class Count

6.5.1.1 Poisson Loss and Expected Count

As mentioned in Section 6.1, one difference between LLP problem and LLC problem is that the label for LLC problem is discrete. So it is not optimal to use a KL-divergence to measure the difference between prediction and the true labels. Instead, we propose to use Poisson Loss - a Poisson distribution based loss function. The Poisson distribution is the discrete probability distribution of the number of events occurring in a given time period, which applies to the LLC setting if we consider the counting is the number of times a class instance appears in this bag [147].

In our framework, we assume that the counting for class j follows a Poisson distribution, i.e. $p(c_i^j = k|\lambda_j) = \frac{\lambda_j^k e^{-\lambda_j}}{k!}$. For simplicity, since we assume the size of the bag is unknown, we assume the count for each class in a bag is independent. Given a bag of output of the network $p_\theta(\mathbf{y} \mid \mathbf{b}_{ij})$, we define $\overline{\mathbf{c}}_i^k$ as the expected count for class k in bag b_i.

$$\overline{\mathbf{c}^k}_i = \bigoplus_{u=1}^{N_B} p_\theta\left(\mathbf{y} \mid \mathbf{b}_{ij}^u\right) \tag{6.7}$$

So we have

$$\hat{\lambda} = \hat{\mathrm{E}}(c_i^k \mid x) = \overline{\mathbf{c}^k}_i \tag{6.8}$$

Then the Poisson loss is defined as the negative log likelihood function

$$L_{reg}(y, \hat{\lambda}) = -\log(p(y|\hat{\lambda})) = \hat{\lambda} - y\log(\hat{\lambda}) + \log(y). \tag{6.9}$$

Because the last term in Equation 6.13 would be a constant for a given bag, it is usually omitted. So, the final loss and its gradient is

$$\begin{aligned} L(y, \hat{\lambda}) &\propto \hat{\lambda} - y\log(\hat{\lambda}) \\ \nabla_{\hat{\lambda}} L(y, \hat{\lambda}) &= 1 - y/\hat{\lambda}. \end{aligned} \tag{6.10}$$

The following are two interesting properties for the Poisson loss. **Adapted gradient:** Unlike other distance functions like mean absolute error, the Poisson loss doesn't have a constant gradient for all input values. Also, when the actual count is large, the gradient value would be relatively smaller. This matches our intuition that when the actual

count is very large, an off-by-1 error matters less than if the actual count is 1 or 2. **Asymmetric gradient:** the gradient is zero when $\hat{\lambda} = y$ but the gradient is different when $\hat{\lambda} = y - 1$ and $\hat{\lambda} = y + 1$, where the absolute gradient would be $1/(y-1)$ and $1/(y+1)$ respectively. The gradient tends to focus on penalizing the under-estimation rather than over-estimation.

In addition, we introduce a classification loss, L_{cls}, and a regularizer $L_{L1} = \sum_{j=1}^{N_B} \frac{1}{N_B} \left\|\overline{\mathbf{p}}_{ij}\right\|_{L1}$ to make our model more robust to sparse bags [148] and ensure the inter-class separability of the learned representations [149]. L_{cls} is nothing but a binary cross-entropy loss to measure whether the network can classify if a class is present/absent in a given bag. We use a unified loss function defined as

$$L_{LLC} = L_{reg} + L_{cls} + \beta L_{L1} \tag{6.11}$$

6.5.1.2 Estimating Class Count

Although we have $\hat{\lambda_j}$ as the expected count for class j, it remains a problem to estimate the exact count for each class. Mathematically, let $\overline{\mathbf{c}}_i^j \in \mathbb{R}$ be the expected count for class j in bag b_i, and then $t_i \in \Delta_K = \left\{c \in \mathbb{Z}_+^K : \sum_{j=1}^K c_i^j = N_B\right\}$, then the exact count for each class in b_i could be obtained by

$$\hat{t}_i = \arg\max_{t \in \Delta_K} \sum_{j=1}^{K} \log\left(p\left(c_i^j = t_i \mid \overline{\mathbf{c}}_i^j\right)\right) \tag{6.12}$$

This is constrained convex optimization and we can use any greedy algorithm to get the optimal solution. Initializing $\hat{t}_i$ as a vector of zeros, for each iteration, we manually calculate the marginal gain to increase $\hat{t}_i^j$ to $\hat{t}_i^j + 1$. We choose to increase the count of a class such that the increment is maximized, and iterate until the sum of the count vector is equal to N_B. Using a heap to track the maximum value, we can design an algorithm with time complexity of $O(log(K) * N_B)$ as shown in Algorithm 2.

6.5.2 Stage-2: Estimation of the Instance Label (Decoder)

In Stage-1, the network outputs the expected count for each class, and then we develop an optimization problem to translate the expected count to the exact count. In stage-2, given the exact count of each class and the predicted probability distribution of each instance $p_\theta(y|b_{ij})$, we devise a assignment problem to obtain the instance-level label. Thus, we have the following linear-sum optimization problem.

$$\begin{aligned} \max_x \sum_j^{N_B} \sum_k^K log(p_{ij}^k) * x_{ij} \\ s.t \\ \sum_k x_{ij}^k = 1, \forall j \in \{0, 1, \cdots N_B - 1\} \\ \sum_j x_{ij}^k = \overline{c_i}^k, \forall k \in \{0, 1, \cdots K - 1\} \\ p_{ij}^k \geq 0 \\ x_{ij}^k \in \{0, 1\}, \forall j, k \end{aligned} \tag{6.13}$$

where $p_{ij}^k = p_\theta(y = k|b_{ij})$, $\overline{c}_i^k$ is the estimated exact count of class k in the bag i and $x_{ij}^k = 1$ if and only if the instance j is assigned with the label k.

6.5.3 Algorithm: Greedy algorithm to obtain exact count and Semi-Weak Label Classification System

The following algorithm provides the greedy approach chosen to translate from the expected count to the exact count.

Algorithm 2 Greedy Algorithm to Translate Expected Count to Exact Count

Input: Number of bag size N_B; Number of classes K; A list of floating points $[a_0, a_1, \cdots, a_K]$; a list of probability density function for Poisson distribution $[f_{a_0}(x), f_{a_1}(x), \cdots, f_{a_K}(x)]$

Output: A list of integers $[t_0, t_1, ..., t_K]$ such that $\sum_{j=0}^K t_j = N_B$.

Initialize: Let $q = heap < int, int >$ be a max-heap that stores pairs. Heap has two methods: $q.pop()$ would pop and return the max value from the heap and $q.push(v, i)$ would add a pair of $< v, i >$ to the heap. The comparison of pair is based the first value in the pair;

Let $ret = [0, 0, \cdots, 0]$ be a zero-initialized vector of size K;

for $i = 1, 2, ...K$ **do**
    $r \leftarrow f_{a_i}(ret[i] + 1) - f_{a_i}(ret[i])$
    $q.push(r, i)$
end for
for $\_ = 2, ...N_B$ **do**
    $v, i \leftarrow q.pop()$
    $ret[i] \leftarrow ret[i] + 1$
    $r \leftarrow f_{a_i}(ret[i] + 1) - f_{a_i}(ret[i])$
    $q.push(r, i)$
end for
return ret

The following section provides the algorithm 3 for our semi-weak label classification system. In theory, the weak label classification system can work for all kinds of input

data for classification whereas as an initial validation of the proposed algorithm, we run experimental analysis on an image classification task.

Algorithm 3 Semi-Weak Label Classification System

Input: *batchsize* number of bags $[(b_0, y_0^B), (b_1, y_1^B)..., (b_N, y_N^B)]$, loss weight α, and regularization weight β, number of epochs n_{epochs} and a loss function for regression L_{reg}
Output: A predicted label for each instance $[y_0, y_1, ..., y_N]$.

for $n = 1, 2, ...n_{epochs}$ **do**
  for $i = 1, 2, ...N_B$ **do**
    Compute the class-wise distribution for x_i.
  end for
  for $k = 1, 2, ...K$ **do**
    Compute $C_k = \sum_{i=1}^{N_B}$.
  end for
  Compute loss, L_{reg} w.r.t label y_B,C_k.
  Compute bag-level binary label using max pooling.
  Compute loss, L_{cls} w.r.t label y_B.
  Compute the regularization term and compute the final loss given α, β
  Update networks based on the combined loss.
end for
for $i = 1, 2, ...N$ **do**
  for $i = 1, 2, ...N_B$ **do**
    compute the the probability distribution for x_i using learned classifier.
  end for
  Enumerate all combinations of counts and get the guess t that maximizes the likelihood of b_i.
  Given t and the probability distribution, using linear sum max algorithm to get the best assignment $y_0, y_1, ...y_{B_N}$
end for

6.6 Experiments and Results

6.6.1 Datasets

We conducted experiments based on the CIFAR-10 dataset. CIFAR-10 dataset consists of 50000 train and 10000 test images with each class consisting of 5000 and 1000 training and testing images, respectively. We created bags of different sizes from each of the dataset images with and without replacement in our analysis. Each bag was created according to the size ranging from 2, 4, 6, 8, 16,32 bag sizes. In the case of the images with the replacement, we make sure that the number of images being repeated is restricted with a reuse parameter, which is controllable to avoid a bag being over-represented by the same image. For each data set generated with a different bag

size, we ensure that the parameter configuration is set such that the maximum amount of the CIFAR-10 data set is used during both the training and testing phase is used to represent the newly generated dataset with semi-weak labels. As the CIFAR-10 dataset has a training to testing image ratio is maintained as 5:1, the same ratio is maintained in different datasets generated as shown in table 6.1

We generate the dataset with the following distribution: Poisson, exponential and uniform distribution. This distribution is based on how the class information is distributed in each given bag. The rationale with the generation of the instances of the bag in the above format was to ensure that the bags are generated based on naturally occurring instances in nature for the Poisson distribution, the exponential distribution, and the uniform distribution. The generation process is the following. 1) randomly sample a class i with equal probability. 2) sample a number n_i from the given distribution and then truncate it into the range between 0 and N_B. 3) sample n_i instances from class i. In the case where the number of instances needed would be more than the total number of instances, the instances would be sampled at most two times. For each parameter setting, we sample the bags with 5 different random seeds. All the experiments are conducted on these 5 trials and the final value is obtained by averaging the best performance on the validation set. In the standard setting, to try to generate more balanced data, where each class has some instances in a bag. Therefore, we selective choose a hyperparameter for the distribution that minimizes the sparsity level of the bag. We define the sparsity of a bag as the number of absent classes divided by the number of classes. In particular, we use the following settings for the Poisson distribution $[(N_B = 2, \lambda = 0.5), (N_B = 4, \lambda = 0.5), (N_B = 8, \lambda = 1.2), (N_B = 16, \lambda = 2.0), (N_B = 32, \lambda = 3.2)]$, and $[(N_B = 8, \lambda = 0.67), (N_B = 16, \lambda = 0.5)]$ The data summary table is provided in Table 6.1. For uniformly distributed samples, directly sample the number N_B of samples from the original dataset and label them with the counting vector.

6.6.2 Architecture

We use a Residual Network with 18 layers as our base backbone for extracting features from the image. Given a bag of images $\{x_0, x_1, ..., x_{N_B}\} \in \mathcal{B}_{\mathcal{N}}{}^{L \times H \times C}$, the embedding $e_i \in \mathcal{R}^{B_N \times K}$ for each instance is first extracted using a Convolutional Neural Network as the feature extractor. Then a linear layer $fc : \mathcal{R}^d \to \mathcal{R}^K$ is used to create a class activation map. Different pooling layer is applied at the end to generate counting prediction vector and logits for multi-label classification problem. All models are trained using Stochastic Gradient Descent with an initial learning rate of 0.01 for 100 epochs. The learning rate would be manually divided by 10 at epochs 30 and 50. The

Table 6.1: Generated Data Summaries

Distribution	Bag Size	lambda	# of training bags	# of testing bags	Avg. Count	Avg. (Std) Sparsity	Dataset Id
Poisson	2	0.5	40,000	8,000	1.13	82%(4%)	p0
	4	0.5	22,000	4,500	1.28	68%(7%)	p1
	8	1.2	10,000	2,000	1.63	50.02%(10.53%)	p2
	16	2	4,000	1,000	2.28	28.9%(12.4%)	p3
	32	3.2	2,000	800	3.65	8.72%(8%)	p4
	8	1.2	5,000	1,000	3.43	8.32%(8%)	p5
	8	1.2	1,000	200	3.56	9%(8.2%)	p6
	16	8	4,000	1,000	6.52	79%(6%)	p7
	16	2	2,000	500	6.32	80%(6.2%)	p8
	16	2	1,000	200	6.21	79%(6.1)	p9
Exponential	8	0.67	10,000	2,000	1.97	58%(11.7%)	e0
	16	0.5	4,000	1,000	2.89	42%(13.9)	e1
Uniform	8	N/A	10,000	2,000	1.41	42%(9.21%)	u0
	16	N/A	4,000	1,000	1.95	18.7%(9.72%)	u1

weight decay is set to 5e-4 for training. Standard data augmentations are utilized to avoid overfitting of CIFAR10 dataset, including random crop with padding of 4 and random horizontal flip with a probability of 0.5.

6.6.3 Baselines

Fully-supervised Upper-bound: We want to argue that comparable results could be achieved by using semi-weak label compared to strong labels. Therefore, we also train ResNet18 in a fully supervised setting on CIFAR10. We trained it 250 epochs with an initial learning rate of 0.1 and then divided it by 10 in the mid-training. This model could be deemed as an upper-bound of the performance in a semi-weak setting. For our analysis with different base network architectures, we further use similar hyperparameters.

Weakly Supervised Baseline: We produce results with the weakly supervised baseline where we consider the loss from counting to be absent and thus we generate the results based on bag level prediction where only presence or absence of the class is available.

Learning from Proportions: If we set the loss function as KL-loss, the our model without the decoder could be used a baseline model to represent the performance of modeling the learning from counts as LLP.

6.6.4 Evaluation

We use precision to evaluate our framework for both instance-level prediction and bag-level prediction. For bag-level prediction, we compute the precision rate using macro averaging. For fully supervised model, we predict the class label individually and label the bag positive for class i if there is at lease one predicted instance for class

i in this bag. Similarly, for semi-weakly supervised model, we label a bag according to the instance-level predicted results.

Table 6.1 provides a detailed summary of all the generated dataset configurations. We use the following settings for the Bag size where $N_B \in [2, 4, 8, 16, 32]$ and the number of training instances are chosen such that we utilize most of the data available in the CIFAR-10 dataset. The average count here represents the aggregated mean of average number of instances per class. We represent the sparsity based on the percentage of classes with zero instance in a bag. Our experimental results for all the dataset configurations from Table 6.1 are summarized in Table 6.2. To remove the effect of randomness, for each dataset setting, we train the model with 5 different random seeds and then average the results across all trials. The comparison between baselines are discussed in the loss ablation section 6.6.9

Table 6.2: Benchmarking Results on Different Datasets using Semi-weak label

Dataset ID	Bag Prec.	Inc. Prec.	Dataset ID	Bag Prec.	Inc. Prec.
p0	94.24	92.76	p7	64.37	85.20
p1	93.78	92.65	p8	89.92	81.92
p2	93.20	91.21	p9	84.97	69.93
p3	92.92	88.07	e0	90.69	91.05
p4	93.9	71.91	e1	87.10	87.65
p5	90.54	87.73	u0	94.86	91.42
p6	77.39	68.91	u1	95.62	87.34

6.6.5 Baseline Results

Fully-supervised baseline:Table 6.3 shows the results obtained with the different classifiers constructed as the base classifier for the classification with the CIFAR-10 dataset. We find that the bag level prediction on the dataset p2 is comparable to the fully-supervised dataset which is expected as the semi-weak label dataset will be upper-bounded by the fully supervised setting.

Table 6.3: Fully supervised upper-bound results with classifier configuration

Classifier	Loss	Bag Prec.	Inc. Prec.
ResNet18	Poisson	94.26	94.40
ResNet34	Poisson	94.67	94.681
ResNet50	Poisson	95.2	96.41
MobileNetV2	Poisson	90.03	89.066

Learning-from-proportion baseline: For the learning-from-proportion baseline, results are shown in Table 6.7 (KL v.s Poisson) because we use KL-loss to proxy the baseline model of learning-from-proportion. This baseline achieved 87% instance-level precision, which is less than our proposed model on dataset p3, i.e., 88.07%. This result also holds if we consider bag size equals to 8 for dataset p2.

Table 6.4: Effect of the Choices of L_{reg} on two standard dataset setting with $N_B \in \{8, 16\}$

Dataset ID	L_{reg}	Bag Prec.	Inc. Prec.
p5	KL	90.26	90.40
p5	L1	92.67	90.681
p5	Poisson	93.2	91.21
p3	KL	87.03	87.066
p3	L1	92.01	85.97
p3	Poisson	92.92	88.08

Learning-from-weak-label baseline: For the learning-from-weak-label baseline, we report those number in Table 6.5. Once we deactivate the counting loss, then the model is converted into the traditional weak label setting. In Table 6.5, without counting loss, the performance is much worse than our proposed semi-weak labeling framework. It only achieves 87% instance-level prediction precision, which is 5% worse than our proposed framework.

Table 6.5: Ablation Study of Removing Regression Loss and Classification Loss. ($\beta \in \{0, 1.0\}$)

Dataset ID	Bag Prec.	Inc. Prec.
p2	93.20	91.21
p2 (w/o L_{cls})	93.23	91.17
p2 (w/o L_{reg})	90.92	87.39
p8	92.92	88.07
p8 (w/o L_{cls})	62.13	83.92
p8 (w/o L_{reg})	41.81	72.04

6.6.6 Ablation Study for the Decoder

As is shown in Table 6.6, instead of using greedy search, we use the proposed three-stage framework to infer the instance-level labels. Results show consistent improvement after using the decoder algorithm. More importantly, when the bag is sparse, the counting for each class is high and thus the estimation would have high variance. Therefore, when the bag is less sparse, greedy search works well. But once the bag size increases or the bag becomes sparse, the high variance of the estimated logits would make the greedy

solution less attractive. Therefore, the decoder contributes more to the performance of the predicted value (+2%) as the greedy predictions become unstable.

Table 6.6: Ablation Study for the Decoder

Dataset ID	Inc. Prec.
p2	91.21
p5	87.73
p7	85.20
p2(+Decoder)	92.08
p7(+Decoder)	88.92
p7(+Decoder)	87.32

6.6.7 Variation with the Number of Training Samples

We investigate the degree to which the size of the dataset relates to the model performance. We progressively sample different number of bags and evaluate them on the test set. We compare the results using different scales of training samples. The dataset settings are illustrated in Table 6.7.

Table 6.7: Scale Testing Results on two standard dataset setting with $N_B \in \{8, 16\}$.

Dataset ID	Bag Prec.	Inc. Prec.
p2	93.2	91.21
p5	90.54	87.74
p6	77.39	68.91
p3	92.93	88.08
p8	89.92	81.92
p9	84.97	69.93

6.6.8 Variation with the Batch Sizes and Architectures

We conduct experiments on variants batch sizes and find different performance characteristics. For all experiments, we train the model with batch size equal to 32, 64, 96, 128 and 256 as seen in Table 6.8 and bag size equal to 8 and 16 as the standard setting. We found that the results with the batch size doesn't provide large variation in the output of the optimization for the bag level training. However, as we increase the batch size there is a drop in performance observed on the instance level prediction.

Table 6.8: Ablation Study of Using different backbones and different batch sizes on dataset p2 $B_N = 8$.

Backbone	Bag Prec.	Inc. Prec.
Resnet18 (bs=32)	94.036	92.096
Resnet18 (bs=64)	93.20	91.21
Resnet18 (bs=96)	93.630	91.944
Resnet18 (bs=128)	93.2	91.23
Resnet18 (bs=192)	92.661	90.649
Resnet18 (bs=256)	92.541	90.183
Resnet34 (bs=128)	93.763	91.763
Resnet50 (bs=128)	93.512	91.893
MobileNetV2 (bs=128)	91.474	89.002

6.6.9 Effectiveness of Different Regression Losses

As is proposed for learning from proportions, KL loss is mostly widely used loss function for comparing the estimated class distribution and the true class distribution. Therefore, we also tried to use KL-loss as well L1 loss, which is usually used for multi instance regression as alternative choices.

As is summarized in Table 6.7, poisson loss function performs better than other loss functions for bag size equal to 8 and 16, in terms of bag level prediction as well as instance-level prediction. This is as expected as poisson distribution is naturally defined for non-negative and integer values like counting. Another explanation is that the data is generated from poisson distribution but according to Table 6.9, even though the dataset is generated from non-poisson distribution, the poisson loss function still achieve comparable values. This implies that poisson distribution is robust.

6.6.10 Effect of Regularizer with Sparse Bags

In order to analyze the model performance on bags with different sparsity, we further create bags with higher expected counts for each class to generate bags with higher sparsity. The specific setting and data statistics are summarized in Table 6.1

Interestingly, we found that the regularized term works really well for sparse bags. It consistently improve the performance thought marginally.

6.6.11 Performances over Different Bag Sizes

As we see in table 6.1, we have different bag sizes generated with a variation of the parameters lambda and number of training bags to create new copies of dataset with different bag size. Table 6.2 shows that as we increase the bag size the performance

Table 6.9: Effect of the Regularizer on Sparse Bags (Dataset ID=p7)

β	Bag Prec.	Inc. Prec.
0.0	63.7	84.67
0.01	64.37	85.20
0.1	65.09	85.52
0.5	65.07	85.87

remains fairly increasing upto a bag size of 8 and then there is a gradual decrease in performance observed with large drop seen with the bag size of 32. Such variation can be attributed to the fact that with the larger bag size the nature of the distribution of the bags to the instance level information per class is artificial and there is a consistent decrease in the performance as we increase bag size from 16 to 32 and beyond.

6.6.12 Comparison with Different Distribution

Besides generating data from poisson distribution, we also train our models with the exponential distribution as well as the uniform distribution. We observe that the results with the poisson distribution is better as compared the exponential distribution which is in accordance with our expectation as the count information in case of the exponential distribution will be sparse than case of the poisson distribution. The analysis can be reviewed in the table 6.2 for case of the dataset id e0 and p2 there is a stark difference with the same bag size and sparsity distribution but difference in the distribution of classes in each bag results in performance drop from 93.2% precision to 90.69% precision.

6.6.13 Example of Count assignment optimization

Figure 6.3 gives an example with bag size of 4 and class size 3. If we negate the reward, then the objective becomes minimizing the "cost", which is referred to a classical linear-sum assignment problem in bipartite graphs [150].

6.7 Using Semi Weak labels for Sound event detection

HTS-AT leverages a hierarchical Transformer-based architecture coupled with multi-scale attention mechanisms designed specifically for audio processing. Unlike traditional convolutional neural networks (CNNs) that primarily capture local time-frequency patterns, HTS-AT effectively models both short-term and long-term dependencies in complex acoustic scenes. By jointly attending to multiple temporal scales, HTS-AT can handle overlapping events and fine-grained temporal structures often encountered in real-world audio. Additionally, its token-semantic representation of audio signals facilitates

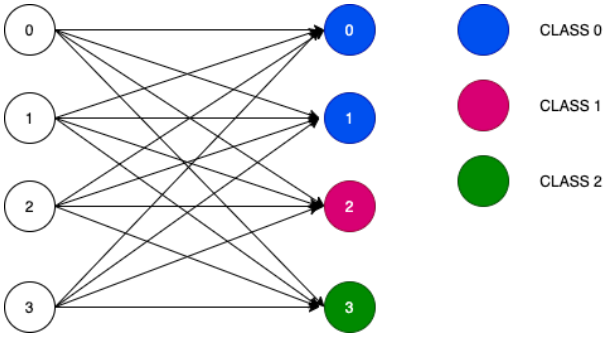


Figure 6.3: An example of assignment problem. The nodes on the left represent the unsigned instance and on the right represent the label. There are 4 instances in a bag and with 2 class-0 instances, 1 class-1 instance and 1 class-2 instance. Each edge has an associated reward p_{ij}, which is probability of instance i belonging to class j. Each instance must have one assignment and the goal is to maximize the reward.

improved interpretability, as each token corresponds to a meaningful acoustic concept. As a result, HTS-AT provides a flexible and robust framework that not only excels in classification tasks but also enhances temporal localization and event segmentation, making it a compelling choice for advancing state-of-the-art SED performance. The Hierarchical Transformer for Sound Event Detection (HTS-AT) outputs an event presence probability map that indicates the probability of an event's occurrence over time. To incorporate count information, we first refine this map and then estimate how many times each event occurs. The main steps involve thresholding, median filtering, and counting the number of event onsets.

6.7.1 Thresholding the Event Presence Map

Let $P(y_c \mid x_t)$ be the probability that event class $c \in \{1, \ldots, C\}$ occurs at time $t \in \{1, \ldots, T\}$. To obtain a binary map indicating event presence or absence, we apply a threshold δ:

$$\hat{P}(y_c \mid x_t) = \begin{cases} 1, & \text{if } P(y_c \mid x_t) > \delta, \\ 0, & \text{otherwise.} \end{cases} \tag{6.14}$$

Here, $\hat{P}(y_c \mid x_t) \in \{0, 1\}$ represents the binarized event presence at time t.

6.7.2 Median Filtering

The binarized presence map may contain noisy spikes or brief gaps. To address this, we apply median filtering across the temporal dimension. Let M be the size of the median filter window. The median-filtered presence map $\hat{P}^{\text{mf}}(y_c \mid x_t)$ is defined as:

$$\hat{P}^{\text{mf}}(y_c \mid x_t) = \text{median}\big(\hat{P}(y_c \mid x_{t-\lfloor M/2 \rfloor}), \ldots, \hat{P}(y_c \mid x_{t+\lfloor M/2 \rfloor})\big). \tag{6.15}$$

This step removes isolated spikes and fills in short gaps, ensuring that only consistent, sustained activations are maintained.

6.7.3 Counting Event Occurrences

Once we have a smoothed, binarized event presence map $\hat{P}^{\text{mf}}(y_c \mid x_t)$, we estimate the number of event occurrences by counting transitions from 0 to 1 as we move forward in time:

$$\text{count}_c = \sum_{t=2}^{T} \max\big(0, \hat{P}^{\text{mf}}(y_c \mid x_t) - \hat{P}^{\text{mf}}(y_c \mid x_{t-1})\big). \tag{6.16}$$

The value count_c is the estimated number of occurrences of event class c within the time window.

6.7.4 Incorporating the Count Loss

If ground-truth counts N_c^{true} are available for each event class c, we can include a count loss to train the model to predict accurate counts. Let N_c^{pred} be the predicted count obtained from count_c. We define the count loss as a mean squared error (MSE):

$$L_{\text{count}} = \frac{1}{C} \sum_{c=1}^{C} (N_c^{\text{pred}} - N_c^{\text{true}})^2. \tag{6.17}$$

This count loss can be combined with the original classification loss $L_{\text{classification}}$:

$$L_{\text{final}} = L_{\text{classification}} + \lambda L_{\text{count}}, \tag{6.18}$$

where λ is a hyperparameter controlling the trade-off between classification performance and counting accuracy.

6.7.5 Results

By integrating the count information, we observe increase in the sound event detection performance for AudioSet dataset from 0.385 mAP (without counts) to 0.403 (with counts) added.

6.8 Conclusion

In this chapter, we propose various ways of "strengthening' weak labels in a manner that does not impose serious additional labelling burden.

We propose "semi-weak" labelling: a novel machine-learning problem that learns from counts. We propose a two-stage framework to do instance-level prediction given only a counting vector for a bag is available. We generated a dataset from CIFAR10 for experimentation. We achieve comparable results with the fully-supervised setting and much better results than the weakly supervised setting. Additionally, we introduced an L1 regularization term that makes our model robust to sparse bags and achieves marginal prediction improvement on sparse bags. We propose additional forms of semi-weak labelling including those that work from bounds on counts, approximate counts, and sequentiality information, all of which are easily obtained. We believe semi-weak labels provide better insights in real-world tasks where counting information can be easily obtained and future research can extend our work on other modalities or mixture of modalities. [1]

[1] Code: https://github.com/cmu-mlsp/ankit-phd-thesis/tree/master/semi-weak-label-learning

Section IV - Enriching Weakly Supervised Learning

In weakly supervised learning, limited label precision poses unique challenges to model robustness and adaptability. However, this shift in labeling strategy introduces unique challenges in ensuring that models remain robust, accurate, and adaptable. The chapters in this section present three distinct but complementary approaches to push these boundaries of what is possible with weakly supervised learning: strategic negative sampling and ontological based enhancements.

Chapter 7 delves into the role of negative sampling in the refinement of classifiers trained on weak labels. By identifying and selecting the most informative negative samples, it highlights a pathway to improve model performance even when only approximate class presence is known. Rather than overloading the classifier with countless uninformative negatives, it shows how more surgical selection can foster better decision boundaries and more nuanced understanding of complex categories.

Chapter 8 focuses on the integration of ontologies - structured, hierarchical representations of domain knowledge - into weak label learning frameworks. Ontologies capture the interconnectedness of concepts, providing models with a robust backbone of domain-specific information. This infusion of structured knowledge helps to disambiguate related classes, bridge semantic gaps, and yield better generalization, especially when data labels are limited or uncertain. The combined effect of these three approaches illustrates a future where weakly supervised models are not only cheaper and faster to train, but also more reliable, interpretable, and domain-aware.

In essence, these chapters together map an exciting territory for weakly supervised learning. They show how we can go beyond simply making do with less information and begin actively leveraging the unique properties of weak labels to discover deeper patterns, refine representations, and harness domain knowledge. The advances discussed here promise to reduce our dependence on costly strong labels, while achieving a new level of performance and understanding in machine learning models.

7

Importance of sampling negative in weak labels

7.1 Problem: Sampling of negatives in weak labels

Data selection is essential for deep learning, which determines the budgets and the performance of the network. Motivated by reducing the total amount of data fed into the neural network, we study the selection of more informative negative samples and the underlying principles to quantify the importance of data. Compared to our baseline, our experiment results on CIFAR-10 and AudioSet propose that negative sampling strategies can improve classification performance under weak label settings. Through these negative sampling strategies, we elegantly remove negative data that are less important while retaining AP and AUC scores even when we use a 1/3 subset rather than the whole data set.

Weak labels imply the knowledge of the presence/absence of an event rather than the exact location of it compared to strong labels. With strongly labeled data, one can only work with a limited number of labels, and the annotator's subjective bias during labeling is unavoidable. However, with the weak label, since what we need is only the presence information, it saves the effort for intensive labeling, and subjective bias is significantly suppressed, which means the scaling is easier as we can gather data quickly, especially when the data acquisition is easy nowadays while labeling is still far more

labor-some.

For a particular class in the weak-label classifier, all samples without instances of this class are considered negative samples. For instance, when learning a classifier to detect cars in an image, all images without cars can be simply viewed as negative samples. Note here the label is weak, it only tells a story that the image contains cars in it, but the exact location information is unknown. In this case, the rest of the parts except the car in the positive labeled image, which themselves are negative samples, could become the dominating factor of our classifier's prediction. To alleviate this situation, negative selecting strategy comes as a rescue, this is where negative sampling comes into place.

As we discussed above, we are interested in learning the decision boundary, which can best separate the positive and negative instances to build highly accurate weak label classifiers. To achieve this goal, we need a strategy to better select the negative samples, telling which of these negative samples are better or more informative for the current task. The strategy to be used here is called ***Negative Sampling***, which leads to the task of our project: Find an optimal sampling strategy for the weak-label classifier. We aim to optimize the decision boundary in weak label classifiers by selecting the most informative negative samples.

To address this problem, the following questions need to be answered:

- Whether some negative samples are more important than the others, if yes, is there an underlying criterion to quantify it;
- More specifically, whether negative data is similar to or different from the target object contributes more to the classification results and how to measure that;
- Can the data selection strategy (Negative Sampling) be modeled for different tasks using weak labels combined with negative sampling;
- To what proportion should the positive against negative instances enough to train a classifier.

We look to address the following: How to choose the reasonable sampling strategy? The sampling strategy we designed should identify the most informative negative samples that can help learn the best classifier with a reasonable label budget, which is also the core problem in active learning research field. That is why we introduce this topic in our project. Even though active learning aims to sample unlabeled data while only negative instances is our focus, the method will intrinsically select informative data samples that can improve the learning of the model and thus enhance the discriminative and

predictive ability of the model. In other words, the active learning algorithm will only select the data samples that are closest to the actual decision boundary disregarding irrelevant data samples from consideration, which will hopefully give us guidance on how to choose our negative samples.

7.2 Related work on Negative Sampling

7.2.1 Active Learning

Several sampling strategies have been proposed for active learning, focusing on identifying the most uncertain samples. For instance, [151] introduced an uncertainty-based active learning algorithm for positive and unlabeled data. This approach estimates the probability density of the positive and unlabeled points separately using Bayes' rule. Then it employs margin-based and entropy-based methods to measure informativeness and select the most uncertain samples for querying.

To enhance the quality of negative samples in tasks with high mutual information, [152] developed an active sampling strategy inspired by [153]. Their approach aims to select the most informative data for contrastive learning using a gradient-based method for uncertainty measurement, coupled with the solution [153] to ensure sample diversity. Additionally, [154] devised an active learning strategy in which the uncertainty and informativeness of data are assessed by a multi-model committee. This method prioritizes minority classes with low variance during sampling to address class imbalance in the training set. Similarly, [155] proposed a kernel machine committee model that integrates Bayesian (RVM) and discriminative (SVM) kernel machines for multi-class data sampling. This model effectively captures decision boundaries and can sample informative data for multi-class training.

Learning-based methods for sampling have also been explored. [156] introduced a learning-based active learning framework, where a sampling model and a boosting model iteratively learn from each other to improve performance. Their sampling model integrates uncertainty sampling and diversity sampling into a unified process for optimization, actively selecting the most representative and informative samples.

7.2.2 Weak Label Learning

[21] studied how characteristics like label density and label corruption needs to be considered for large scale audio or sound event detection (AED) problem, which basically means the portion of positive data in positive bags and influence of mislabeled data. They also proposed a CNN based weak label learning model for large scale AED. [157] introduced semi-weak labels as weak label with counting information, and developed a

two-stage framework for semi-weak label learning. It not only learns classification at the bag level but can also be extended to instance level classifier with counts assigned. However, semi-weak labels are not often available, especially for audio or video data.

7.2.3 Negative Sampling

For Knowledge Graph Embedding problem where high score negative samples are rare, [158] developed a more stable and efficient GAN-based method with caching. High-quality triplets are cached and sampled by the generator when training. The cache is updated with Importance Sampling method to adapt to its dynamically changed distribution. [159] studied the influence of negative sampling on objective and risk of embedding learning, and proposed a negative sampling strategy MCNS based on their theory, which approximates the positive distribution with self-contrast approximation and accelerated by Metropolis-Hastings. [160] incorporates the richness of graph structure and designed a structure-aware sampling strategy to sample semantic meaningful negative data for Knowledge Graphs.

7.3 Baseline Selection

To the best of our knowledge, importance of negative sampling in weak label learning is a novel research area as for now. As can be seen above, although many research and studies have been conducted in related fields, like weak label learning, active learning, etc., few of them have made a detailed discussion if we could achieve better or worse results by sampling different negative samples. So in this project, we first implement random sampling on negative data as a baseline score. Without loss of generality, we are going to conduct experiments on both image bag and weak label audio classification.

7.3.1 Image Bag Classification

Aiming at developing effective negative sampling strategy on more general form of weakly labeled data, we generate image bags from randomly selected images of CIFAR-10 such that instances in the same bag may possibly be irrelevant. We start from bags with fixed size 5, and uniformly distributed the label density, which denotes the proportion of positive data in positive bags, among all the bags. To alleviate the influence caused by single image being learned unevenly multiple times in a single epoch, each image only occurs in one bag.

Although the full dataset is unbalanced for a single class, its wide coverage and large size are important to the robustness of training. Any subset sampled from the original dataset is likely to be at a disadvantage. Therefore, results on full weakly

labeled dataset naturally become the ideal benchmark that we intend to approximate and even surpass. As for the baseline, a randomly sampled balanced subset is used for comparison and demonstration of other strategies' effectiveness. Each epoch, it will re-sample same amount of negative bags as all the positive ones.

For downstream classification problem, we pick a simple CNN model and a complex ResNet18 model to examine strategies' effectiveness on different models.

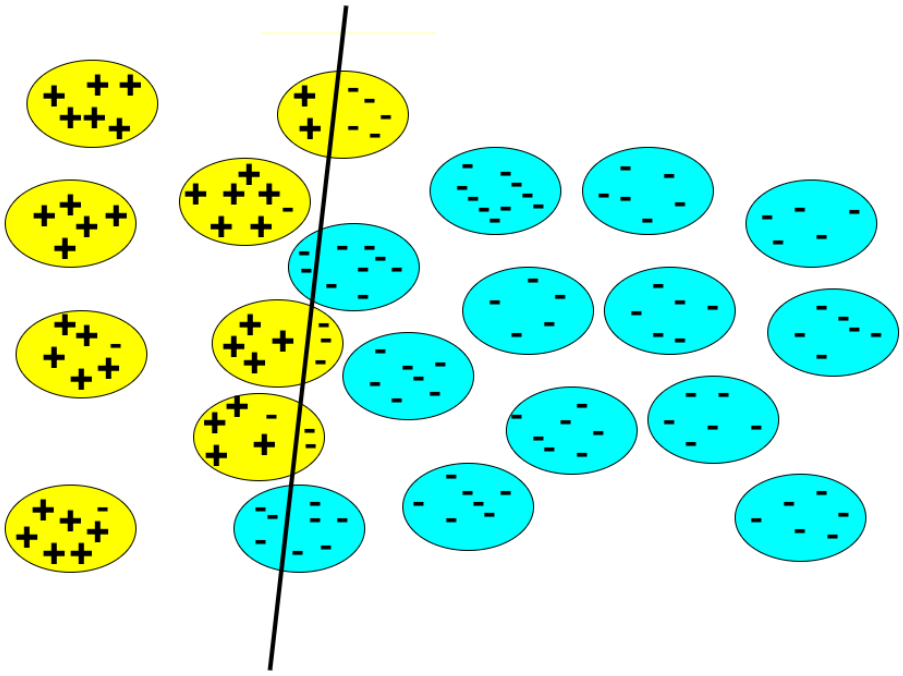


Figure 7.1: Learning Decision boundary showcasing importance of negative sampling

7.3.2 Audio Classification

AudioSet is a large weakly labeled and multi-class dataset consisting of 527 classes on sound events. The dataset is consists of 3 subsets, namely balanced training set, unbalanced training set and evaluation set. Considering the consumption of time and computing resource, we use the balanced training set as the training set and evaluation set for testing. The balanced training set contains 22k samples and the evaluation set contains 20k samples. However, some YouTube URLs are not available when we download audio clips from YouTube. 21515 audio clips for the balanced training set and 16322 samples for the evaluation set were downloaded in all, which is 97% and 80% of the original set respectively. For preprocessing, the audio clips are padded or truncated into 10 seconds waveform. Then we convert them into log Mel spectrograms, which are the acoustic features of the audio recordings. Hence, each feature input of $X \in R^{1056\times128}$ corresponds to the log Mel representation of a recording segment with

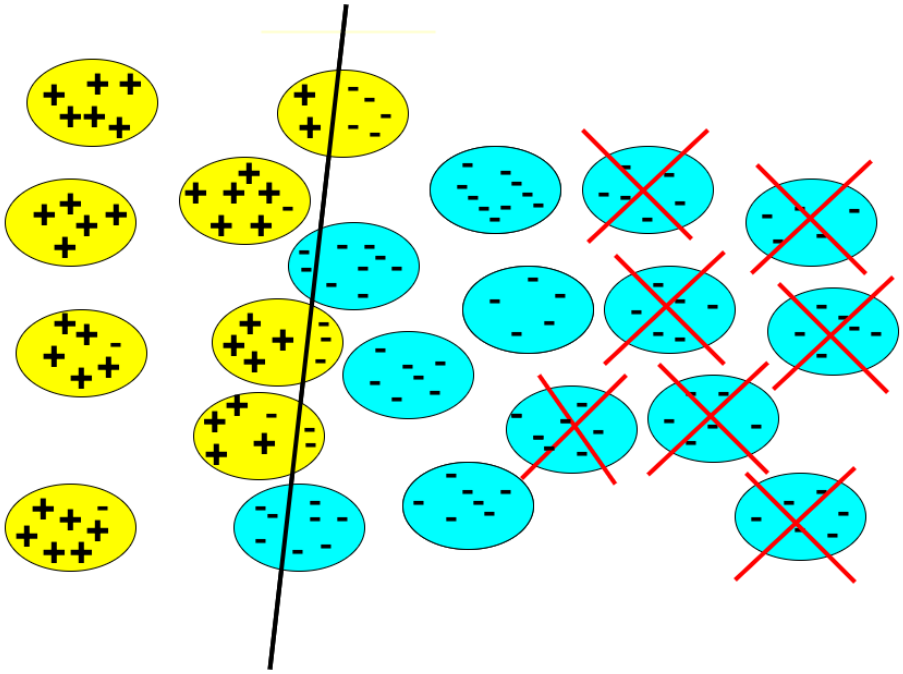


Figure 7.2: Figure showing the negative bags with minimal contribution towards learning the decision boundary

different lengths. All of the audio recordings are sampled with a sample rate of 44.1 kHz.

To form the list of training for the baseline, we define positive and negative data as follows: first, we name all the samples that contain audio segments from a pre-selected subset of classes as **Positive** and the left as **Negative**. For the experiment study, negative audio bags were sampled by different sampling strategies based on a specified Positive-Negative-ratio in each epoch of training. In our baseline experiment, we use a random sampling strategy, which means that each negative sample would be selected under the same probability.

We focus on a subset of 50 classes over the entire AudioSet. A thing to note is that we removed the first and second frequent class, which transformed the balanced train set into an imbalanced set with a large pool of negative bags to sample. The difference of $AudioSet_{subset50}$ is that it contains all the audio recordings but only contains labels in the subset of classes. Table 7.1 shows the top 5 classes information in AudioSet50.

"Music" and "Speech" classes have many positive data samples in the whole train and evaluation set. However, most of classes like "vehicle", "in small room" and "animal", whose positive samples only make up a little proportion to the whole train set, are imbalanced.

We choose PSLA [161], one of the state-of-the-art methods used in audio classification,

as our model structure for training and evaluation. Our implementation is based on the PyTorch framework. Hyper-parameters are tuned on our dataset according to [161].

We applied the same data augmentation as in paper [161]: MixUp and Time and Frequency masking to improve the performance of the classification on audio. The input spectrogram has the shape of (1056, 128). Frequency masking is applied to the frequency domain. The mask window is sampled from $f\ uniform(0, F)$ and the start frequency for masking is drawn from $f_0\ uniform(0, 128 - f)$. Time masking is applied to the time domain so that spectrogram in $[t_0, t_0 + t]$ frames are masked, where $t\ uniform(0, T)$ and $t_0\ uniform(0, 1056 - t)$. We use the same parameter $F = 48$ and $T = 192$ as [PSLA]. Mixup augmentation combines the waveform of 2 different training audio recordings x_i, x_j and their labels y_i, y_j:

$$x = \lambda x_i + (1 - \lambda)x_j y = \lambda y_i + (1 - \lambda)y_j \tag{7.1}$$

where λ is drawn from a beta distribution:

$$Beta(\alpha, \alpha) : prob(x; \alpha, \alpha) = \frac{x^{\alpha-1}(1-x)^{\alpha-1}}{B(\alpha, \alpha)}$$

We set $\alpha = 10$ and a mix-up rate of 0.5 to achieve better classification performance.

Table 7.1: Top 5 classes in AudioSet40

Name	*Mid*[1]	*Class number*	*Frequency of occurrence*
music	/m/04rlf	137	6244
speech	/m/09x0r	0	5735
vehicle	/m/07yv9	300	860
Inside, small room	/t/dd00125	506	764
Animal	/m/0jbk	72	734

7.4 Experiments Results

7.4.1 Image Bag Classification

Experiments for image bag classification are conducted with Python v3.8.11, PyTorch v1.9.0 and CUDA v11.1 on NVIDIA GeForce GTX 1070. For inference from output logits to labels, we tried 3 methods, softmax followed by mean, mean followed by softmax and softmax followed by max. As the results show that mean followed by softmax is the most effective one, we stick to this computation during following runs for consistency. It is reasonable that output of softmax is bounded from 0 to 1, if few instances in the bag are positive and have high probability outputs, the averaged

[1]In AudioSet, the labels are stored as integer indexes. *Mid* represents the mapping for each class.

probability of that class is still low, and can be even lower than class that has all hard-to-distinguish negative instances. What's more, we also tried out the influence of resample frequency on this task by comparing results of resampling every epoch and resampling every 3 epochs. However, no obvious relation can be observed between these two settings, and we adopt resampling every epoch for later experiments.

To better evaluate the effectiveness of sampling strategy and the robustness to different classes and seeds and postive-negative ratios, we consider several strategies from the literature, which are listed below. Each are implemented for sampling from pool of negative bags and getting a truncated training dataset.

1. **Random**: Randomly select k bags with uniform distribution at each epoch.

2. **Least Confidence**: An uncertainty-based strategy that samples k bags with lowest prediction confidence, which is defined as the difference between positive probability and negative probability.

3. **SVM-Margin**: An uncertainty-based strategy that samples k bags with lowest margin. A SVM model is trained first on full dataset and used to generate margin scores of bags.

4. **Entropy**: An uncertainty-based strategy that samples k bags with lowest entropy scores. The entropy score is computed as the entropy of predicted positive-negative distribution

5. **Gradient Embedding**: An uncertainty-based strategy that selects k bags that generates largest gradient in terms of L2 norm.

6. **BADGE**: A strategy combining uncertainty and diversity by picking bags that have gradient embedding as centers of k clusters in the pool. It selects the bag with the largest gradient embedding and use k-MEANS++ to find the rest ([153]).

7. **KL Divergence**: A similarity-based strategy that selects k negative bags that are closest to positive bags in terms of KL Divergence.

The first is a standard sampling method, the next four are uncertainty-based, the sixth is a mixture of uncertainty and diversity-based method, and the last is similarity-based. By randomly choosing the negative bags, we want to produce the same amount of negative bags as the sampled version of the negative bags since they will guarantee the same condition of total bags to be selected for the task, and this will produce the baseline for the sampling strategy. By comparing the final AP and ROC-AUC scores of

Table 7.2: Comparison between different models using sampling strategies

Method	*Sampling strategy*	*Class 0 AP*	*Class 0 AUC*	*Class 1 AP*	*Class 1 AUC*
CNN	Full Set	0.88	0.95	0.93	0.97
CNN	Random	0.83	0.93	0.87	0.95
CNN	Least Confidence	0.81	0.92	0.82	0.93
CNN	BADGE	0.85	0.94	0.87	0.95
CNN	Entropy	0.81	0.92	0.82	0.93
CNN	KL_embedding	0.83	0.93	0.89	0.95
CNN	KL_prob	0.87	0.95	0.88	0.95
CNN	Grad_embedding	0.83	0.93	0.84	0.94
CNN	SVM-Margin	0.87	0.95	0.87	0.95
ResNet	Full Set	NaN	0.01		
ResNet	Random	0.37	0.65		
ResNet	Least Confidence	0.23	0.25		

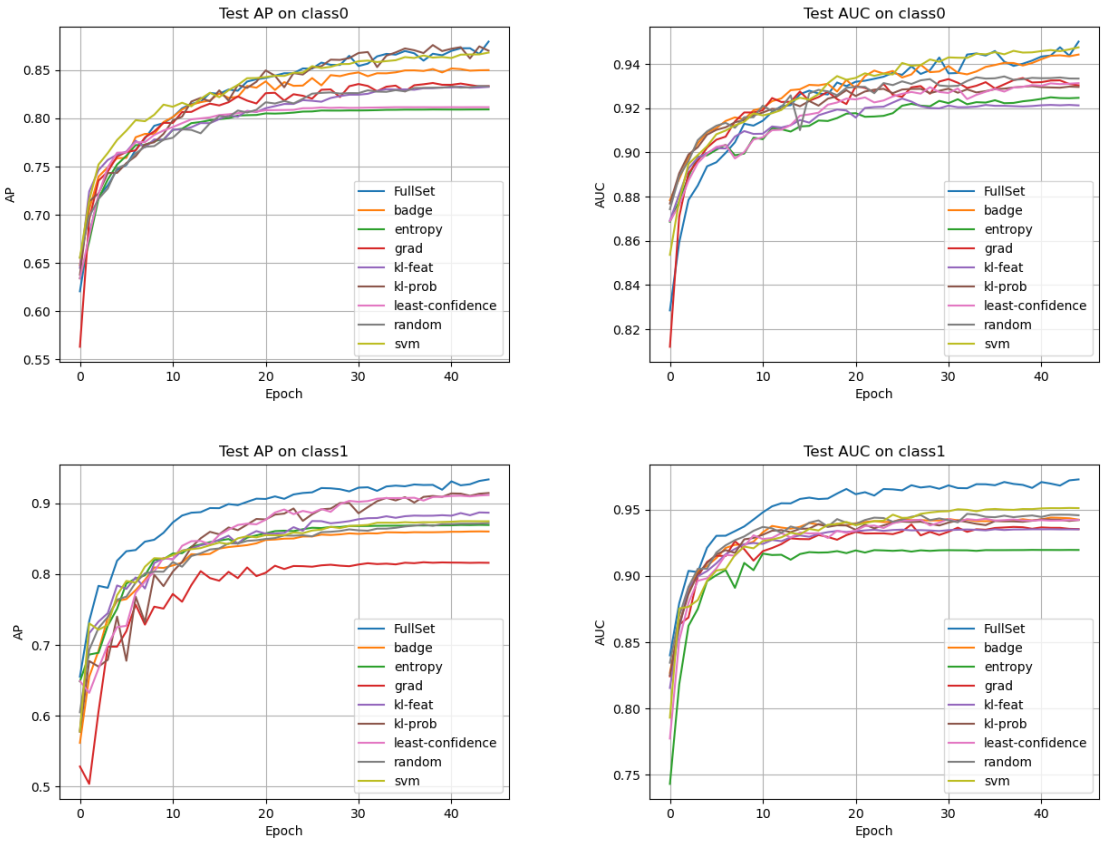


Figure 7.3: The left column shows the curve of AP, the right column shows the curve of ROC-AUC; Two rows from top to bottom represent class 0 and class 1 with seed 0, respectively.

the proposed sampling strategy and the random version, we can easily find whether the proposed sampling is effective or not.

Figure 7.3 and Table 7.2 shows the results of 2 classes. In the experiments, KL-Divergence method is implemented in two ways, one computes KLD based on the embedding extracted from the model, and the other uses output probability distribution. As we can see, CNN+fullset, which is our baseline, it has relative good performance for test data due to the large amount of data. The performance with other negative sampling strategies like Gradient Embedding are not consistently as good as the baseline. However, strategies like BADGE, SVM-Margin, KL-prob can beat Random most of the time, and achieve AP and AUC asymptotical to Full dataset.

7.4.2 Audio Classification

We run experiments using Python v3.8.8, Pytorch v1.10.0 and CUDA v11.4 on NVIDIA GeForce RTX 2080Ti. By comparing the result of multiple experiments, we decided to use the following parameters and training strategy: Most of the samples have length of 10 seconds so we pad or truncate the other audio clips to 10 seconds. We use batch size of 32 on the json of data files. Before each epoch of training, we sample the negative with the sampling strategies and keep all the positive samples. Then we will perform data augmentation on those samples and shuffle them. The optimizer we used in our experiments is Adam, with learning rate setting as $1e^{-3}$ for 30 epochs in total. Weight decay of $5e^{-7}$ is added to our model to alleviate overfitting. We adopt the parameters and strategies mentioned before for the following experiments.

7.5 Conclusion

We have experimented with several sampling strategies on the CIFAR10 and Audioset. It turns out that KL divergence of final output, SVM-Margin and BADGE will give a consistent better performance than random and quite similar to that of the full set. These strategies quantify the similarity of the positive bags and the negative ones then effectively select samples with more information, which leads to a better classification performance for the weak label.

In this chapter, we explore the importance of negative sampling in weak label learning, focusing on the selection of more informative negative samples to improve classification performance. Our experiments on CIFAR-10 and AudioSet demonstrated that negative sampling strategies can significantly enhance the performance of weak

[1]P-N ratio represents the ratio between the number of positive samples and negative samples, which is an important hyperparameter during training.

Table 7.3: Baseline model scores for Top 5 classes in AudioSet40

Class Name	*P-N Ratio*[1]	*AP*	*AUC*
Music_random	1.0	0.8600	0.9049
Music_margin	1.0	0.8520	0.8980
Music_Badge	1.0	0.8315	0.8899
Music_grad	1.0	0.8608	0.9040
Speech_random	1.0	0.7999	0.8919
Speech_margin	1.0	0.8256	0.8990
Speech_Badge	1.0	0.8040	0.8964
Speech_grad	1.0	0.8140	0.8939
Vehicle_random	0.1	0.4295	0.9066
Vehicle_margin	0.1	0.4172	0.9005
Vehicle_Badge	0.1	0.4262	0.9083
Vehicle_grad	0.1	0.4206	0.9089
In Small Room_random	0.1	0.1420	0.7807
In Small Room_margin	0.1	0.1310	0.7585
In Small Room_Badge	0.1	0.1353	0.7722
In Small Room_grad	0.1	0.1560	0.7914
Animal_random	0.1	0.2410	0.8263
Animal_margin	0.1	0.2523	0.8368
Animal_Badge	0.1	0.2983	0.8522
Animal_grad	0.1	0.2689	0.7943

label classifiers. By selectively removing less important negative data, we were able to maintain high AP and AUC scores even when using a reduced subset of the dataset. We highlighted the advantages of weak labels, which require only the presence information of an event, making data acquisition and scaling easier compared to strong labels. This approach reduces the effort and subjective bias associated with intensive labeling.

Our literature review covered various sampling strategies in active learning, weak label learning, and negative sampling, providing a comprehensive background for our study. We implemented random sampling as a baseline and conducted experiments on image bag classification using CIFAR-10 and audio classification using AudioSet.

For image bag classification, we generated image bags from CIFAR-10 and examined the effectiveness of different sampling strategies using simple CNN and ResNet18 models. For audio classification, we used a subset of AudioSet and applied state-of-the-art methods like PSLA, along with data augmentation techniques such as MixUp and Time and Frequency masking.

In summary, our results indicate that negative sampling plays a pivotal role in weak label learning. Developing efficient sampling strategies can enhance classification performance while minimizing data needs. This direction of paves the way for additional investigation in the areas of weak label learning and negative sampling.

8

Learning with Ontologies

Ontologies play a crucial role in the formal representation of knowledge by defining the concepts and properties within a specific domain, as well as the relationships between these concepts. They provide a structured framework that facilitates the organization, sharing, and reuse of knowledge across various applications. Prominent ontologies such as WordNet and AudioSet exemplify the state-of-the-art in capturing domain-specific information, offering comprehensive vocabularies and hierarchical relationships that enhance the understanding and processing of data.

The significance of ontologies extends to numerous fields, including natural language processing, image classification, and knowledge graph completion. Using the rich semantic information embedded within ontologies, previous studies have demonstrated improvements in model performance, particularly in tasks that require a deep understanding of domain-specific concepts and their interrelations.

In this chapter, we explore the potential of using ontological information to enhance learning from weakly labeled data. Weakly labeled data, which only require the presence or absence of an event, are advantageous due to their relative ease of collection compared to strongly labeled data. Specifically, we employ the AudioSet ontology and dataset, which comprises audio clips weakly labeled with ontology concepts and incorporates "Is A" relationships between these concepts.

Our approach begins with a reimplementation of the model proposed by [Sound Event

Ontology Reference], adapted to accommodate a multilabel scenario. Building upon this foundation, we introduce a Graph Convolutional Network (GCN) to effectively model the ontology information, enabling more nuanced learning of the underlying concepts. Through our investigations, we discover that while the baseline Siamese network does not exhibit improved performance when incorporating ontology information in a weak and multilabel context, the GCN approach successfully captures the ontology knowledge, enhancing performance under these conditions.

Furthermore, we examine the resilience of different modules to noise introduced by weak labels and their ability to integrate ontology information more effectively. Our most effective model, the Siamese-GCN hybrid, achieves a mean Average Precision (mAP) of 0.45 and an Area Under the Curve (AUC) of 0.87 for lower-level concepts, and a mAP of 0.72 and an AUC of 0.86 for higher-level concepts. These results indicate a notable improvement over the baseline Siamese network, although the gains are comparable to models that do not use ontology information.

This chapter underscores the importance of ontologies in enhancing machine learning models, particularly in scenarios involving weakly labeled and multi-label data. By integrating structured domain knowledge through advanced techniques such as graph-convolutional networks, we demonstrate the potential for more robust and semantically informed learning processes.

8.1 Importance of Utilizing Ontologies

Ontologies offer structured, hierarchical representations of domain knowledge through defined concepts and the relationships among them. They enable a machine to generalize from unknown or unfamiliar instances to broader, more recognizable categories. For example, a listener may not identify the exact animal making a particular sound, yet still recognize it as an "animal sound." In the same way, a machine can leverage ontological information to classify an object it has never explicitly encountered as belonging to a higher-level class. Incorporating these hierarchical relations from ontologies can help refine model predictions, especially when faced with semantically similar observations or ambiguous subclasses. Previous works have explored using such external knowledge both to guide feature extraction [162, 163] and to integrate this structured information directly into model architectures [164, 165, 166, 167].

This chapter investigates whether learning ontological classes from weakly labeled data can be improved by incorporating such external knowledge. Consider an audio clip containing *dog howling*, *baby crying*, and *engine idling*, yet only weakly labeled as containing *dog howling*. Using ontological information, the model might infer that the sound of a howling dog implies broader categories like *animal sounds* or *living*

things. Even if the clip also featured a *cat meowing* but remained unlabeled as such, the ontology could guide the model toward identifying the higher-level class *animal sounds* or *living things*, despite incomplete labeling. Thus, ontological knowledge can help the model interpret hidden cues in the data, improving classification performance for both annotated and unannotated but related categories.

The appeal of weakly labeled datasets lies in their scalability and reduced annotation cost, as they require only the presence or absence of an event to be known. Google's AudioSet, for instance, employs weak labels at massive scale [168]. However, these labels often introduce noise that complicates model training. Ontologies may help mitigate this issue by furnishing structured domain insights that strengthen a model's interpretive capabilities, even under noisy conditions.

This chapter expands on prior research into ontology-guided classification, hierarchical learning, and knowledge graph utilization. We implement and evaluate two models that incorporate ontology information to predict classes and hierarchies from weak labels. Through our experiments, we aim to gain a clearer understanding of how ontological embeddings affect performance in weakly supervised scenarios and to assess the strengths and limitations of these approaches. The following sections discuss related literature, detail the methods and models employed, and present an in-depth analysis of our results and conclusions.

8.2 Methods

8.2.1 Dataset and Ontology

We use the AudioSet dataset, which consists of an ontology of weakly labeled 632 audio event classes and 2,084,320 human-labeled 10-second sound clips drawn from YouTube videos. The 128-dimensional Audioset features and ontology files are made available by Google and accessed online from previous work [21], [168]. Feature embeddings provided by [168] are extracted from a VGGish model [169] followed by a PCA to reduce down to 128 dimensions and each ten second clip is represented by ten 128-dimension feature vectors, i.e. one feature vector represents one second of one clip. Each clip in the dataset can contain mutiple labels where the clsses encompass a wide variety of sounds like human noises, music, animals or vehicles.

AudioSet is an audio dataset that consists of 2,084,320 10-second recordings, each with feature encodings and an event label. Each 10-second recording has ten 1-second segments and ten 128-dimensional feature encoding. For each 1-second segment of the 10-second clip, there is a corresponding 128 dimension embedded representations. Each clip in the dataset can contain mutiple labels where the clsses encompass a wide variety

of sounds like human noises, music, animals or vehicles. AudioSet is made available through [21] and [168], whose feature embedding is derived using [168] with methods from [169].

We use balanced training set of Audioset for training our models, containing about 22,160 ten second clips. We then preprocess the corresponding labels of each audio clip to get the labels in the top two levels of the Audioset ontology. The validation set is about 20% of the training set size and is drawn from the unbalanced training set of Audioset, and the Audioset test set contains about 20,383 audio clips. The same label processing is done for the validation and test set to obtain two levels of ontology labels. Although there are more levels to the Audioset ontology, we choose to focus on the top two levels to fit our model architectures. "" To give a more general idea of the Audioset data, consider a weakly labeled audio clip which could have multiple labels with no timing information, but the labels could be related due to their ontology. For example, a human voice and hands could possibly co-occur because they are both natural sounds. The labels could also be related due to their dependency, such as a human crying could co-occur with sad music. On the other hand the labels could also not be related with each other. For instance, a dog barking sound and car engine sound could co-occur in a clip accidentally.

8.2.2 Framework

We consider multi-labeled training data $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)\}$ where $\mathbf{x}_i \in \mathcal{X}$ is a single 128-dimension audio feature representation and the corresponding $\mathbf{y}_i$ is a collection of labels $\{(y_1^1, y_1^2, ..., y_1^{T_1}), ..., (y_k^1, y_k^2, ..., y_k^{T_k})\}$ with k equal to the number of ontology levels and T_i equal to the number of labels for $\mathbf{x}$ at the i^{th} ontology level. Thus in our post-processed dataset we have $k = 2$ and will refer to these as subclass or "level 1" labels and superclass or "level 2" labels.

8.2.3 MLP Network without Ontology Information

To investigate whether incorporating the ontology information is beneficial to use with weakly labeled data, we first need to train a model which uses no ontology information. We implement a simple MLP network with three hidden layers and a final output layer that collectively predicts ontology classes. More concretely, the final layer has size $\sum_{i=1}^{k} T_i$ to predict all ontology labels. Each hidden layer is of size 512 followed by a BatchNorm layer, ReLU activation, and a Dropout layer. This model makes no use of the ontology information to aid in the prediction of the ontology labels, making it a good baseline to compare to our models which will incorporate the ontology information.

The final outputs pass through a final sigmoid activation for the multi-label scenario and the model aims to minimize the binary cross entropy loss.

8.2.4 Twin Neural Network Model with Ontology-based embeddings and Ontological layer

The ontology based model from [170] is a recent contribution that specifically focuses on audio data while also providing a method for prediction of classes at different ontology levels. Much of the prior work on ontology prediction has been in different areas such as medicine (drug classification) or text classification, whereas [170] presents a framework that is more relevant and recent. This model will therefore serve as a baseline for the models which will incorporate ontology information. The framework with modification for multi-label scenarios is shown in Figure 8.1.

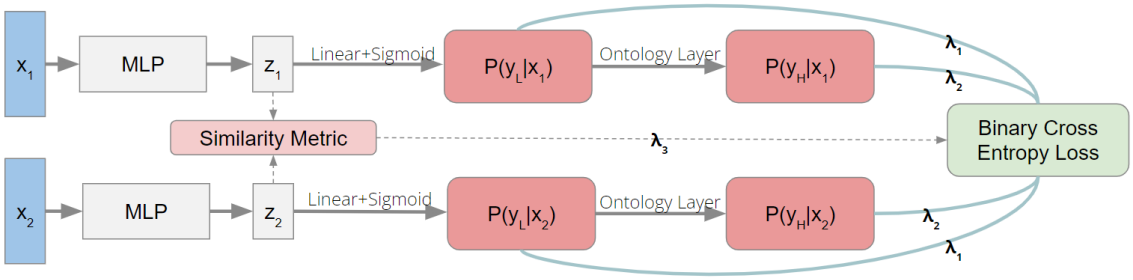


Figure 8.1: Architecture of Twin Neural Network + Ontological Layer with modification to fit the multi-label task

Now we present the components of the model and its mathematical formulation, which is an extension of the work in [170] to learn ontology-based embeddings for classification of multi-labeled data. We use a Siamese neural network (SNN) to separate the ontology-based embeddings by imposing the embedding distance close to 0 if the input pairs are from the same subclass, close to 1 if they are from different sub classes, but the same super class, and close to 2 if they are from different super classes. To fit in the multi-label scenario, we applied two kinds of sampling methods, the first one is that samples fall into the same sub (super) class category if and only if they have exactly the same subclasses; samples fall into different sub (super) class category if any of their sub (super) classes is different. The second approach is that samples fall into the same sub (super) class category if and only if there is any sub (super) class in their intersection; samples fall into different sub (super) class category if there is no any sub (super) class in their intersection.

The base model architecture of the Siamese net is the same MLP with three hidden layers from section 3.3, and each branch of the Siamese net shares the MLP parameters. Given a pair of input vectors $\mathbf{x_1}, \mathbf{x_2}$, the output of the MLP is an ontology embedding $\mathbf{z_1} = f(\mathbf{x_1}), \mathbf{z_2} = f(\mathbf{x_2})$. The embedding vectors $\mathbf{z_1}, \mathbf{z_2}$ produce subclass probabilities after a sigmoid activation, which is used for the prediction of multi-labeled data.

The ontological layer, $\mathbf{M}$, then relates the class probabilities of level 1 to those in level 2 of the ontology through the following relation:

$$\mathbf{p}(y_2|\mathbf{x}) = \mathbf{M} \cdot \mathbf{p}(y_1|\mathbf{x}) \tag{8.1}$$

We modify the ontological layer proposed in [170] in order to fit the multi-label data scenario, where $\mathbf{M}$ is constructed such that it averages the probabilities of the subclasses within a single superclass. Note that this layer is fixed and not trainable, it depends strictly on the ontology of the training data.

The final loss function incorporates the binary cross entropy loss of the level 1 classes, $\mathcal{L}_1$, and the level 2 classes, $\mathcal{L}_2$, with the embedding loss, $D_w = (||\mathbf{z_1} - \mathbf{z_2}||_2 - d)^2$ where $d \in \{0, 1, 2\}$ according to the type of input pair.

$$\mathcal{L} = \lambda_1(\mathcal{L}_1^1 + \mathcal{L}_1^2) + \lambda_2(\mathcal{L}_2^1 + \mathcal{L}_2^2) + \lambda_3 D_w \tag{8.2}$$

The base network takes an input feature of dimension 128 before the output layer to the 42 classes in the first ontology level and then through the ontology layer to the 7 classes in the second level. The baseline paper does not actually report evaluation metrics for the Audioset data, as they focused on other audio datasets: Urban Sounds - US8K and data from the Making Sense of Sounds Challenge. These are both single labeled datasets.

8.2.5 Graph Convolutional Network

To model both the co-occurrence information introduced from weak labels and the domain knowledge from ontology, we seek to utilize graph embedding approaches to extend the Siamese network architecture. In [171] and [172], a Graph Convolution Network (GCN) is shown to be effective for learning useful node representations through the information in a correlation graph. The essential idea is to update the node representations by aggregating information from neighboring nodes.

Following the ideas of previous work, [171, 172, 173] , we define the subclass labels and superclass labels as the nodes of the graph and aim to learn the label representation. The knowledge in the graph is encoded as a correlation matrix, which is a crucial part of the GCN. We will describe how it is constructed in Section 8.2.5.1.

We use one-hot encoding of labels as the initial node representation and use 2 GCN layers to extract embeddings with neighboring information. For each GCN layer we use 2 linear layers to transform the input embedding of the node itself and a graph

convolution to aggregate the embeddings from its neighbor. Given a label embedding $Z \in \mathbb{R}^{C \times d}$ (where C is the number of nodes and d is the dimensionality of node features), the graph convolution operations is:

$$Z^{l+1} = h(A' Z^l W_1^l W_2^l) \tag{8.3}$$

where $W_1^l \in R^{d \times d'}$ and $W2^l \in R^{d' \times d''}$ are 2 transformation matrices to be learned and $A' \in R^{C \times C}$ is the correlation matrix and $h(.)$ is a non-linear operation which is a LeakyReLU in our experiments. The dimensions of each the linear layers are 280 and 512 for the first GCN layer and 320 and 128 for the second GCN layer.

8.2.5.1 Correlation Matrix

The GCN learns node representations by collecting information from other nodes based on the correlation matrix provided. Thus, how we build the correlation matrix is crucial but also challenging for GCN. In this work, we referred to previous work [171, 172] and experimented on 3 different correlation matrices.

Labels Co-occurrence based Correlation Matrix: [172] proposed a way to model label dependency in the form of conditional probability, i.e. $P(L_j \mid L_i)$ denotes the probability of occurrence of L_j when L_i is present. To construct the correlation matrix, we count the co-occurrence of label pairs present in training set to get a matrix $M \in R^{C \times C}$, where $M_{i,j}$ denotes the co-occurrence time of L_i and L_j. We then divide M by the occurrence time of L_i in training set to get the conditional probability P. To prevent a long-tail distribution where some rare co-occurrences may be noisy, we binarize P by setting a tunable threshold t. The correlation matrix A is set to 1 if P is above the t and set to 0 when P is below the t, where $A \in R^{C \times C}$.

Ontology-based Method One: [171] proposed the correlation matrix A to denote label pairs who have same parents. $A_{i,j} = 1$ when L_i and L_j have same parents; $A_{i,j} = 0$ otherwise.

Ontology-based Method Two: [171] proposed the correlation matrix A to denote label pairs who have edges in between them. In the dataset we are using, edges only occur between parents and children, so we set $A_{i,j} = 1$ when L_i is a child of L_j or the other way around; otherwise, $A_{i,j} = 0$.

To prevent over-smoothing, after binarizing, we re-weight A to get the desired correlation matrix A' by setting $A'_{i,j} = p$ when $i = j$ and $A'_{i,j} = (1-p)/\sum_j A_j$ when $A_{i,j} = 1$.

8.2.6 Siamese Network with Graph Convolutional Network

The Graph Convolution Network module described in Section 8.2.5 can convert the label embedding from a one-hot representation to an embedding aggregating neighbor

information. We use a matrix multiplication to get the similarity between a given audio clip embedding and every label's embedding. Given these similarity values, we then use a SoftMax activation and Binary Cross Entropy loss to calculate the loss between our outputs and the target labels. The overall framework is shown in Figure 8.2.

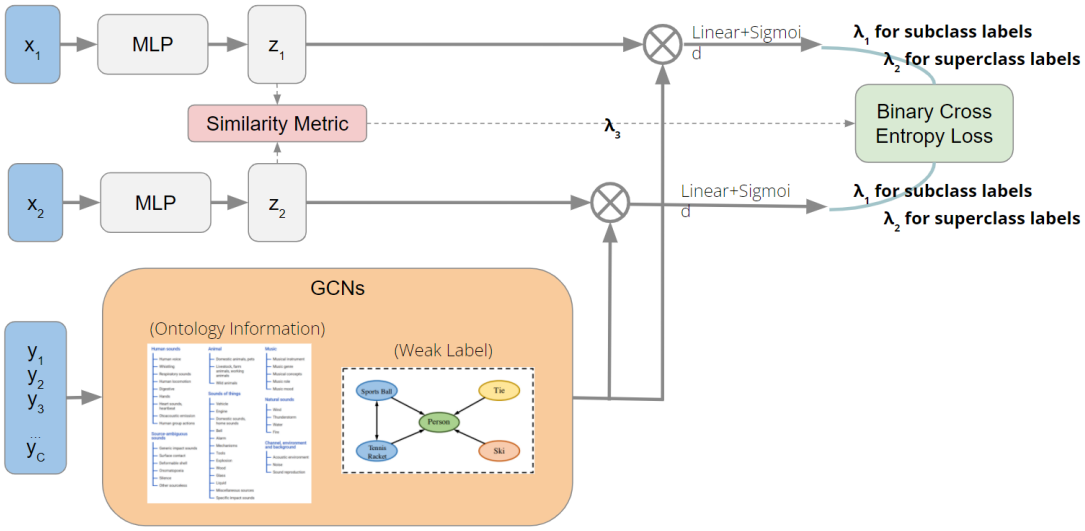


Figure 8.2: The framework of Siamese Network + GCN

Here we replace the baseline Ontological Layer with the label embeddings that the GCN generates. To combine it with Siamese Network, we still keep the λ_1, λ_2 and λ_3 for the loss term. Next we will discuss the experiments in more detail and the results from the various models.

8.3 Experiments

8.3.1 Evaluation Metrics

The performance of our models is evaluated using weighted average precision and AUC from predictions. All reported metrics are on the test set of the Audioset dataset. As discussed previously, we extract the labels for each segment from the top two levels of the Audioset ontology. The average precision (AP) metric is first computed for each class by considering the precision (fraction of true positive labels out of all predicted positive labels) - recall (fraction of true positive labels out of actual positive labels) curve at different thresholds. Thus it is an indication of how well the model can identify positive classes in the data. The other metric we consider is AUC, which considers negative labels by computing the area under the TPR (true positive rate) - FPR (false positive rate) curve and is an indication of how well the model can distinguish between classes. The AP/AUC metric for the classes on the same ontology level is combined

through a weighted average based on the proportion of each class that is present in the training data to get a final weighted AP and weighted AUC score for each level.

8.3.2 Performance of MLP Model with no Ontology Information

The MLP model with no ontology information achieves a weighted AP/AUC score of 0.45/0.87, respectively for subclasses. The weighted AP/AUC score for the superclasses are 0.71/0.86, as shown in Table 8.1. The model is trained for about 70 epochs at a learning rate of 2e-3. Please refer to the Appendix for more hyperparameter training details for each of the models.

8.3.3 Performance of Siamese Network with Ontological Layer

The Siamese model with Ontology layer achieves a subclass weighted AP/AUC = 0.36/0.81 and superclass weighted AP/AUC = 0.39/0.65. After hyperparameter tuning of the loss function, we were able to achieve results in Table 8.1 with $\lambda_1 = 1.5, \lambda_2 = 1, \lambda_3 = 0.25$. The experiments found that these metrics can be tuned based on the λ values in the loss function to control the contributions from each level of the ontology. A more detailed table of the hyperparameter tuning is presented in the appendix, and the results in the summary table are chosen as a good balance between the two ontology levels.

This experiment shows there should be more research work to apply a Siamese network to a multi-label scenario, such as how to generate pairs of the same sub/superclass as well as their similarity metric. We further investigated this problem by considering two definitions: one in which a pair of data is of the "same" subclass if they have exactly the same labels and another in which "same" means that two pairs just have some intersection in their labels. Our experiments suggest that the latter definition is better for the multi-label scenario. However, the performance of this baseline Siamese model which uses ontology information suggests that this external knowledge is still difficult to embed into a model for weakly labeled data. Furthermore, we believe that this definition of "same" or different" sub/superclass is still ambiguous and introduces even more noise into the model as it attempts to cluster pairs that are not really the "same" subclass. This can explain why we see such poor performance metrics compared to the MLP with no ontology information.

8.3.4 Performance of Siamese-GCN Model

The extension of the Siamese model replaces the Ontology layer with a Graph Convolution network to embed ontology information. We trained the model by Adam optimizer for about 30 epochs. We use 2 Linear Layer as the node embedding of GCN and apply

Model	Weighted AP		Weighted AUC	
	Subclass	Superclass	Subclass	Superclass
MLP	0.4509	0.7056	0.8706	0.8556
Siam. + Ont.	0.3653	0.3876	0.8055	0.6505
Siam. + GCN	0.4285	0.6790	0.8460	0.8280
MLP + GCN	0.4590	0.7117	0.8751	0.8602

Table 8.1: mAP and AUC Results of different models. Siam. = Siamese, Ont. = Ontology.

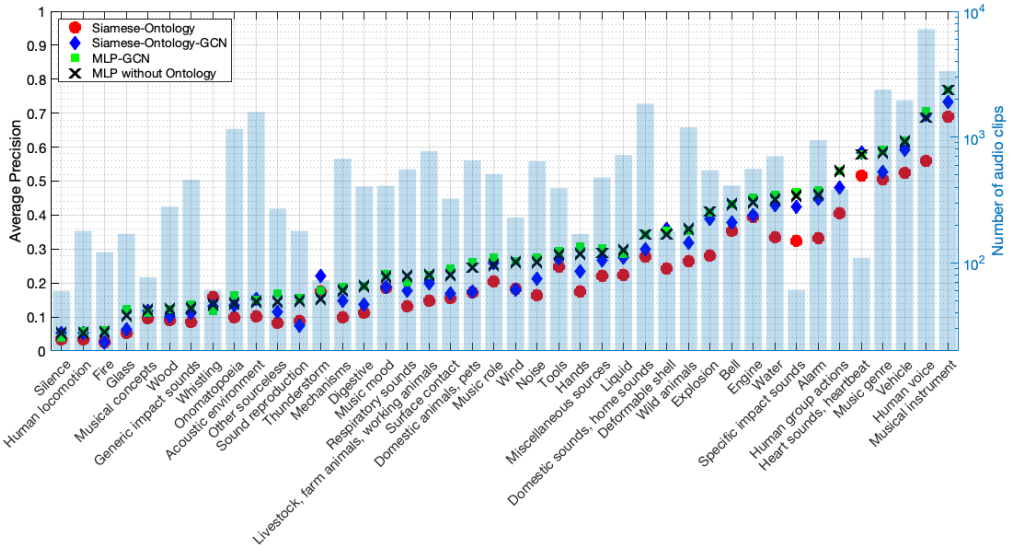


Figure 8.3: mAP across different low-level labels

2 Layer GCN. The performance is sensitive to the hyper-parameters λ_1, λ_2 and λ_3 of the loss function because it would amplify or reduce learning rate for different loss terms. We find the observation is that the greater λ is not necessary equal to the greater learning rate.

We also can see that the Siamese network with GCN could tolerate noises introduced from weakly labelled data better than the baseline Siamese with an un-trainable Ontological Layer. This suggests that the GCN can better capture the ontology hierarchy and that it can even overcome the noise introduced by attempting to find pairs for the Siamese net. Overall, the evaluation metrics for this model are close to the baseline MLP with no ontology information.

8.3.5 Performance of MLP-GCN Model

To study the utility of the Siamese net we also implement an MLP-GCN using the same GCN structure as the one in the Siamese GCN. We investigate which correlation

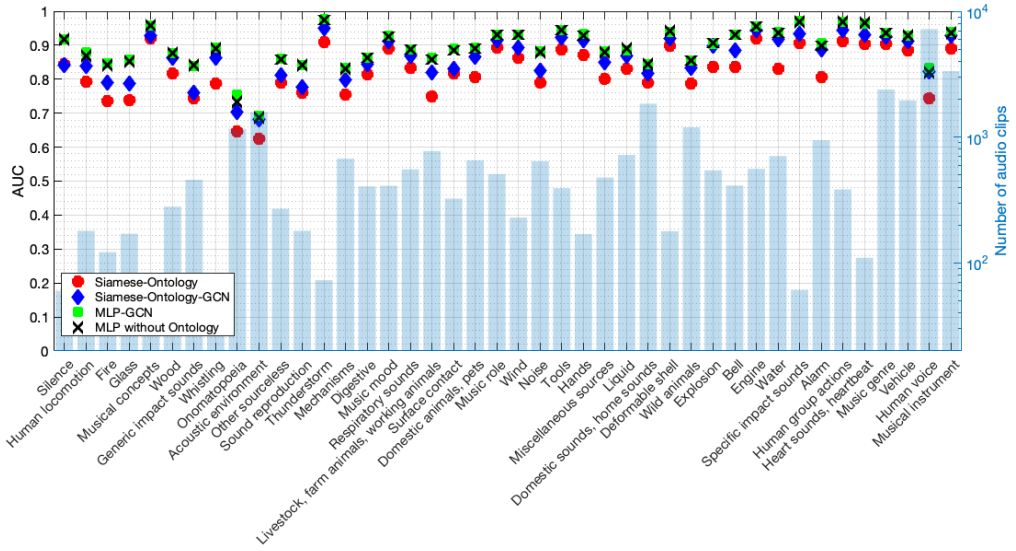


Figure 8.4: AUC across different low-level labels

matrices can provide the best improvement in performance. Among the 3 correlation matrices, we found label-co-occurrence based correlation works best to model ontology information in the weak label scenario. Consequently, we did more experiments on the two trainable parameters: t, p.

From our experiments we found $p = 0.2, t = 0.08$ achieved the best results. Also although this model performs better than the previous models, the improvement w.r.t an MLP without an Ontological Layer is still limited. This suggests that even with the ontology information embedded into the GCN, it is still hard to get significant improvements in classification results. It further demonstrates that the architecture of the Siamese net is not a great fit for the multi-label scenario. Using just a simple MLP to learn embeddings, it can already achieve relatively good performance with the GCN. Overall, we found that there were certain classes for which it was always difficult to achieve high AP or AUC scores as demonstrated in Figures 8.3, 8.4, 8.5. For these classes, e.g. glass, fire, or silence, we analyzed which data points contain those classes and found that they are often multi-labeled with more commonly found classes in the training set, such as human voice, or domestic sounds. This could make it difficult for the model to learn different representations for those classes.

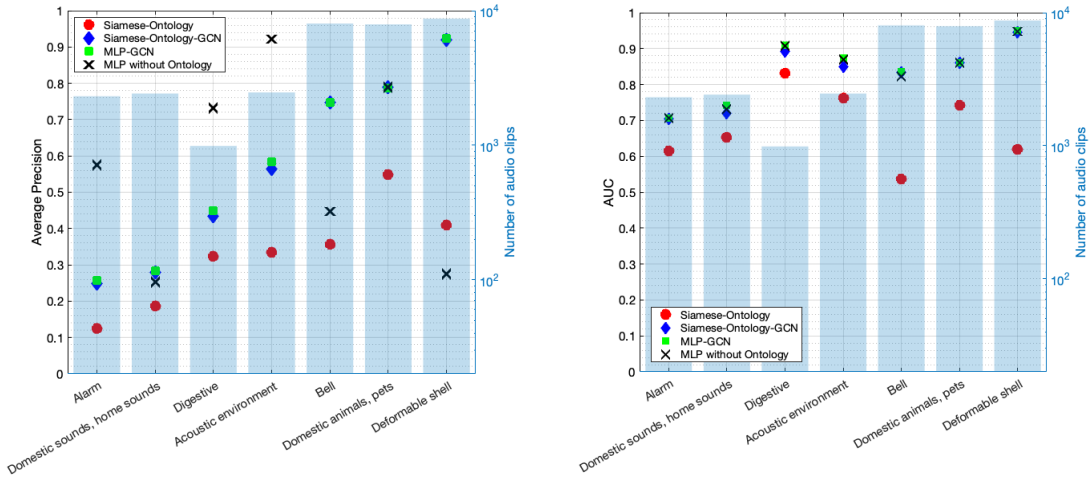


Figure 8.5: AP (left) and AUC (right) across different superclass labels

8.4 Hyperparameters

8.4.1 MLP Model with no Ontology Information

The simple MLP model is trained using the Adam optimizer with a learning rate=$2e-3$ and weight decay=$1e-4$, trained for around 73 epochs.

8.4.2 Siamese with Ontology Layer

The following results are for the Siamese network with Ontology layer.

λ_1	λ_2	λ_3	Low Level mAP	High Level mAP	Low Level AUC	High Level AUC
1.5	1	0.25	0.3379	0.4470	0.7904	0.7061
1.75	1	0.25	0.3480	0.4357	0.7941	0.6959
2	1	0.25	0.3511	0.4175	0.7969	0.6799
2	0.75	0.5	0.3653	0.3876	0.8055	0.6505
2	1	0.5	0.3586	0.4196	0.8004	0.6845

Table 8.2: mAP and AUC Results of MLP-GCN with different p parameters when $t = 0.168$

8.4.3 Siamese-GCN

λ_1	λ_2	λ_3	lr	Low Level mAP	High Level mAP	λ_1	λ_2	λ_3	lr	Low Level AUC	High Level AUC
20	2	0.5	$1e-3$	0.34	0.625	20	2	0.5	$1e-3$	0.796	0.788
100	100	0.5	$1e-3$	0.4285	0.679	100	100	0.5	$1e-3$	0.846	0.828
1	1	0.005	$1e-1$	0.306	0.615	1	1	0.005	$1e-1$	0.797	0.772

Table 8.3: mAP and AUC results of Siamese GCN with different λ and learning rate (lr)

8.4.4 MLP-GCN

p	Low Level mAP	High Level mAP	Low Level AUC	High Level AUC
0.1	0.4477	0.7091	0.8571	0.8690
0.2	0.4469	0.7098	0.8693	0.8575
0.3	0.4467	0.7083	0.8568	0.8690

Table 8.4: mAP and AUC Results of MLP-GCN with different p parameters when $t = 0.168$

t	Low Level mAP	High Level mAP	Low Level AUC	High Level AUC
0.010	0.4455	0.7072	0.8687	0.8566
0.025	0.4479	0.7077	0.8694	0.8567
0.080	0.4501	0.7141	0.8710	0.8613
0.100	0.4471	0.7128	0.8696	0.8597
0.150	0.4462	0.7093	0.8691	0.8577

Table 8.5: mAP and AUC Results of MLP-GCN with different t parameters when $p = 0.2$

8.4.5 Overall Observations

Our experiments highlight the importance of careful hyperparameter selection: factors such as λ-values, learning rates, and the parameters p and t all significantly influence performance. Identifying the optimal settings for these parameters is crucial to achieving stable and robust results.

From a model comparison perspective, the Siamese-GCN configuration—when appropriately tuned—generally outperformed the Siamese network equipped with a static Ontology Layer for lower-level concepts, improving both mAP and AUC. Moreover, the MLP-GCN model demonstrated competitive performance, especially at higher ontology levels, where its results rivaled or exceeded those of the more complex architectures. This indicates that, while intricate models can yield gains in certain scenarios, simpler frameworks enriched with graph-based label embeddings may also offer substantial benefits.

8.5 Conclusions

To conclude, we observed that the Siamese model with ontology layer has the worst performance, while a simple MLP obtains better results, and the combination of MLP with GCN works even better. It is possible for the Siamese architecture to have a negative impact on prediction, introducing confusion through the construction of input pairs for

data that is weakly multi-labeled. Although the GCN provides slight improvement, it appears the model still has difficulty in differentiating ambiguous subclasses or using the hidden knowledge in weakly labeled data. The way to incorporate ontology information may differ with context (dataset, ontological architecture, network structure, etc.); additional investigation is needed to determine the best ways to use it. For a dataset like AudioSet, where each instance has multiple, or even wrong labels, having a simple ontology layer at the output of a Siamese network might not be a good choice. So the next step is to narrow the scope of dataset and try different approaches to embed ontological information and approaches that could also make use of deeper levels of ontology information.

Section V: Unified Formalism

This section presents a unified formalism that synthesizes and generalizes a wide range of imprecise labeling problems into a single, coherent framework. By modeling true labels as latent variables and leveraging the expectation-maximization (EM) principle, this formalism establishes a common ground for integrating partial labels, noisy labels, and unlabeled instances—all of which have traditionally been addressed by separate, specialized methods. This approach is the first of its kind: rather than relying on ad-hoc designs tailored to each type of imprecision, it offers a principled, theoretically grounded solution that can accommodate and seamlessly adapt to any form of label uncertainty.

The following chapter will delve into this unified viewpoint, detailing how the proposed framework surpasses existing solutions and achieves robust, effective performance across a wide range of challenging imprecise label scenarios. This endeavor represents the culmination of our work, setting the stage for future research that can build upon this versatile and comprehensive foundation.

9

Imprecise Label Learning

In this chapter, we propose a unified formalism for that we aim to tie together the following tasks. The unified formalism will be completing the task of

Learning with reduced labeling standards, such as noisy label, partial label, and supplementary unlabeled data, which we generically refer to as *imprecise* label, is a commonplace challenge in machine learning tasks. Previous methods tend to propose specific designs for every emerging imprecise label configuration, which is usually unsustainable when multiple configurations of imprecision coexist. In this paper, we introduce imprecise label learning (ILL), a framework for the unification of learning with various imprecise label configurations. ILL leverages expectation-maximization (EM) for modeling the imprecise label information, treating the precise labels as latent variables. Instead of approximating the correct labels for training, it considers the entire distribution of all possible labeling entailed by the imprecise information. We demonstrate that ILL can seamlessly adapt to partial label learning, semi-supervised learning, noisy label learning, and, more importantly, a mixture of these settings, with closed-form learning objectives derived from the unified EM modeling. Notably, ILL surpasses the existing specified techniques for handling imprecise labels, marking the first practical and unified framework with robust and effective performance across various challenging settings. We hope our work will inspire further research on this topic, unleashing the full potential of ILL in wider scenarios where precise labels are

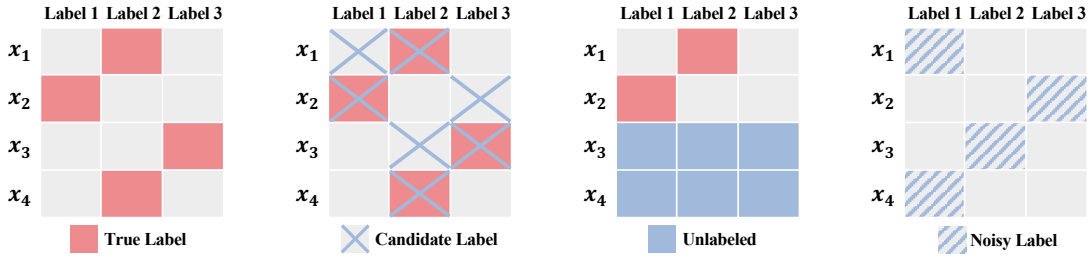
expensive and complicated to obtain.

9.1 Introduction

One of the critical challenges in machine learning is the collection of annotated data for model training [174, 175, 176, 177, 178, 179]. Ideally, every data instance would be fully annotated with precise labels. However, collecting such data can be expensive, time-consuming, and error-prone. Often, the labels can be intrinsically difficult to ascertain precisely. Factors such as a lack of annotator expertise and privacy concerns can also negatively affect the quality and completeness of the annotations.

In an attempt to circumvent this limitation, several methods have been proposed to permit model learning from the data annotated with reduced labeling standards, which are generally easier to obtain. We will refer to such labels as *imprecise*. Figure 9.1 illustrates some typical mechanisms of label imprecision that are commonly addressed in the literature. Label imprecision requires a modification of the standard supervised training mechanism to build models for each specific case. For instance, *partial label learning* (PLL) [180, 181, 182, 183, 184, 185, 2] allows instances to have a set of candidate labels, instead of a single definitive one. *Semi-supervised Learning* (SSL) [186, 187, 188, 189, 3, 190, 191, 192, 193, 194] seeks to enhance the generalization ability when only a small set of labeled data is available, supplemented by a larger unlabeled set. *Noisy label learning* (NLL) [195, 196, 112, 197, 198, 199, 200, 201, 202, 203, 204, 205] deals with noisy scenarios where the labels are corrupted or incorrect. There is a greater variety of other forms of label imprecision, including crowd-sourcing [206, 207], programmable weak supervision [208, 209], and bag-level supervision [210, 211, 212, 213, 214, 215], among others.

While prior arts have demonstrated success in handling individual configurations of label imprecision, their approaches often differ substantially. They are tailored to a *specific* form of imprecision, as depicted in Figure 9.16. Such specificity not only imposes the necessity of devising a solution for emerging types of label imprecision scenarios, but also complicates the deployment in practical settings, where the annotations can be highly complex and may involve *multiple coexisting and interleaved* imprecision configurations. For instance, considering a scenario where both noisy labels and partial labels appear together, it might be challenging to adapt previous methods in NLL or PLL to this scenario since they either rely on the assumption of definite labels [207] or the existence of the correct label among label candidates [216], thus requiring additional algorithmic design. In fact, a few recent works have attempted to address the combinations of imprecise labels in this way, such as partial noisy label [217, 218] and semi-supervised partial label learning [219, 220]. However, simply utilizing a more



(a) Full Label (b) Partial Label (c) Semi-Supervised (d) Noisy Label

Figure 9.1: Illustration of the full label and imprecise label configurations. We use an example dataset of 4 training instances and 3 classes. (a) Full label, the annotation is a single true label; (b) Partial label, the annotation is a label candidate set containing true label; (c) Semi-supervised, only part of the dataset is labeled, and the others are unlabeled; (d) Noisy label, the annotation is mislabeled.

sophisticated or ad-hoc design can hardly scale to other settings. In addition, most of these approaches attempt to infer the correct labels given the imprecise information (*e.g.* through consistency with adjacent data [186, 221], iterative refinement [222, 223], average over given labels [224, 225], etc., to train the model, which inevitably accumulates error during training.

In this chapter, we formulate the problem from a different perspective: rather than taking the imprecise label information provided as a potentially noisy or incomplete attempt at assigning labels to instances, we treat it generically as the information that imposes a deterministic or statistical restriction of the actual applicable true labels. We then train the model over the distribution of all possible labeling entailed by the given imprecise information. More specifically, for a dataset with samples X and imprecise label information I, we treat the inaccessible full and precise labels Y as a latent variable. The model is then trained to maximize the likelihood of the provided information I. Since the likelihood computed over the joint probability $P(X, I; \theta) = \sum_Y P(X, I, Y; \theta)$ must marginalize out Y, the actual information I provided could permit a potentially exponential number of labeling. To deal with the resulting challenge of maximizing the logarithm of an expectation, we use the common approach of *expectation-maximization* (EM) [226], where the E-step computes the expectation of $P(X, I, Y; \theta)$ given the posterior of current belief $P(Y|X, I; \theta^t)$ at time step t and the M-step maximizes the tight variational lower bound over $P(X, I; \theta)$. The overall framework is thus largely agnostic to the various nature of label imprecision, with the imprecise label only affecting the manner in which the posterior $P(Y|X, I; \theta^t)$ is computed. In fact, current approaches designed for various imprecise label scenarios can be treated as specific instances of our framework. Our approach can serve as a solution towards a *unified and generalized* view for learning with *various* imprecise labels.

While there exist earlier attempts on generalized or EM solutions for different (other) imprecise supervisions or fuzzy observations [227, 228, 229, 230, 231, 213, 232, 233, 234], they usually require additional assumptions and approximations on the imprecise information for learnablility [216, 235], thus presenting limited scalability on practical settings [229]. On the contrary, the unified framework we propose subsumes all of these and naturally extends to the more practical "mixed" style of data, where different types of imprecise labels coexist. Moreover, for noisy labels, our framework inherently enables the learning of a *noise model*, as we will show in Section 9.7.2. Through comprehensive experiments, we demonstrate that the proposed imprecise label learning (ILL) framework not only outperforms previous methods for dealing with single imprecise labels of PLL, NLL, and SSL, but also presents robustness and effectiveness for mixed imprecise label learning (MILL) settings, leveraging the full potential to more challenging scenarios. Our contributions are summarized as follows:

- We propose an EM framework towards the unification of learning from *various* imprecise labels.
- We establish scalable and consistent state-of-the-art (SOTA) performance with the proposed method on partial label learning, semi-supervised learning, and noisy label learning, demonstrating our method's robustness in more diverse, complex label noise scenarios.
- To the best of our knowledge, our work is the first to show the robustness and effectiveness of a single unified method for handling the mixture of various imprecise labels.

9.2 Labeling scenarios and their distribution

In traditional supervised learning, each instance comes with a definitive label—e.g., positive or negative. However, under weak supervision or indirect forms of labeling, the provided information about the data is more constrained and does not directly specify which instances are positive or negative. Instead, we may only be informed about some property of the label configuration as a whole. This changes the problem from treating labels as known facts to treating them as *latent variables*, and the "label information" as *constraints* or *clues* that define a probability distribution over all valid labeling configurations.

9.2.1 Illustrative Example

9.2.1.1 Positive Bag (Weak Label)

Consider a bag containing three instances: X, Y, and Z. We are only given that the bag is "positive." as shown in figure 9.2 Under standard Multiple Instance Learning (MIL) assumptions, this means that at least one instance in the bag must be positive. However, we are not told which one is positive.

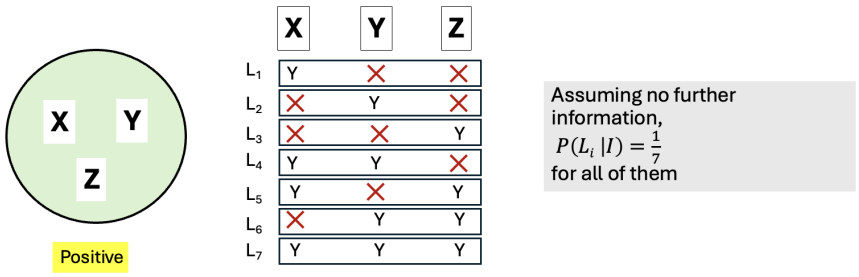


Figure 9.2: Example showing multiple labeling possibilities (L_1–L_7) when only the given information that bag is positive.

Without any additional information, there are multiple valid configurations of instance-level labels consistent with the bag-level "positive" label. Let us denote a positive instance by Y and a negative instance by X.

Since there are three instances, each can be either positive or negative, giving:

$$2^3 = 8 \tag{9.1}$$

possible ways to assign binary labels. Out of these 8 configurations, the only invalid labeling is the one where all three are negative (X, X, X). Thus, there are:

$$8 - 1 = 7 \tag{9.2}$$

valid configurations:

$$(Y, X, X), \quad (X, Y, X), \quad (X, X, Y), \quad (Y, Y, X), \quad (Y, X, Y), \quad (X, Y, Y), \quad (Y, Y, Y).$$

Since we have no reason to prefer one valid configuration over another, we assign equal probability to each of these 7 valid labelings:

$$P(L_i \mid I) = \frac{1}{7} \quad \forall i \in \{1, 2, \ldots, 7\}, \tag{9.3}$$

where I is the information: "This bag is positive." The result is a uniform distribution over all labelings that satisfy the bag-level constraint.

9.2.1.2 Bag with a Given Positive Instance Count

Another form of weak information could be: "This bag has exactly two positive instances." Consider again a bag of three instances (X, Y, Z). We ask: how many label configurations have exactly two positives?

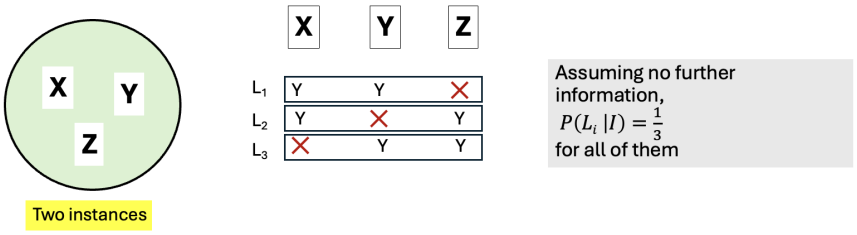


Figure 9.3: Example showing bag with given positive instance count

The total number of configurations with exactly two positives out of three instances is:

$$(Y, Y, X), \quad (Y, X, Y), \quad (X, Y, Y).$$

There are exactly 3 such configurations. Without any other information, we assign:

$$P(L_i \mid I) = \frac{1}{3} \quad \forall i \in \{1, 2, 3\}, \tag{9.4}$$

where I is: "This bag has two instances that are positive."

9.2.1.3 Key Insight

What we have effectively done is to translate weak or indirect label information into a probability distribution over possible instance-level label assignments. Instead of treating the weak information as a single "truth," we recognize that it only *narrows down* the space of possible labelings. Each labeling consistent with the provided information is one "hypothesis." In the absence of further evidence, we consider them all equally likely.

This is a Bayesian perspective: we incorporate the given label information as constraints that define a conditional distribution over latent labels. Thus, weak supervision guides a probability distribution over all valid configurations, rather than giving a single deterministic labeling.

9.2.2 Labels With Uncertainty (Probabilistic Confidence Scores)

9.2.2.1 Scenario

Consider a set of instances for which we have probabilistic confidences rather than hard labels. For example, let us have three instances X, Y, and Z. We assume each instance can be either "2" (positive) or "Not 2" (negative). Instead of a definitive label, we have probabilities indicating the likelihood of each instance being a "2."

Specifically:

$$P(X = 2) = 0.2, \quad P(Y = 2) = 0.7, \quad P(Z = 2) = 0.1.$$

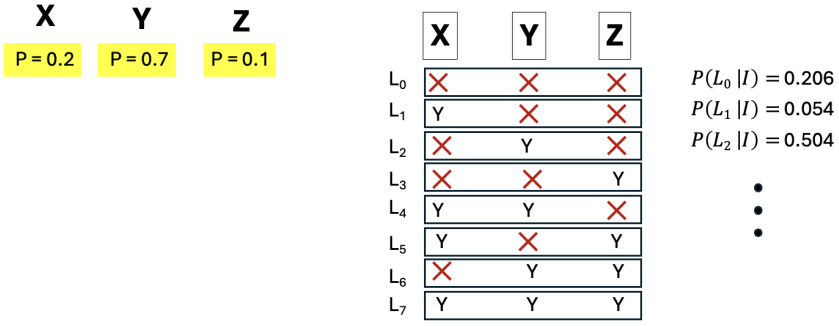


Figure 9.4: Example showing Uncertain label configuration

From these probabilities, it follows:
$P(X = \text{Not } 2) = 1-0.2 = 0.8, \quad P(Y = \text{Not } 2) = 1-0.7 = 0.3, \quad P(Z = \text{Not } 2) = 1-0.1 = 0.9.$

9.2.2.2 Distribution Over Label Configurations

Let each labeling configuration be denoted by L_i, where L_i specifies a particular assignment of each instance to "2" or "Not 2." Since we have 3 instances, each with 2 possible labels, there are:

$$2^3 = 8 \tag{9.5}$$

possible configurations.

For example, consider a configuration:

$$L_2 = (X = 2,\ Y = 2,\ Z = \text{Not } 2).$$

The probability of this configuration, given the individual instance probabilities, assuming independence, is:

$$P(L_2 \mid I) = P(X = 2) \cdot P(Y = 2) \cdot P(Z = \text{Not } 2) = (0.2)(0.7)(0.9). \tag{9.6}$$

We do this for every configuration L_i. The unnormalized probability of configuration L_i is:

$$P(L_i \mid I) = \prod_{j \in \{X,Y,Z\}} P(j = l_{i,j}), \tag{9.7}$$

where $l_{i,j}$ is the label assigned to instance j in configuration L_i.

To ensure these form a proper probability distribution, we normalize by the sum over all possible configurations:

$$P(L_i \mid I) = \frac{\prod_{j \in \{X,Y,Z\}} P(j = l_{i,j})}{\sum_k \prod_{j \in \{X,Y,Z\}} P(j = l_{k,j})}. \tag{9.8}$$

9.2.2.3 Benefits

Incorporating uncertain labels in this way lets the model reason directly about uncertainty. Rather than forcing a single "truth" for each instance, the model considers

multiple plausible configurations, each weighted by its probability. This is especially useful when obtaining exact labels is challenging or when labeling arises from weak signals or noisy sensors.

9.2.3 Labels With Noise

9.2.3.1 Scenario

In some cases, the provided labels may not only be uncertain, but explicitly noisy. That is, the label you observe might be incorrect due to flawed data collection, annotation errors, or sensor failures. Suppose we have three instances that *should* ideally be:

$$(X = 2,\ Y = \text{Not } 2,\ Z = \text{Not } 2),$$

but due to noise, the label for Z may sometimes be misrecorded as "2" when it should not be.

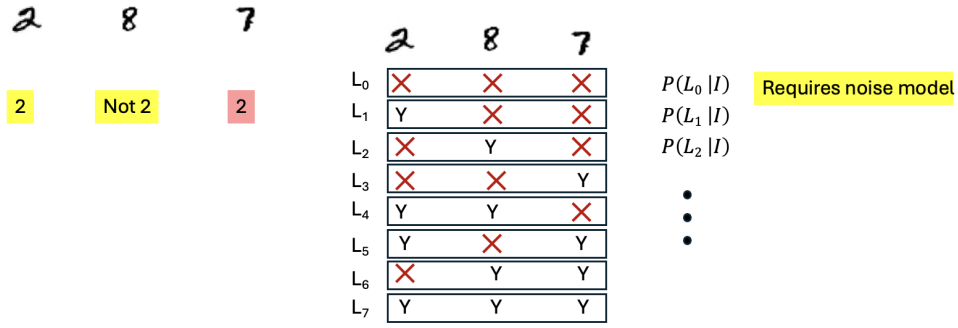


Figure 9.5: Noisy model required for labels with noise

9.2.3.2 Modeling Noise

To handle this, we introduce a *noise model*, which gives the probability of observing an incorrect label. For example, suppose:

$$P(\text{label flip}) = \epsilon = 0.05.$$

If the original intended label for Z is "Not 2," the probability of observing "Z = 2" due to noise is $\epsilon = 0.05$. Conversely, the probability of a correct label is $1 - \epsilon = 0.95$.

Thus, if the observed label is "Z = 2," the probability that this observation is correct is low in this scenario. We can incorporate such noise models by adjusting the probabilities:

$$P(Z = \text{Observed Label} \mid Z = \text{True Label}) = \begin{cases} 1 - \epsilon & \text{if True Label} = \text{Observed Label}, \\ \epsilon & \text{otherwise.} \end{cases} \tag{9.9}$$

9.2.3.3 Inference With a Noise Model

Given a set of noisy observed labels, we combine the noise probabilities with prior beliefs about the true labels. We then compute a posterior distribution over all possible "true" label configurations:

$$P(L_i \mid I, \text{Noise Model}) \propto P(I \mid L_i, \text{Noise Model}) P(L_i), \tag{9.10}$$

where I is the information from observed labels, and L_i represents a hypothetical true labeling. Here, $P(I \mid L_i, \text{Noise Model})$ accounts for how likely we are to observe the given noisy labels if L_i were the true configuration.

Benefits of the approach : By modeling noise explicitly, we avoid treating observed but incorrect labels as ground truth. This reduces the risk of overfitting to faulty annotations and leads to more robust, trustworthy models that better reflect the actual underlying relationships.

9.2.4 Strong Labels (Perfect Information)

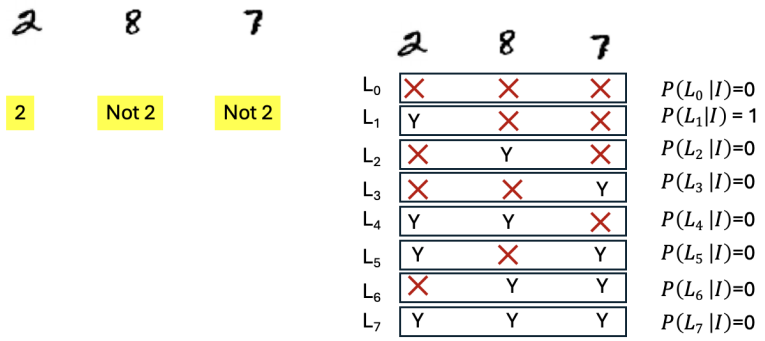


Figure 9.6: Strong labels

9.2.4.1 Scenario

At the other extreme, we may have perfect, noise-free labels. Suppose we are certain that:

$$X = 2, \quad Y = \text{Not } 2, \quad Z = \text{Not } 2.$$

In this ideal scenario, the label configuration is known with absolute certainty:

$$P(L_{\text{true}} \mid I) = 1, \tag{9.11}$$

for the correct configuration L_{true}, and:

$$P(L_i \mid I) = 0 \quad \text{for all } i \neq \text{true}. \tag{9.12}$$

No uncertainty or noise modeling is needed since the data are completely reliable. Models trained under these conditions may achieve excellent predictive performance, but this is often challenging to achieve in practice.

9.3 Basic EM formulation

Let $X = \{x_1, x_2, \cdots\}$ represent the features and $Y = \{y_1, y_2, \cdots\}$ represent their labels. We are separating the labels from the features for this explanation, however the data, in fact, comprise the tuples (x, y).

Let I be *information* given to us about the data. The information may be an explicit labelling, e.g. $I \implies Y = \{0, 1, 0, \cdots\}$ for a binary classification problem, or $I \implies Y = \{class(x_1), class(x_2), \cdots\}$ more generally. Such a condition would represent a *strong* labelling of the data. Alternately, for a binary classification problem, $I \implies \sum_{i=1}^{N} y_i \geq 1$ (indicating that among the set (or *bag*) of instances $\{x_1, \cdots, x_N\}$, at least one instance is positive. I might also take other forms, e.g. it might comprise bounds on the number of positive instance, noisy labels, partial labels (where only some instances or labelled), or any other form of information.

From a data perspective, we are only given X and I. We must learn the distribution of $P(x, y)$ from these givens.

Let $P(X, I; \theta)$ represent a parametric form for the joint distribution of X and I. We have:

$$P(X, I; \theta) = \sum_{Y} P(X, Y, I; \theta)$$

The maximum likelihood principle requires us to find:

$$\begin{aligned} \hat{\theta} &= \arg\max_{\theta} \log P(X, I; \theta) \\ &= \arg\max_{\theta} \log \sum_{Y} P(X, Y, I; \theta) \end{aligned} \tag{9.13}$$

Here Y represents the *true* label set for X. If I represents a full (strong) labelling of X, then $I \implies Y$ and the above simply collapses to

$$\hat{\theta} = \arg\max_{\theta} \log P(X, Y; \theta) \tag{9.14}$$

which, for generative models, is decomposed into

$$\hat{\theta} = \arg\max_{\theta} \log P(Y; \theta) + \log P(X|Y; \theta)$$

and, for discriminative models, is decomposed into

$$\hat{\theta} = \arg\max_{\theta} \log P(X) + \log P(Y|X; \theta)$$

In the most general case, however, we cannot assume strong labels, i.e $I \not\implies Y$, and

may only represent partial, incomplete, or noisy information about Y. In that case, Equation 9.26 must be explicitly optimized.

We know from long years of painful experience that Equation 9.26 cannot generally be solved in closed form and requires iterative hill-climbing solutions. Of these, arguably the most popular is the Expectation Maximzation algorithm, which iteratively maximizes a tight variational lower bound on the log likelihood, which in our case becomes

$$\begin{aligned} \theta^{k+1} &= \arg\max_{\theta} E_{Y|X,I;\theta^k} \log P(X, Y, I; \theta) \\ &= \arg\max_{\theta} E_{Y|X,I;\theta^k} \log P(X, Y; \theta) + \log P(I|X, Y; \theta) \end{aligned} \tag{9.15}$$

where θ^k is the k^{th} estimate of the optimal θ.

The precise nature of the solution to the Equation 9.15 will depend on the nature the data and the information provided I. If I repersents noisy labels, then $P(I|X, Y; \theta)$ represents a potentially learnable noise model.

In problems where I represents some information about the *true* labels (i.e. not noisy labels), e.g. the actual labels, bounds on the labels, weak labels etc, $P(I|X, Y; \theta) = P(I|Y)$, i.e. once the actual labels are given, the probability of I no longer depends on X or parameters θ. In this case Equation 9.15 reduces to:

$$\theta^{k+1} = \arg\max_{\theta} E_{Y|X,I;\theta^k} \log P(X, Y; \theta) \tag{9.16}$$

In particular, for discriminative models, such as neural networks, it becomes

$$\theta^{k+1} = \arg\max_{\theta} E_{Y|X,I;\theta^k} \log P(Y|X; \theta) \tag{9.17}$$

Note that this solution depends on our ability to compute $P(Y|X, I; \theta)$. In some cases we can compute this in closed form; in others we may require approximations (converting the EM solution to the more generic approximate solution of maximizing variational lower bounds).

9.4 IID observations

Let us now consider the case where the data are IID, i.e. each instance of (x_i, y_i) is statistically independent of every other instance. Assume without loss of generality that the set of observations occurs as a sequence $\{(x_1, y_1), (x_2, y_2), \cdots\}$. The specific order of the sequence does not matter for the IID case (although it might in other cases).

First, we will assume conditional independence of the y_is, given X (recalling that $X = \{x_1, x_2, \cdots\}$), i.e.

$$P(Y|X;\theta) = \prod_i P(y_i|X;\theta) \tag{9.18}$$

$$\log P(Y|X;\theta) = \sum_i \log P(y_i|X;\theta) \tag{9.19}$$

This assumption is accurate if computed by any model where the labels predicted for any instance are not fed back to the net as input when computing the labels for other inputs.

Using the linearity of the expectation operator, we can write the following. Here, in Equation 9.21 we have adopted the $P_{y_i}(y)$ subscripted notation to distinguish between the random variable y_i and the value y that it takes.

$$E_{Y|X,I;\theta^k} \log P(Y|X;\theta) = \sum_i E_{Y|X,I;\theta^k} \log P(y_i|X;\theta) \tag{9.20}$$

$$= \sum_i \sum_y P_{y_i}(y|X,I;\theta^k) \log P_{y_i}(y|X;\theta) \tag{9.21}$$

The problem thus reduces to that of computing $P_{y_i}(y|X,I;\theta^k)$. We can write

$$P_{y_i}(y|X,I;\theta^k) = \frac{1}{P(I|X;\theta^k)} P_{y_i}(y,I|X;\theta^k) \tag{9.22}$$

We recognize $P(I|X;\theta^k)$ as the probability of I given the input data, as computed using the current model with parameter θ^k, and $P_{y_i}(y,I|X;\theta^k)$ as the joint probability of I and the label of the i^{th} instance taking the value y, given X, also as computed using the current model with parameter θ^k.

In order to compute $P_{y_i}(y,I|X;\theta^k)$, we treat the problem of assigning labels $y_1, y_2, \cdots$ to the inputs $x_1, x_2, \cdots$ as one of generating a sequence of symbols $y_1, y_2, \cdots$, $y_i \in \{0,1\}$,such that at least one of the symbols y_i is 1.

Figure 9.7 shows a finite-state automaton that generates symbol sequences with this feature. In this figure, the green dot repersents a state that outputs a symbol 1, while the red dot is a state that outputs 0. A sequence of symbols that mandatorily includes a 1 can be one of four kinds: one that begins with a 1 and ends with a 0, one that begins with a 0 and ends with a 1, one that begins and ends with 0s, but has one or more 1s between them, and one that begins and ends with a 1, and optionally has one or more intervening 0s. The four branches of the automaton show these four possibilities. Note that if one were to additionally assume that the first and last symbol in the sequence are always a 0, then the automaton becomes much simpler, including only the second branch (from the top) in the figure.

Figure 9.8 represents the set of all possible label sequences of length 6, that can be generated by the automaton of Figure 9.7 in the form of a graph, or *trellis*. This trellis, which is just a product of the automaton of Figure 9.7 and a linear graph (constructed

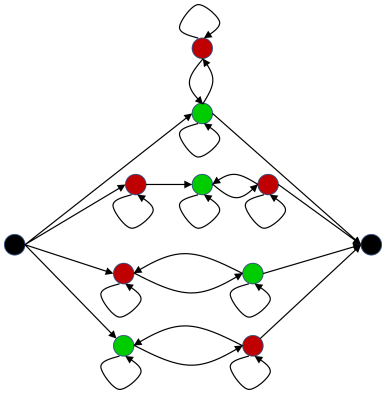


Figure 9.7: Finite state graph representing all ways of aligning a sequences of instances such that at least one instance is from the positive class. Green dots represent the "positive" state (positive instances) and red dots represent the "negative" state. The four branches from the top to the bottom represent the cases where (a) the sequence both begins and ends with a positive label, (b) the sequence begins and ends with a negative label, buth as at least one positive instance, (c) the sequence begins with a negative instance and ends with a positive instance, and (d) the sequence begins with a positive instance and ends with a negative instance. All edges have weight 1.

simply by "unrolling" the automaton so that every transition actually transitions to the next symbol), can be extended to accommodate an input of any length.

If we were to restrict our original automaton such that the first and last symbols in the sequence were both 0, the trellis would only consist of the second subtrellis (comprising 3 states) in the figure.

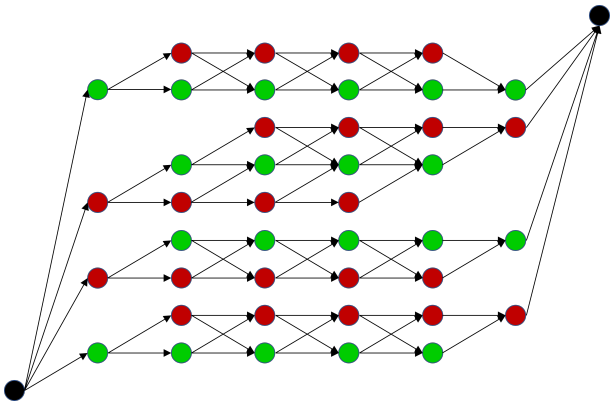


Figure 9.8: "Trellis" representing all possible ways of labelling a sequence of six instances such that at least one instance is positive. This is composed from the graph of Figure 9.7.

We are now set to compute $P_{y_i}(y, I|X; \theta^k)$.

Using our current model for $P(y|x; \theta^k)$ we compute $P_{y_i}(y|X; \theta^k), \quad y \in \{0, 1\}, \quad \forall i$. We then associate $P_{y_i}(0|X; \theta^k)$ and $P_{y_i}(1|X; \theta^k)$ with the red and green states of the i^{th}

coloured column of the trellis, as illustrated by Figure 9.9. The black nodes, representing the source and destination states get a value 1, as do all edges.

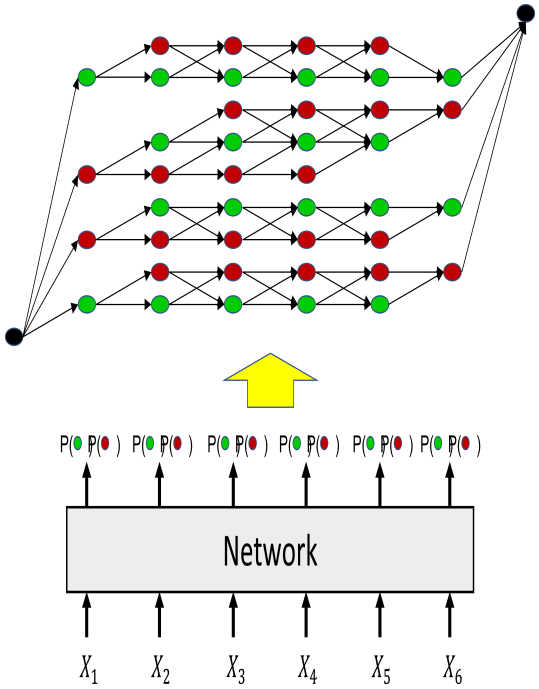


Figure 9.9: The entire CTC model for the trellis of Figure 9.8. At each instance, the probabilities assigned to the "positive" (green) nodes and "negative" (red) nodes is computed by the neural network.

$P_{y_i}(y, I|X; \theta^k)$ can subsequently be computed using the forward-backward algorithm. $P(I|X; \theta^k)$ is merely the forward probability in the final absorbing node of the trellis. These can be used in Equation 9.22 to compute $P_{y_i}(y|X, I; \theta^k)$, which can then be used to compute the objective of Equation 9.21, which must be optimized via gradient ascent.

9.5 Generalizing

The example above is only an illustration for a simple binary classification problem with the trivial case of a weak label. The key takeaway lesson is that any kind of labelling information I can be used within an EM objective to obtain linear time (in the number of inputs) updates to the model *provided I can be stated as a (language accepted by a) finite-state automaton.* This, in fact, subsumes many cases, leading to very simple solutions.

Figure 9.15 shows several examples of FSAs for various types of label information for a binary classification tasks. Here we have used non-deterministic finite automata (NFA), and shown them using the more conventional notation, which shows symbols

on the edges, rather than the nodes of the automaton. The symbol "1" represents the positive class and "0" represents the negative class. The examples include FSAs for strong labelling (Figure 9.10), for the case where no labels are provided and any labelling is potentially valid (Figure 9.14), a simpler version of the weak-label solution of Figure 9.7 (Figure 9.12), label conditions that give us an exact count of positives – two, in the shown example (Figure 9.13), and a label condition that gives us a minimum count on positives – two in our example (Figure 9.14) .

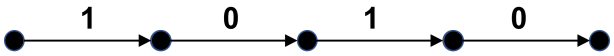


Figure 9.10: An FSA for the strong label sequence "1, 0, 1, 0".

Figure 9.11: FSA for unlabelled data.

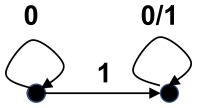


Figure 9.12: FSA for weak labelling: "at least one positive instance".

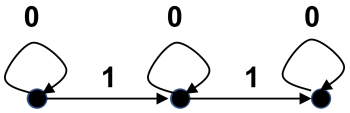


Figure 9.13: FSA representing the event that the collection has exactly two positive instances.

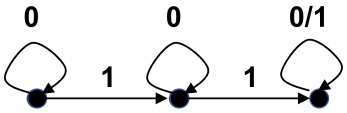


Figure 9.14: FSA for labellings with *at least* two positive instances.

Figure 9.15: Finite-state automata (nondeterministic) representing different kinds of label information I for a binary classification task.

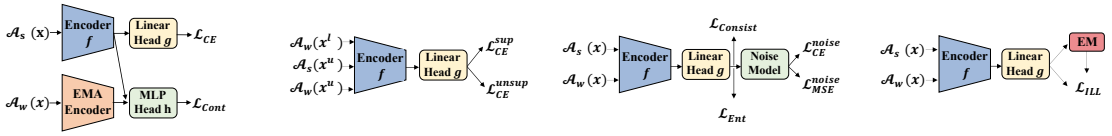
Although the examples of Figure 9.15 are specific to binary classification, they are easily extended to multi-class classification, simply by extending the symbol set for the label "language" accepted by the automatons.

In all cases, the complexity of computing the loss remains linear in the number of instances in the collection. Since the branching factor of the FSAs is typically not greater than $O(C)$, where C is the number of classes, loss computation is also upper bounded quadratically by the number of classes, in most cases.

9.6 Baselines solutions from different imprecise label settings

In this section, we illustrate the notations and baselines from different imprecise label settings that adopt various solutions. We will show later how our proposed method generalize and subsumes these prior arts. Let $\mathcal{X}$ denote the input space,

and $\mathcal{Y} = [C] := \{1, \ldots, C\}$ represent the label space with C distinct labels. A fully annotated training dataset of size N is represented as $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i\in[N]}$. Learning with imprecise labels involves approximating the mapping function $f \circ g : \mathcal{X} \to \mathcal{Y}$ from a training dataset where the true label y is not fully revealed from the annotation process. Here f is the backbone for feature extraction, g refers to the classifier built on top of the features, and the output from $f \circ g$ is the predicted probability $\mathbf{p}(y|\mathbf{x};\theta)$, where θ is the learnable parameter for $f \circ g$. In this study, we primarily consider three imprecise label configurations (as illustrated in Figure 9.1) and their corresponding representative learning paradigms (as shown in Figure 9.16), namely partial label learning, semi-supervised learning, and noisy label learning.



(a) Partial Label (b) Semi-Supervised (c) Noisy Label (d) Imprecise Label

Figure 9.16: Baseline model pipelines for various imprecise label configurations. (a) PiCO [2] for partial label learning. (b) FixMatch [3] for semi-supervised learning. (c) SOP [4] for noisy label learning. (d) The proposed unified framework. It accommodates *any* imprecise label configurations and also mixed imprecise labels with an EM formulation.

Partial label learning (PLL). PLL aims to learn with a candidate label set $\mathbf{s} \subset \mathcal{Y}$, where the ground truth label $y \in \mathcal{Y}$ is concealed in $\mathbf{s}$. The training data for partial labels thus becomes $\mathcal{D}_{\text{PLL}} = \{(\mathbf{x}_i, \mathbf{s}_i)\}_{i\in[N]}$. PiCO [2] is a recent contrastive method that employs class prototypes to enhance label disambiguation (as shown in Figure 9.16(a)). It optimizes the cross-entropy (CE)[1] loss between the prediction of the augmented training sample $\mathcal{A}_{\text{w}}(\mathbf{x})$ and the disambiguated labels $\hat{\mathbf{s}}$. PiCO learns a set of class prototypes from the features associated with the same pseudo-targets. A contrastive loss, based on MOCO [236], is employed to better learn the feature space, drawing the projected and normalized features $\mathbf{z}_{\text{w}}$ and $\mathbf{z}_{\text{s}}$ of the two augmented versions of data $\mathcal{A}_{\text{w}}(\mathbf{x})$ and $\mathcal{A}_{\text{s}}(\mathbf{x})$ [2] closer. The objective of PiCO is formulated as:

$$\mathcal{L}_{\text{PiCO}} = \mathcal{L}_{\text{CE}}\left(\mathbf{p}(y|\mathcal{A}_{\text{w}}(\mathbf{x});\theta), \hat{\mathbf{s}}\right) + \mathcal{L}_{\text{Cont}}\left(\mathbf{z}_{\text{w}}, \mathbf{z}_{\text{s}}, \mathcal{M}\right). \tag{9.23}$$

Semi-supervised learning (SSL). For SSL, we can define the labeled dataset as $\mathcal{D}^{\text{L}}_{\text{SSL}} = \{(\mathbf{x}^{\text{l}}_i, y^{\text{l}}_i)\}_{i\in[N^{\text{L}}]}$, and the unlabeled dataset as $\mathcal{D}^{\text{U}} = \{\mathbf{x}^{\text{u}}_j\}_{j\in[N^{\text{L}}+1, N^{\text{L}}+N^{\text{U}}]}$, with $N^{\text{L}} \ll N^{\text{U}}$. A general confidence-thresholding based self-training [221, 3] pipeline for

[1]For simplicity, we use $\mathcal{L}_{\mathbf{CE}}$ for labels of the formats of class indices, one-hot vectors, and class probabilities.

[2]We use $\mathcal{A}_{\text{w}}$ to indicate the weaker data augmentation and $\mathcal{A}_{\text{s}}$ to indicate the stronger data augmentation.

SSL is shown in Figure 9.16(b). Consider FixMatch [3] as an example; there are usually two loss components: the supervised CE loss on labeled data and the unsupervised CE loss on unlabeled data. For the unsupervised objective, the pseudo-labels $\hat{y}^{\mathrm{u}}$ from the network itself are used to train on the unlabeled data. A "strong-weak" augmentation [221] is commonly adopted. To ensure the quality of the pseudo-labels, only the pseudo-labels whose confidence scores $\hat{p}^{\mathrm{u}}$ are greater than a threshold τ are selected to participate in training:

$$\mathcal{L}_{\mathrm{Fix}} = \mathcal{L}_{\mathrm{CE}}\left(\mathbf{p}(y|\mathcal{A}_{\mathrm{w}}(\mathbf{x}^{\mathrm{l}});\theta), y^{\mathrm{l}}\right) + \mathbb{1}\left(\hat{p}^{\mathrm{u}} \geq \tau\right)\mathcal{L}_{\mathrm{CE}}\left(\mathbf{p}(y|\mathcal{A}_{\mathrm{s}}(\mathbf{x}^{\mathrm{u}});\theta), \hat{y}^{\mathrm{u}}\right). \tag{9.24}$$

Noisy label learning (NLL). NLL aims at learning with a dataset of corrupted labels, $\mathcal{D}_{\mathrm{NLL}} = \{(\mathbf{x}_i, \hat{y}_i)\}_{i\in[N]}$. We illustrate the NLL pipeline (in Figure 9.16(c)) with the recent sparse over-parameterization (SOP) model [4], where a sparse *noise model* consisting of parameters $\mathbf{u}_i, \mathbf{v}_i \in [-1, 1]^C$ for each sample is adopted. The noise model transforms the network prediction from the true label distribution into the noisy label distribution. A CE loss and a mean-squared-error (MSE) loss optimize parameter $\{\mathbf{u}_i\}$ and $\{\mathbf{v}_i\}$ respectively:

$$\mathcal{L}_{\mathrm{SOP}} = \mathcal{L}_{\mathrm{CE}}\left(\phi\left(\mathbf{p}(y|\mathcal{A}_{\mathrm{w}}(\mathbf{x});\theta) + \mathbf{m}\right), \hat{y}\right) + \mathcal{L}_{\mathrm{MSE}}\left(\mathbf{p}(y|\mathcal{A}_{\mathrm{w}}(\mathbf{x});\theta) + \mathbf{m}, \hat{y}\right), \tag{9.25}$$

where ϕ denotes the L_∞ normalization and $\mathbf{m}_i = \mathbf{u}_i \odot \mathbf{u}_i \odot \hat{\mathbf{y}}_i^{\mathrm{oh}} - \mathbf{v}_i \odot \mathbf{v}_i \odot \left(1 - \hat{\mathbf{y}}_i^{\mathrm{oh}}\right)$, with $\hat{\mathbf{y}}_i^{\mathrm{oh}}$ referring to the one-hot version of y_i. Consistency regularization with strong-weak augmentation and entropy class-balance regularization are additionally utilized for better performance in SOP [4].

9.7 Imprecise Label Learning

Although current techniques demonstrate potential in addressing particular forms of imprecise labels, they frequently fall short in adaptability and transferability to more complicated and more realistic scenarios where multiple imprecise label types coexist and interleave. This section first defines the proposed expectation-maximization (EM) formulation for learning with various imprecise labels. Then, we demonstrate that our unified framework seamlessly extends to partial label learning, semi-supervised label learning, noisy label learning, and the more challenging setting of mixed imprecise label learning. Connections and generalization to previous pipelines can also be drawn clearly under the proposed EM framework.

9.7.1 A Unified Framework for Learning with Imprecise Labels

Exploiting information from imprecise labels. The challenge of learning with imprecise labels lies in learning effectively with inaccurate or incomplete annotation information. Per the analysis above, prior works catering to specific individual imprecise

labels either explicitly or implicitly attempt to infer the precise labels from the imprecise label information. For example, partial label learning concentrates on the disambiguation of the ground truth label from the label candidates [2, 237, 218] or averaging equally over the label candidates [238]. In semi-supervised learning, after the model initially learns from the labeled data, the pseudo-labels are treated as correct labels and utilized to conduct self-training on the unlabeled data [239, 3]. Similarly, for noisy label learning, an integral part that helps mitigate overfitting to random noise is the implementation of an accurate noise model capable of identifying and rectifying the incorrect labels [203, 4], thereby ensuring the reliability of the learning process. However, inferring the correct labels from the imprecise labels or utilizing the imprecise labels directly can be very challenging and usually leads to errors accumulated during training [239, 240], which is also known as the confirmation bias. In this work, we take a different approach: we consider all possible labeling along with their likelihood that the imprecise labels fulfill to train the model, rather than using a single rectified label from the imprecise information. Such an approach also eliminates the requirements for designing different methods for various imprecise labels and provides a unified formulation instead, where closed-form solutions can be derived.

A unified framework for learning with imprecise labels (ILL). Let $\{\mathbf{x}_i\}_{i\in[N]}$ represent the features as realizations from X and $\{y_i\}_{i\in[N]}$ represent their precise labels as realizations from Y for the training data. Ideally, Y would be fully specified for X. In the imprecise label scenario, however, Y is not provided; instead we obtain imprecise label information I. We view I not as *labels*, but more abstractly as a variable representing the *information* about the labels. From this perspective, the actual labels Y would have a distribution $P(Y|I)$, and I can present in various forms. When the information I provided is the precise true label of the data, $P(Y|I)$ would be a delta distribution, taking a value 1 at the true label, and 0 elsewhere. If I represents partial labels, then $P(Y|I)$ would have non-zero value over the candidate labels, and be 0 elsewhere. When I represents a set of noisy labels, $P(Y|I)$ would represent the distribution of the true labels, given the noisy labels. When I does not contain any information, i.e., unlabeled data, Y can take any value.

By the maximum likelihood estimation (MLE) principle, we must estimate the model to maximize the likelihood of the data/information we have been provided, namely X and I. Let $P(X, I; \theta)$ represent a parametric form for the joint distribution of X and I[3] Explicitly considering the labels Y, we have $P(X, I; \theta) = \sum_Y P(X, Y, I; \theta)$. The

[3]The actual parameters θ may apply only to some component such as $P(Y|X; \theta)$ of the overall distribution; we will nonetheless tag the entire distribution $P(X, I; \theta)$ with θ to indicate that it is dependent on θ overall.

maximum likelihood principle requires us to find:

$$\theta^* = \arg\max_{\theta} \ \log P(X, I; \theta) = \arg\max_{\theta} \ \log \sum_{Y} P(X, Y, I; \theta), \tag{9.26}$$

with θ^* denotes the optimal value of θ. Equation (9.26) features the log of an expectation and cannot generally be solved in closed-form, and requires iterative hill-climbing solutions. Of these, arguably the most popular is the expectation-maximization (EM) algorithm [226], which iteratively maximizes a tight variational lower bound on the log-likelihood. In our case, applying it becomes:

$$\begin{aligned} \theta^{t+1} &= \arg\max_{\theta} \mathbb{E}_{Y|X,I;\theta^t} \left[\log P(X, Y, I; \theta)\right] \\ &= \arg\max_{\theta} \mathbb{E}_{Y|X,I;\theta^t} \left[\log P(Y|X; \theta) + \log P(I|X, Y; \theta)\right], \end{aligned} \tag{9.27}$$

where θ^t is the t^{th} estimate of the optimal θ. Note that $P(X; \theta)$ is omitted from Equation (9.27) as $P(X)$ is not based on θ. The detailed derivation of the variational lower bound is shown in Appendix D.3.1. There are several implications from Equation (9.27). (i) The expectation over the posterior $P(Y|X, I; \theta^t)$ equates to considering *all* labeling entailed by the imprecise label information I, rather than any single (possibly corrected) choice of label. For independent instances setting studied in this paper, we can derive closed-form training objectives from this formulation as shown in Section 9.7.2. (ii) The property of the second term $\log P(I|X, Y; \theta)$ is dependent on the nature of imprecise label I. If I contains information about the true labels Y, such as the actual labels or the label candidates, it can be reduced to $P(I|Y)$, *i.e.*, the probability of I is no longer dependent on X or θ and thus can be ignored from Equation (9.27). If I represents the noisy labels, $P(I|X, Y; \theta)$ instead includes a potentially learnable noise model. (iii) It is a general framework towards the unification of any label configuration, including full labels, partial labels, low-resource labels, noisy labels, etc. In this work, we specialize the proposed EM framework to PLL, SSL, NLL, and the mixture of them in the following.

9.7.2 Instantiating the Unified EM Formulation

We illustrate how to seamlessly expand the formulation from Equation (9.27) to partial label learning, semi-supervised learning, noisy label learning, and mixture settings, with derived closed-form loss function[4] for each setting here. The actual imprecise labels only affect the manner in which the posterior $P(Y|X, I; \theta^t)$ is computed for each setting. We show that all learning objectives derived from Equation (9.27) naturally include a consistency term with the posterior as the soft target. We also demonstrate that the proposed unified EM framework closely connects with the prior arts, which reveals the potential reason behind the success of these techniques. Note that while we only demonstrate the application of the proposed framework to four settings here, it can

[4]To formulate the loss function, we convert the problem to minimization of the negative log-likelihood.

also be flexibly extended to other settings. More details of derivation below are shown in Appendix D.3.

Partial label learning (PLL). The imprecise label I for partial labels is defined as the label candidate sets S containing the true labels. These partial labels indicate that the posterior $P(Y|X,S;\theta^t)$ can only assign its masses on the candidate labels. Since S contains the information about the true labels Y, $P(S|X,Y;\theta)$ reduces to $P(S|Y)$, and thus can be ignored. We also demonstrate with instance dependent partial labels that maintains $P(S|X,Y;\theta)$ in Appendix D.4.2.2. Defining the label candidates as $\{\mathbf{s}_i\}_{i\in[N]}$ and substituting it in Equation (9.27), we have the loss function of PLL derived using ILL framework:

$$\mathcal{L}_{\text{ILL}}^{\text{PLL}} = -\sum_{Y\in[C]} P(Y|X,S;\theta^t)\log P(Y|X;\theta) \equiv \mathcal{L}_{\text{CE}}\left(\mathbf{p}(y|\mathcal{A}_{\text{s}}(\mathbf{x});\theta), \mathbf{p}(y|\mathcal{A}_{\text{w}}(\mathbf{x}),\mathbf{s};\theta^t)\right), \tag{9.28}$$

where $\mathbf{p}(y|\mathcal{A}_{\text{w}}(\mathbf{x}),\mathbf{s};\theta^t)$ is the normalized probability that $\sum_{k\in C} p_k = 1$, and $p_k = 0, \forall k \in \mathbf{s}$. Equation (9.28) corresponds exactly to consistency regularization [221], with the normalized predicted probability as the soft pseudo-targets. This realization on PLL shares similar insights as [185] which exploits a gradually induced loss weight for PLL on multiple augmentations of the data. However, our framework is much simpler and more concise as shown in Appendix D.4.2.2, which does not require additional techniques.

Semi-supervised learning (SSL) In SSL, the input X consists of the labeled data X^{L} and the unlabeled data X^{U}. The imprecise label for SSL is realized as the limited number of full labels Y^{L} for X^{L}. The labels Y^{U} for unlabeled X^{U} are unknown and become the latent variable. Interestingly, for the unlabeled data, there is no constraint on possible labels it can take. The posterior $P(Y^{\text{U}}|X^{\text{L}},X^{\text{U}},Y^{\text{L}};\theta)$, which is the actual prediction from the network, can be directly utilized as soft targets for self-training. Since Y^{L} is conditionally independent with Y^{U} given X, the second term of Equation (9.27): $P(Y^{\text{L}}|X^{\text{L}},X^{\text{U}},Y^{\text{U}};\theta)$, is reduced to $P(Y^{\text{L}}|X^{\text{L}};\theta)$, which corresponds to the supervised objective on labeled data. The loss function for SSL thus becomes:

$$\begin{aligned}\mathcal{L}_{\text{ILL}}^{\text{SSL}} &= -\sum_{Y\in[C]} P(Y^{\text{U}}|X^{\text{U}},X^{\text{L}},Y^{\text{L}};\theta^t)\log P(Y^{\text{U}}|X^{\text{U}},X^{\text{L}};\theta) - \log P(Y^{\text{L}}|X^{\text{L}};\theta) \\ &\equiv \mathcal{L}_{\text{CE}}\left(\mathbf{p}(y|\mathcal{A}_{\text{s}}(\mathbf{x}^{\text{u}});\theta), \mathbf{p}(y|\mathcal{A}_{\text{w}}(\mathbf{x}^{\text{u}});\theta^t)\right) + \mathcal{L}_{\text{CE}}\left(\mathbf{p}(y|\mathcal{A}_{\text{w}}(\mathbf{x}^{\text{l}});\theta), y^{\text{l}}\right)\end{aligned} \tag{9.29}$$

The first term corresponds to the unsupervised consistency regularization usually employed in SSL, and the second term refers to the supervised CE loss only on labeled data. Equation (9.29) has several advantages over the previous methods. It adopts the prediction as soft-targets of all possible labeling on unlabeled data, potentially circumventing the confirmation bias caused by pseudo-labeling and naturally utilizing all unlabeled data which resolves the quantity-quality trade-off commonly existing in

SSL [3, 194]. It also indicates that previous pseudo-labeling with confidence threshold implicitly conducts the EM optimization, where the maximal probable prediction approximates the expectation, and the degree of the approximation is determined by the threshold τ, rationalizing the effectiveness of dynamic thresholding.

Noisy label learning (NLL). Things become more complicated here since the noisy labels $\hat{Y}$ do not directly reveal the true information about Y, thus $P(\hat{Y}|Y, X; \theta)$ inherently involves a noise model that needs to be learned. We define a simplified instance-independent[5] noise transition model $\mathcal{T}(\hat{Y}|Y; \omega)$ with parameters ω, and take a slightly different way to formulate the loss function for NLL from the ILL framework:

$$\begin{aligned}\mathcal{L}_{\text{ILL}}^{\text{NLL}} &= -\sum_{Y\in[C]} P(Y|X, \hat{Y}; \theta^t, \omega^t) \log P(Y|X, \hat{Y}; \theta, \omega^t) - \log P(\hat{Y}|X; \theta, \omega) \\ &\equiv \mathcal{L}_{\text{CE}}\left(\mathbf{p}(y|\mathcal{A}_{\text{s}}(\mathbf{x}), \hat{y}; \theta, \omega^t), \mathbf{p}(y|\mathcal{A}_{\text{w}}(\mathbf{x}), \hat{y}; \theta^t, \omega^t)\right) + \mathcal{L}_{\text{CE}}\left(\mathbf{p}(\hat{y}|\mathcal{A}_{\text{w}}(\mathbf{x}); \theta, \omega), \hat{y}\right),\end{aligned} \tag{9.30}$$

where the parameters ω and θ are learned end-to-end. The first term corresponds to the consistency regularization of prediction conditioned on noisy labels and the second term corresponds to the supervised loss on noisy predictions that are converted from the ground truth predictions. Both quantities are computed using the noise transition model given the noisy label $\hat{y}$:

$$\mathbf{p}(y|\mathbf{x}, \hat{y}; \theta, \omega^t) \propto \mathbf{p}(y|\mathbf{x}; \theta)\mathcal{T}(\hat{y}|y; \omega^t), \text{and } \mathbf{p}(\hat{y}|\mathbf{x}; \theta, \omega) = \sum_{y\in[C]} \mathbf{p}(y|\mathbf{x}; \theta)\mathcal{T}(\hat{y}|y; \omega). \tag{9.31}$$

Mixture imprecise label learning (MILL). We additionally consider a more practical setting, mixture of imprecise label learning, with partial labels, noisy labels, and unlabeled data interleaved together. On the unlabeled data, the unsupervised objective is the same as the unsupervised consistency regularization of SSL as shown in Equation (9.29). The labeled data here present partial and noisy labels $\hat{\mathbf{s}}$. Thus the noisy supervised objective in Equation (9.31) becomes the supervised consistency regularization as in Equation (9.28) of partial label setting to train the noise transition model, and the noisy unsupervised objective becomes the consistency regularization of the prediction conditioned on noisy partial labels. Thus we have the loss function for MILL derived as:

$$\begin{aligned}\mathcal{L}_{\text{ILL}}^{\text{MILL}} &= \mathcal{L}_{\text{CE}}\left(\mathbf{p}\left(y \mid \mathcal{A}_{\text{s}}(\mathbf{x}^l), \hat{\mathbf{s}}^l; \theta, \omega^t\right), \mathbf{p}\left(y \mid \mathcal{A}_{\text{w}}(\mathbf{x}^l), \hat{\mathbf{s}}^l; \theta^t, \omega^t\right)\right) \\ &+ \mathcal{L}_{\text{CE}}\left(\mathbf{p}\left(\hat{y} \mid \mathcal{A}_{\text{w}}(\mathbf{x}^l); \theta, \omega\right), \hat{\mathbf{s}}^l\right) \\ &+ \mathcal{L}_{\text{CE}}\left(\mathbf{p}(y|\mathcal{A}_{\text{s}}(\mathbf{x}^{\text{u}}); \theta), \mathbf{p}(y|\mathcal{A}_{\text{w}}(\mathbf{x}^{\text{u}}); \theta^t)\right)\end{aligned} \tag{9.32}$$

[5] A more complicated instance-dependent noise model $\mathcal{T}(\hat{Y}|Y, X; \omega)$ can also be formulated under our unified framework, but not considered in this work. Also, since we use $\mathcal{T}$ both in forward fashion and backward fashion, it is unidentifiable in this work.

Table 9.1: Accuracy of different partial ratio q on CIFAR-10, CIFAR-100, and CUB-200 for **partial label learning**. The best and the second best results are indicated in **bold** and underline respectively.

Dataset	CIFAR-10			CIFAR-100			CUB-200
Partial Ratio q	0.1	0.3	0.5	0.01	0.05	0.1	0.05
Fully-Supervised		94.91±0.07			73.56±0.10		-
LWS [184]	90.30±0.60	88.99±1.43	86.16±0.85	65.78±0.02	59.56±0.33	53.53±0.08	39.74±0.47
PRODEN [222]	90.24±0.32	89.38±0.31	87.78±0.07	62.60±0.02	60.73±0.03	56.80±0.29	62.56±0.10
CC [182]	82.30±0.21	79.08±0.07	74.05±0.35	49.76±0.45	47.62±0.08	35.72±0.47	55.61±0.02
MSE [245]	79.97±0.45	75.65±0.28	67.09±0.66	49.17±0.05	46.02±1.82	43.81±0.49	22.07±2.36
EXP [245]	79.23±0.10	75.79±0.21	70.34±1.32	44.45±1.50	41.05±1.40	29.27±2.81	9.44±2.32
PiCO [2]	94.39±0.18	94.18±0.12	93.58±0.06	73.09±0.34	72.74±0.30	69.91±0.24	**72.17±0.72**
Ours	**96.37±0.08**	**96.26±0.03**	**95.91±0.05**	**75.31±0.19**	**74.58±0.03**	**74.00±0.02**	70.77±0.29

We can compute both quantity through the noise transition model:

$$\mathbf{p}(y|\mathbf{x},\hat{\mathbf{s}};\theta,\omega^t) \propto \mathbf{p}(y|\mathbf{x};\theta)\prod_{\hat{y}\in\hat{\mathbf{s}}}\mathcal{T}(y|\hat{y};\omega^t), \text{and } \mathbf{p}(\hat{y}|\mathbf{x};\theta,\omega) = \sum_{y\in[C]}\mathbf{p}(y|\mathbf{x};\theta)\mathcal{T}(\hat{y}|y;\omega). \tag{9.33}$$

9.8 Experiments

In this section, we conduct extensive experiments to evaluate ILL. Albeit simple, the ILL framework achieves comparable state-of-the-art performance regarding previous methods on partial label learning, semi-supervised learning, and noisy label learning. Moreover, our experiments show that ILL could be easily extended to a more practical setting with a mixture of various imprecise label configurations. For all settings, we additionally adopt an entropy loss for balancing learned cluster sizes [241, 242], similarly as [4, 193]. Experiments are conducted with three runs using NVIDIA V100 GPUs.

9.8.1 Partial Label Learning

Setup. Following [2], we evaluate our method on partial label learning setting using CIFAR-10 [243], CIFAR-100 [243], and CUB-200 [244]. We generate partially labeled datasets by flipping negative labels to false positive labels with a probability q, denoted as a partial ratio. The $C-1$ negative labels are then uniformly aggregated into the ground truth label to form a set of label candidates. We consider $q \in \{0.1, 0.3, 0.5\}$ for CIFAR-10, $q \in \{0.01, 0.05, 0.1\}$ for CIFAR-100, and $q = 0.05$ for CUB-200. We choose six baselines for PLL using ResNet-18 [174]: LWS [184], PRODEN [222], CC [182], MSE and EXP [245], and PiCO [2]. The detailed hyper-parameters, comparison with the more recent method R-CR [185] that utilizes a different training recipe and model [246], and comparison with instance-dependent partial labels [247] are shown in Appendix D.4.2.2.

Results. The results for PLL are shown in Table 9.1. Our method achieves the

Table 9.2: Error rate of different number of labels l on CIFAR-100, STL-10, IMDB, and Amazon Review datasets for **semi-supervised learning**.

Datasets	CIFAR-100		STL-10		IMDB		Amazon Review	
# Labels l	200	400	40	100	20	100	250	1000
AdaMatch [250]	22.32±1.73	16.66±0.62	13.64±2.49	7.62±1.90	8.09±0.99	7.11±0.20	45.40±0.96	**40.16±0.49**
FixMatch [3]	29.60±0.90	19.56±0.52	16.15±1.89	8.11±0.68	7.72±0.33	7.33±0.13	47.61±0.83	43.05±0.54
FlexMatch [191]	26.76±1.12	18.24±0.36	14.40±3.11	8.17±0.78	7.82±0.77	7.41±0.38	45.73±1.60	42.25±0.33
CoMatch [251]	35.08±0.69	25.35±0.50	15.12±1.88	9.56±1.35	7.44±0.30	7.72±1.14	48.76±0.90	43.36±0.21
SimMatch [252]	23.78±1.08	17.06±0.78	11.77±3.20	**7.55±1.86**	7.93±0.55	**7.08±0.33**	45.91±0.95	42.21±0.30
FreeMatch [193]	**21.40±0.30**	**15.65±0.26**	12.73±3.22	8.52±0.53	8.94±0.21	7.95±0.45	46.41±0.60	42.64±0.06
SoftMatch [194]	22.67±1.32	16.84±0.66	13.55±3.16	7.84±1.72	7.76±0.58	7.97±0.72	45.29±0.95	42.21±0.20
Ours	22.06±1.06	16.40±0.54	**11.09±0.71**	8.10±1.02	**7.32±0.12**	7.64±0.67	**43.96±0.32**	42.32±0.02

best performance compared to the baseline methods. Perhaps more surprisingly, on CIFAR-10 and CIFAR-100, our method even outperforms the fully-supervised reference, indicating the potential better generalization capability using the proposed framework, sharing similar insights as in Wu et al. [185]. While PiCO adopts a contrastive learning objective, our method still surpasses PiCO by an average of **2.13**% on CIFAR-10 and **2.72**% on CIFAR-100. Our approach can be further enhanced by incorporating contrastive learning objectives, potentially leading to more significant performance.

9.8.2 Semi-Supervised Learning

Setup. For experiments of SSL, we follow the training and evaluation protocols of USB [5] on image and text classification. To construct the labeled dataset for semi-supervised learning, we uniformly select l/C samples from each class and treat the remaining samples as the unlabeled dataset. We present the results on CIFAR-100 and STL-10 [243] for image classification, and IMDB [248] and Amazon Review [249] for text classification. We compare with the current methods with confidence thresholding, such as FixMatch [3], AdaMatch [250], FlexMatch [191], FreeMatch [193], and SoftMatch [194]. We also compare with methods with the contrastive loss, CoMatch [251] and SimMatch [252]. A full comparison of the USB datasets and hyper-parameters is shown in Appendix D.4.3.

Results. We present the results for SSL on Table 9.2. Although no individual SSL algorithm dominates the USB benchmark [5], our method still shows competitive performance. Notably, our method performs best on STL-10 with 40 labels and Amazon Review with 250 labels, outperforming the previous best by **0.68**% and **1.33**%. In the other settings, the performance of our method is also very close to the best-performing methods. More remarkably, our method does not employ any thresholding, re-weighting, or contrastive techniques to achieve current results, demonstrating a significant potential to be further explored.

Table 9.3: Accuracy of synthetic noise on CIFAR-10 and CIFAR-100 and instance noise on Clothing1M and WebVision for **noisy label learning**. We use noise ratio of $\{0.2, 0.5, 0.8\}$ for synthetic symmetric noise and 0.4 for asymmetric label noise. The instance noise ratio is unknown.

Dataset	CIFAR-10				CIFAR-100				Clothing1M	WebVision
Noise Type	Sym.			Asym.	Sym.			Asym.	Ins.	Ins.
Noise Ratio η	0.2	0.5	0.8	0.4	0.2	0.5	0.8	0.4	-	-
CE	87.20	80.70	65.80	82.20	58.10	47.10	23.80	43.30	69.10	-
Mixup [255]	93.50	87.90	72.30	-	69.90	57.30	33.60	-	-	-
DivideMix [203]	96.10	94.60	93.20	93.40	77.10	74.60	60.20	72.10	**74.26**	77.32
ELR [202]	95.80	94.80	93.30	93.00	77.70	73.80	60.80	77.50	72.90	76.20
SOP [4]	96.30	95.50	94.00	93.80	**78.80**	**75.90**	63.30	**78.00**	73.50	76.60
Ours	**96.78**$\pm$0.11	**96.60**$\pm$0.15	**94.31**$\pm$0.07	**94.75**$\pm$0.81	77.49$\pm$0.28	75.51$\pm$0.52	**66.46**$\pm$0.72	75.82$\pm$1.89	74.02$\pm$0.12	**79.37**$\pm$0.09

9.8.3 Noisy Label Learning

Setup. We conduct the experiments of NLL following SOP [4] on both synthetic symmetric/asymmetric noise on CIFAR-10 and CIFAR-100, and more realistic and larger-scale instance noise on Clothing1M [253], and WebVision [254]. To introduce the synthetic symmetric noise to CIFAR-10 and CIFAR-100, we uniformly flip labels for a probability η into other classes. For asymmetric noise, we only randomly flip the labels for particular pairs of classes. We mainly select three previous best methods as baselines: DivideMix [203]; ELR [202]; and SOP [4]. We also include the normal cross-entropy (CE) training and mixup [255] as baselines. More comparisons of other methods [256, 198] and on CIFAR-10N [257] with training details and more baselines [258, 198] are shown in Appendix D.4.4.

Results. We present the noisy label learning results in Table 9.3. The proposed method is comparable to the previous best methods. On synthetic noise of CIFAR-10, our method demonstrates the best performance on both symmetric noise and asymmetric noise. On CIFAR-100, our method generally produces similar results comparable to SOP. One may notice that our method shows inferior performance on asymmetric noise of CIFAR-100; we argue this is mainly due to the oversimplification of the noise transition model. Our method also achieves the best results on WebVision, outperforming the previous best by **2.05**%. On Clothing1M, our results are also very close to DivideMix, which trains for 80 epochs compared to 10 epochs in ours.

9.8.4 Mixed Imprecise Label Learning

Setup. We evaluate on CIFAR-10 and CIFAR-100 in a more challenging and realistic setting, the mixture of various imprecise label configurations, with unlabeled, partially labeled, and noisy labeled data existing simultaneously. We first sample the labeled dataset and treat other samples as the unlabeled. On the labeled dataset, we generate partial labels and randomly corrupt the true label of the partial labels. We set

Table 9.4: Accuracy comparison of **mixture of different imprecise labels**. We report results of full labels, partial ratio q of 0.1 (0.01) and 0.3 (0.05) for CIFAR-10 (CIFAR-100), and noise ratio η of 0.1, 0.2, and 0.3 for CIFAR-10 and CIFAR-100.

Method	q	CIFAR-10, l=50000			q	CIFAR-100, l=50000		
		η=0.1	η=0.2	η=0.3		η=0.1	η=0.2	η=0.3
PiCO+ [259]	0.1	93.64	93.13	92.18	0.01	71.42	70.22	66.14
IRNet [237]		93.44	92.57	92.38		71.17	70.10	68.77
DALI [218]		94.15	94.04	93.77		72.26	71.98	71.04
PiCO+ Mixup [218]		94.58	94.74	94.43		75.04	74.31	71.79
DALI Mixup [218]		95.83	95.86	95.75		76.52	76.55	76.09
Ours		**96.47**$\pm$0.11	**96.09**$\pm$0.20	**95.83**$\pm$0.05		**77.53**$\pm$0.24	**76.96**$\pm$0.02	**76.43**$\pm$0.27
PiCO+ [259]	0.3	92.32	92.22	89.95	0.05	69.40	66.67	62.24
IRNet [237]		92.81	92.18	91.35		70.73	69.33	68.09
DALI [218]		93.44	93.25	92.42		72.28	71.35	70.05
PiCO+ Mixup [218]		94.02	94.03	92.94		73.06	71.37	67.56
DALI Mixup [218]		95.52	95.41	94.67		76.87	75.23	74.49
Ours		**96.2**$\pm$0.02	**95.87**$\pm$0.14	**95.22**$\pm$0.06		**77.07**$\pm$0.16	**76.34**$\pm$0.08	**75.13**$\pm$0.63

$l \in \{1000, 5000, 50000\}$ for CIFAR-10, and $l \in \{5000, 10000, 50000\}$ for CIFAR-100. For partial labels, we set $q \in \{0.1, 0.3, 0.5\}$ for CIFAR-10, and $q \in \{0.01, 0.05, 0.1\}$ for CIFAR-100. For noisy labels, we set $\eta \in \{0, 0.1, 0.2, 0.3\}$ for both datasets. Since there is no prior work that can handle all settings all at once, we compare on partial noisy label learning with PiCO+ [259], IRNet [237], and DALI [218]. Although there are also prior efforts on partial semi-supervised learning [219, 220], they do not scale on simple dataset even on CIFAR-10. Thus, we did not include them in comparison. We conduct additional validation of our method on more complex settings for partial noisy labels with unlabeled data to demonstrate its robustness to various imprecise labels.

Results. We report the comparison with partial noisy label learning methods in Table 9.4. Compared to previous methods, the proposed method achieves the best performance. Despite the simplicity, our method outperforms PiCO+ and DALI with mixup, showing the effectiveness of dealing with mixed imprecise labels. We also report the results of our methods on more mixed imprecise label configurations in Table 9.5. Our method demonstrates significant robustness against various settings of the size of labeled data, partial ratio, and noise ratio. Note that this is the first work that naturally deals with all three imprecise label configurations simultaneously, with superior performance than previous methods handling specific types or combinations of label configurations. This indicates the enormous potential of our work in realistic applications for handling more practical and complicated data annotations common in real world applications.

9.9 Results for Imprecise Label Learning on Audio Analysis

In addition to the vision and text domains, the proposed ILL framework can also be applied to audio analysis tasks, where annotations often come in the form of noisy,

Table 9.5: Robust test accuracy results of our method on **more mixture of imprecise label configurations**. l, q and η are the number of labels, partial, and noise ratio.

l	q	CIFAR10 η=0.0	CIFAR10 η=0.1	CIFAR10 η=0.2	CIFAR10 η=0.3	l	q	CIFAR100 η=0.0	CIFAR100 η=0.1	CIFAR100 η=0.2	CIFAR100 η=0.3
5,000	0.1	95.29±0.18	93.90±0.11	92.02±0.22	89.02±0.63	10,000	0.01	69.90±0.23	68.74±0.15	66.87±0.34	65.34±0.02
	0.3	95.13±0.16	92.95±0.37	90.14±0.61	87.31±0.27		0.05	69.85±0.20	68.08±0.28	66.78±0.43	64.83±0.17
	0.5	95.04±0.10	92.18±0.52	88.39±0.62	83.09±0.56		0.10	68.92±0.45	67.15±0.63	64.44±1.29	60.26±1.96
1,000	0.1	94.48±0.09	91.68±0.17	87.17±0.51	81.04±1.13	5,000	0.01	65.66±0.27	63.13±0.27	60.93±0.17	58.36±0.56
	0.3	94.35±0.05	89.94±1.90	82.06±1.52	69.20±2.16		0.05	65.06±0.04	62.28±0.47	58.92±0.34	53.24±1.69
	0.5	93.92±0.29	86.34±2.37	70.86±2.78	38.19±6.55		0.10	63.32±0.55	58.73±1.33	53.27±1.57	46.19±1.04

partial, or weak labels. For example, in large-scale sound event detection (SED) scenarios, collecting perfectly accurate labels for every time frame is expensive and prone to error. Instead, annotators might provide bag-level labels (indicating that a sound is present in at least one segment), partial labels (specifying a small candidate set of possible events), or noisy labels (due to human errors or automatic labeling pipelines).

When applying ILL to audio tasks, key points include:

1. **Temporal Segmentation and Aggregation**: Audio data is a time series data consisting of time segments. The ILL framework can work at the segment level, allowing the model to consider multiple possible label assignments for each segment and integrate them into a consistent posterior distribution.

2. **Noise Modeling in Audio Labels**: Just as in image or text data, noisy labels can be modeled with a noise transition matrix. The ILL framework iteratively refines the estimation of this matrix, thus improving the quality of inferred true labels, even if the initial annotations are imprecise.

3. **Partial Label Handling**: Partial labels in audio may arise when annotators are uncertain which of several candidate events occurs in a given segment. ILL can model the set of candidate labels as a probability distribution, eliminating the need for a single hard guess and reducing the risk of error propagation.

4. **Unlabeled and Weakly-Labeled Data**: Large-scale audio corpora often include unlabeled audio or audio with weak labels that only specify coarse attributes (e.g., presence of a sound class somewhere in a long recording). ILL can handle these cases by treating them as latent label distributions and leveraging EM steps to naturally integrate them into the training process.

As demonstrated in the table below, applying the ILL framework to a variety of audio datasets and tasks—ranging from general audio classification to sound event detection—yields competitive performance. More importantly, it maintains the flexibility

and robustness to handle various imprecise label conditions that are common in audio analysis scenarios.

Task	Dataset	Base Model	Bag Length	mAP
Audio Classification	FSDNoisy18k	Distill-Hubert	10s	68.24
Audio Classification	ESC-50	Hubert-Base	10s	87.25
Audio Classification	AudioSet	Hubert-Large	5s	25.2*
Sound Event Detection	DESED	Distill-Hubert	10s	58.17
Partial Label (SED)	DESED	Distill-Hubert	10s	44.2
Noisy Label (SED)	DESED	Distill-Hubert	10s	52.17 (10%) 49.42 (20%) 37.07 (50%)

Table 9.6: Summary of results on various audio tasks under imprecise labeling conditions.

9.10 Conclusion

We present the imprecise label learning (ILL) framework, a unified and consolidated solution for learning from all types of imprecise labels. ILL effectively employs an expectation-maximization (EM) algorithm for maximum likelihood estimation (MLE) of the distribution over the latent ground truth labels Y, imprecise label information I, and data X. It naturally extends and encompasses previous formulations for various imprecise label settings, achieving promising results. Notably, in scenarios where mixed configurations of imprecise labels coexist, our method exhibits substantial robustness against diverse forms of label imprecision. The potential **broader impact** of the ILL framework is substantial. It stands poised to transform domains where obtaining precise labels poses a challenge, offering a simple, unified, and effective approach to such contexts. Beyond the three imprecise label configurations we have demonstrated in this study, the ILL framework shows promise for an extension to more intricate scenarios such as multi-instance learning [210] and multi-label crowd-sourcing learning [206]. However, it is also crucial to acknowledge the **limitations** of the ILL framework. Although its effectiveness has been substantiated on relatively smaller-scale datasets, additional empirical validation is necessary to assess its scalability to larger datasets. Furthermore, our study only considers balanced datasets; thus, the performance of the ILL framework when dealing with imbalanced data and open-set data still remains an open area for future exploration. We hope that our study will constitute a significant stride towards a comprehensive solution for imprecise label learning and catalyze further research in this crucial field. [6]

[6]Code is available at https://github.com/Hhhhhhao/General-Framework-Weak-Supervision

10

Future work and Discussion

The advancements presented in this thesis lay a solid foundation for addressing the challenges of computational audition, particularly in scenarios involving weakly labeled or noisy data. However, the rapidly evolving landscape of machine learning and audio processing presents numerous opportunities to extend and enhance the methodologies developed herein. This chapter outlines several promising directions for future research, building upon the strategies and findings of this work. These avenues aim to further improve the robustness, scalability, and applicability of computational audio systems in diverse and complex environments.

10.1 Enhancing Learning from Weak and Imprecise Labels

10.1.1 Refining Weak Label Learning Strategies

Although weak labeling significantly reduces the human effort required to build large datasets, there remains potential to further enhance learning from such labels. Future research can focus on:

- **Hierarchical Labeling Approaches**: Exploring hierarchical labeling systems, where sound events are organized into categories and subcategories, can provide richer supervisory signals. Hierarchical learning strategies can leverage the

relationships between different levels of labels to improve model performance, especially in complex acoustic environments.

- **Dynamic Label Refinement**: Implementing dynamic label-refinement processes, where model predictions iteratively refine the weak labels, can enhance label quality over time. Techniques such as expectation-maximization (EM) and iterative self-training can be applied to progressively improve label accuracy during training.

10.1.2 Integrating Additional Cues and Semi-Weak Labels

Incorporating easily obtainable additional information, such as event counts, durations, or coarse temporal locations, can strengthen weak labels. Future work may involve:

- **Exploiting Semi-Weak Labels**: Developing models that can effectively utilize semi-weak labels to bridge the gap between weak and strong labels without significant annotation overhead.
- **Multi-Instance Learning**: Applying multi-instance learning frameworks to handle bags of instances with associated labels, enabling the model to learn from sets of audio segments labeled at a higher level of abstraction.

10.2 Leveraging Unlabeled Data

10.2.1 Advancing Unsupervised and Semi-Supervised Learning

Given the abundance of unlabeled audio data, future research can focus on methods to exploit this resource:

- **Self-Supervised Learning Paradigms**: Investigate self-supervised learning techniques that leverage inherent structures within audio data (e.g., contrastive learning, masked audio modeling) to pre-train models that require minimal labeled data for fine-tuning.
- **Generative Models for Label Synthesis**: Employ generative models, such as Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs), to synthesize realistic labels or augment existing datasets, thereby expanding the training data available for supervised learning tasks.

10.3 Integration of Ontologies and Hierarchical Information

Ontologies provide valuable semantic relationships between sound events. Future work can focus on:

- **Ontology-Infused Model Architectures**: Incorporate ontological knowledge directly into model architectures, enabling the exploitation of hierarchical relationships between sound classes.

- **Dynamic Ontology Construction**: Develop methods for dynamic, data-driven ontology construction that can adapt as new data or classes are encountered.

- **Ontology-Guided Regularization**: Implement regularization techniques that penalize predictions violating known ontological relationships, improving model consistency with domain knowledge.

10.4 Advanced Deep Learning Architectures

10.4.1 Exploring Transformer-Based Models and alternate architectures

Transformers have shown great success in natural language processing and have shown their potential in audio processing [260]. However, challenges remain in their application to audio.

The primary challenges are **High Computational Cost**: The quadratic complexity of self-attention with respect to sequence length makes Transformers resource-intensive for long audio sequences. This is particularly problematic for real-time processing or devices with limited computational capacity. **Data Inefficiency**: Transformers typically require vast amounts of labeled data, which is often scarce in specialized audio domains. **Sequence Length Limitations**: The fixed input size of Transformers necessitates strategies like chunking or truncation, which can lead to loss of contextual information in variable-length audio inputs. **Inefficient Local Feature Extraction**: Unlike CNNs, Transformers do not inherently capture local patterns efficiently, which are crucial for audio tasks where local features like timbre and transients are important. **Overhead of Positional Encoding**: The temporal nature of audio is not as naturally aligned with the positional encodings used in Transformers, compared to the recurrence in RNNs or convolutions in CNNs.

- **Efficient Transformer Architectures**: Research into transformer architectures optimized for audio, addressing issues such as high computational cost and sequence length limitations, can make them more applicable to audio tasks.

- **Hybrid Models**: Develop hybrid architectures that combine transformers with convolutional neural networks (CNNs) or recurrent neural networks (RNNs) to capture both local and global patterns in audio data.

10.4.2 Adapting Models for Real-Time Processing

For applications requiring low latency:

- **Model Optimization**: Explore techniques for model compression and optimization, such as quantization and pruning, to enable deployment on devices with limited computational resources.
- **Streaming Architectures**: Design architectures that can process audio data in a streaming fashion, allowing real-time analysis without the need to process the entire input sequence at once.

10.5 Multimodal and Cross-Modal Learning

10.5.1 Integrating Multimodal Data

Sound perception often occurs alongside other sensory inputs. Future research can focus on:

- **Multimodal Representations**: Develop unified representations that integrate audio with visual and textual information, facilitating more comprehensive understanding.
- **Cross-Modal Learning**: Investigate learning techniques that leverage correspondence between modalities, enhancing capabilities such as sound localization and event recognition.

10.6 Real-World Applications and Ethical Considerations

10.6.1 Deployment in Diverse Environments

To ensure models generalize well across different settings:

- **Domain Adaptation and Generalization**: Develop methods to adapt models to new domains with minimal retraining, possibly through domain-invariant feature learning or transfer learning.
- **Robustness to Environmental Variability**: Enhance model robustness to variations in recording conditions, noise levels, and device characteristics.

10.6.2 Ethical and Privacy Concerns

Models may reinforce biases or violate privacy, particularly in surveillance or healthcare. As audio analysis systems become more pervasive:

- **Privacy-Preserving Techniques**: Implement methods such as federated learning and differential privacy to protect user data while maintaining model performance for audio understanding systems and leveraging the power of imprecise labels.
- **Bias Mitigation**: Ensure datasets and models are free from biases that could lead to unfair or discriminatory outcomes.

Scalable and Generalizable Models

Challenges: Current models often lack scalability across diverse datasets and environments.

- Develop *foundation models* for audio, analogous to GPT models in NLP, capable of generalizing across multiple audio tasks such as sound event detection and audio classification.
- Invest in *self-supervised learning* techniques to utilize large-scale unlabeled datasets, reducing dependence on annotated data.
- Explore *domain generalization* methods to create robust models for unseen conditions without requiring domain-specific fine-tuning.

10.7 Conclusion

The future of computational audition is rich with potential, driven by advancements in machine learning, data availability, and interdisciplinary collaboration. By addressing the challenges outlined in this chapter, researchers can further enhance the ability of machines to interpret and understand complex acoustic environments. The strategies and frameworks developed in this thesis provide a stepping stone towards more intelligent, resilient, and versatile audio processing systems. As the field progresses, continued innovation and exploration will be essential to unlock the full capabilities of computational audition, ultimately enabling machines to interact with the world of sounds in ways that mirror, and even surpass, human auditory perception.

Appendix

A

Unsupervised Test Set Adaptation

Deep neural networks (DNNs) can achieve high accuracy when trained and tested on data drawn from the same distribution. However, it is common for test data to differ in subtle ways from the training data, leading to performance degradation. Domain adaptation techniques often focus on bridging the distribution gap between source and target domains, but these methods assume that the target domain data are available during training. In contrast, we focus on a scenario where only unlabeled test data are available at test time, and the test distribution itself may differ from that of the training data.

Prior work such as Black Box Shift Estimation (BBSE) [261] tackles shift by reweighting predictions at test time. In this chapter, we extend this concept by introducing an affine transformation that adapts the network's predictions or embeddings to better fit the test data distribution. The transformation parameters are learned by minimizing a divergence measure between the network's predictions and pseudo-labels derived from the test data, thus establishing a link between the distributions of the train and test sets. Applying this affine transformation at test time makes our approach model- and data-agnostic, enabling easy adoption across a wide range of tasks.

A.1 Using Affine Transformation for Test time adaptation

The generalization of a trained deep neural network (DNN) to a test dataset can be a challenging problem due to the distribution shift between the training and test data. To address this issue, domain adaptation methods have been developed to enable DNNs to generalize from a source domain to a target domain. In this study, we focus on the problem of unsupervised adaptation of DNNs to a test dataset with no labels. Our model is based on the recent work of Varsavsky et al. [2020] and hypothesizes that the train and test data sets form an affine transformation.

We propose two methods for applying an affine transformation to the DNN in order to improve its generalization to the test data. In the first method, the affine transformation is applied after the final linear layer, resulting in a transformed output $\hat{Z}$. The transformation parameters A and b are learned by minimizing the average divergence between $\sigma(\hat{Z})$ and $y(X_i)$ for different batch sizes, as shown in Eq (1). The range is from 1 to N_T, where N_T is the total number of test samples.

$$\arg\min_{A,b} \sum_{i=1}^{N_T} div(\sigma(\hat{Z}) - y(X_i)) \tag{A.1}$$
$$\text{s.t } \hat{Z} = AZ + b$$

In the second method, the affine transformation is applied before the final linear layer on the latent features or embeddings R, resulting in a transformed output $\hat{R}$. The transformation parameters A and b are again learned by minimizing the average divergence between $\sigma(\hat{R})$and $y(Xi)$ for different batch sizes, as shown in Eq (2).

$$\arg\min_{A,b} \sum_{i=1}^{N_T} div(\sigma(\hat{R}) - y(X_i)) \tag{A.2}$$
$$\text{s.t } \hat{R} = AR + b$$

We apply the affine transformation using two methodologies to align the distribution of the test data with that of the train data, thereby enhancing the DNN's generalization ability to the test dataset. In both scenarios, the transformation parameters A and b are acquired by minimizing the divergence between the transformed output and the network's predictions on the test data. This unsupervised adaptation technique is both flexible and data-independent, permitting a broad range of potential mappings between the train and test data distributions, and is readily applicable to any task. We anticipate that our proposed techniques will markedly boost the DNN's performance on the test dataset.

A.2 Literature Review

A.2.1 Test-time Unsupervised Domain Adaptation

Varsavsky et al. [262] proposed an approach to unsupervised domain adaptation (UDA) for transfer learning in medical imaging. They found that the misalignment between the source and target domains led to a decision boundary that was not generalizable to the target domain. To address this issue, they used test-time adaptation and adversarial learning to align the source and target domains using a domain discriminator. This transformed the decision boundary from the source domain to the target domain.

To address this issue, the authors proposed an evaluation framework for UDA that performs test-time UDA on each subject separately, rather than measuring the model's ability to generalize to unseen data in the target domain. This approach, called "one-shot UDA," is shown to outperform a UDA method that has seen more data from the target domain but not the specific target subject. The authors also proposed a UDA method based on adversarial learning and consistency under augmentation, which is applied to multiple sclerosis lesion segmentation. This method is designed to be applicable to other tasks in medical imaging.

Overall, the authors' approach offers a promising way to improve the generalization of CNNs in medical imaging, particularly when labeled data is not available in the target domain. However, the authors did not consider that the test set itself may have a different distribution than what the model was trained on, which is an important factor in ensuring the model's performance. Accounting for the distribution mismatch at test time is important since even with domain adaptation from source to target, the model itself has not adapted to the target's test set.

A.2.2 Test Entropy Minimization: Tent

Wang et al. [263] propose test entropy minimization (Tent), a method for fully unsupervised test-time adaptation. Tent reduces prediction uncertainty by minimizing entropy on test predictions, thereby improving generalization. Their results on the ImageNet-C dataset show substantial improvements over robust training baselines.

While Tent focuses on entropy minimization, our approach uses an affine transformation that can incorporate various divergence measures. Our method complements ideas like Tent, as both aim to refine model outputs at test time without access to labeled data.

A.2.3 Blackbox Shift Estimation (BBSE)

BBSE [261] addresses label shift by reweighting predictions to correct for distributional differences between training and test sets. It relies on assumptions that the relationship between features and labels remains stable. BBSE can achieve improved target accuracy under label shift scenarios.

This corresponds to the task of anticausal learning, or predicting causes, which is relevant in contexts such as disease diagnosis. The authors contribute to the development of BBSE by analyzing its properties, including its consistency and error bounds, and by demonstrating its use in statistical tests to detect label shift among distributions. They also show how BBSE can be corrected through the use of important-weighted empirical risk minimization (EMR) and provide empirical validation of the method. BBSE has several advantages, including data dimensionality-independent accuracy, the ability to work with a variety of predictors, including those that may be biased or inaccurate, low sample complexity, and generalizability to arbitrary machine learning models.

The Black Box Shift Estimation (BBSE) method exhibits a decrease in the estimation error and improved target accuracy as the concentration parameter α increases in the simulations of the label shift using Dirichlet shift. The largest feasible value for the KMM experiment was 8000 due to computational limitations. BBSE, along with important weighted empirical risk minimization (EMR), offers a promising approach for detecting and correcting distribution shift between training and test sets in order to improve classification accuracy. The method involves estimating $w(y)$, defined as the product of the target distribution $q(y)$ and the source distribution $p(y)$, and using it in the EMR framework for the correction of the change. This estimation relies on the assumptions of label shift, that the training data include all relevant classes, and that the confusion matrix is invertible (i.e. the outputs of each class are linearly independent).

A.3 Dataset

We will utilize large-scale sound event datasets such as AudioSet [38] for our experiments. While state-of-the-art deep learning models have achieved near-perfect accuracy (approximately 98–99%) on benchmark image datasets like MNIST [264] and CIFAR [177], these results leave little room to discern the effectiveness of different learning strategies. In contrast, AudioSet is considerably more challenging: despite recent advances, the highest reported mean average precisions (mAP) remain relatively low, at 0.521 [265] and 0.567 [260]. This performance gap makes it easier to observe meaningful differences arising from various training methodologies.

In our work, we focus specifically on the balanced training subset of AudioSet. On this subset, the Audio Spectrogram Transformer (AST) model achieves a 0.347 mAP without weight averaging, which improves to 0.378 mAP when weight averaging is employed. Thus, the differences in performance driven by different learning strategies become more evident on this challenging dataset.

A.4 Baseline Model

We use the Audio Spectrogram Transformer (AST) architecture by Yuan Gong Gong et al. [260] as our primary baseline (https://github.com/ YuanGongND/ast). This model achieves new state-of-the-art results of 0.485 mAP on AudioSet, and easily allows us to load pretrained weights before the final sigmoid activation layer.

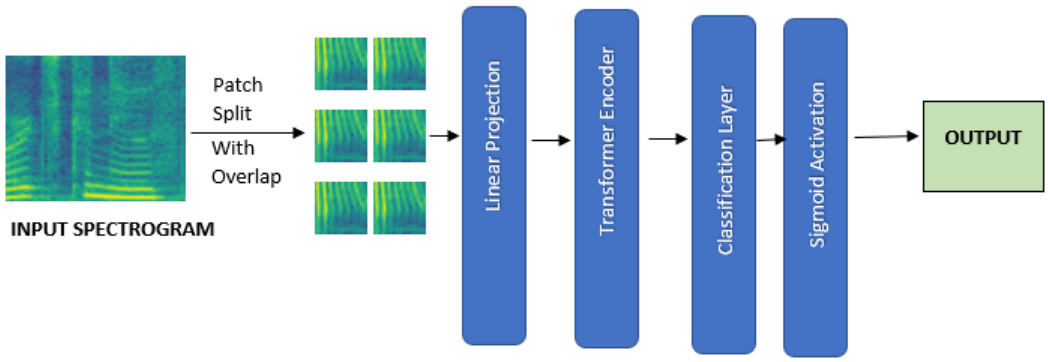


Figure A.1: AST Baseline Model

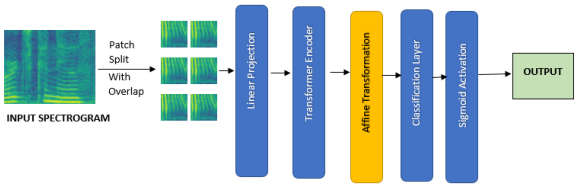


Figure A.2: AST Adaptation on R

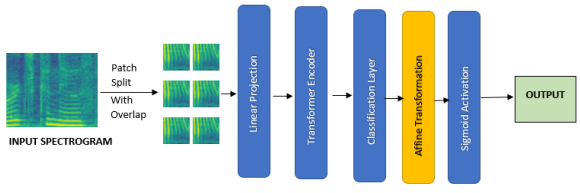


Figure A.3: AST Adaptation on Z

We modify the architecture to obtain both the model outputs (Z) and embeddings before the classification layer (R). Without the affine transformation, the mAP score that was achieved from the pretrained model is 0.449 according to the paper. We implemented the pipeline and got a mAP score of 0.445. This indicates that our baseline implementation is correct.

A.5 Experiments

A.5.1 Learning Strategies

We experiment with the following strategies.

1. Type of Weight initialization (Identity vs Xavier).
2. Adaptation on Z vs Adaptation on R.

To perform adaptation, we freeze the AST model's weights and define a wrapper class that defines an affine Layer on top of the AST model. We pass the inputs through the AST model with pre-trained weights to obtain Z or R depending on our adaptation strategy, and then train the affine Layer with the test set pseudo-labels for the unsupervised case. For the supervised case, we use the ground truth labels of the test set to obtain an upper bound on generalization capability of the model to the test set.

For each of these strategies, we try to learn the affine transformation for two types of weight matrices - diagonal vs full matrix. For a $N \times N$ matrix A where N refers to the length of the embedding to which we apply the affine transformation, we have:

- If A is a diagonal matrix, we learn N parameters.
- If A is a full matrix, we learn N^2 parameters.

We create pseudo-labels for the multi-label examples by using a comparison threshold of 0.7 on the AST model's logits, and used the top 8 resulting candidates as the pseudo-labels for each example.

For the diagonal matrix case, we register a hook during the backward pass for the affine layer's weight matrix to be multiplied with an identity mask that preserves only the diagonal entries of the weight matrix, thus forcing the weight matrix of the affine layer to be diagonal throughout adaptation.

A.6 Results

We ran adaptation on both the true labels ytrue and predicted labels ypred. ypred is obtained by passing the test set through the model, and taking the argmax of the output logits from the model. By running adaptation on ytrue, we can get the mAP upper bound, which we can compare with adaptation on ypred. We refer to the adaptation on ytrue as Supervised Adaptation, and adaptation on ypred as Unsupervised Adaptation.

The acoustic context could be present both in the form of the acoustic background acoustic *textures* (such as a noisy outdoor environment, an indoor recording environment, or the background sound of a running car), long-duration acoustic backgrounds, e.g. the sound of a prolonged boat horn, with definitive start and end times, and short-duration sounds such as gunshots.

Learning to automatically detect sounds of this variety faces the following challenges:

- Diversity: The diversity of sound types to be recognized, which as discussed above can range from continuous background textures to sharp, short-duration sounds, represent very different characteristics, potentially requiring very different types of analyses.

- Labelling: Traditional learning frameworks for AI models are based on "strong labels" – annotations that explicitly identify if an "instance" belongs to a target class or not. This runs into multiple problems in the audio setting. Firstly, the very definition of "instance" is fuzzy – a 30 second recording could entirely match a given background acoustic class (e.g. "indoors"), in which case the entire recording is an instance of that class, whereas the same recording could also contain a brief segment with a gunshot, where it is only reasonable to consider that segment of the audio as an instance of "gunshot", whereas the rest is clearly not. Secondly, even where instances may be considered to be brief and clearly demarcated, actually labelling the data at the requisite temporal resolution can be challenging and imprecise and, not to mention, expensive.

- The number and variety of sound types that can occur in the world is very large, and accordingly the number of sound types that must be detected in the context of voice forensics too can be very large. Often the sound types of particular interest in this setting are also rare. Thus, collecting sufficient training data to model all of the well can be challenging or impossible.

To solve these problems, we used the following approach.

Class diversity

As the number of potential sound categories that must be recognized is very large, it is not possible to obtain sufficient training data to model all of them accurately. Within the forensic setting, although any single situation may require the identification of only a few sound categories, the amount of training data for those categories is typically small. Furthermore, even if a relatively significant amount of data were made available for each of these classes, the *counter* classes – the sound types that do *not* belong

to the target classes, remains impossibly large. Effective detection of a class requires knowledge not only of what the class *does* sound like, but also of what it *does not* sound like, and hence these classes must also be represented during training.

To deal with this our framework uses as its preliminary stage a state-of-art *universal representation* of sound, a feature representation that ideally captures the salient characteristics of *all* sound types. This representation is derived by a neural network that is trained on a sufficiently large number of sound categories, so that the features it derives may be expected to generalize beyond the classes used to train it. The network itself must specifically be trained for such generalization.

In order to fulfill these requirements, we use a state-of-art sound feature extraction system trained on a large number of classes and data.In our work here we have specifically used the embedding extraction framework from the AST by Yuan Gong Gong et al. [260] and BEATS [266] models, the two leading architectures currently for sound feature extraction and classification. The Audio Spectrogram Transformer (AST) architecture was chosen as our primary baseline (https://github.com/YuanGongND/ast). This model is trained on "AudioSet". AST achieves state-of-the-art results of 0.485 mAP on the testset of "AudioSet". The BEATS model treats sound as a language comprised of automatically derived subunits, and learns representations that best model it. BEATS representations can achieve an mAP of 0.506 on the AudioSet.

For our work we treat these feature extractors as immutable, and focus on the downstream task of detecting the target classes for our forensic setting.

Labelling ambiguity

The natural diversity of the sound types also introduces labelling ambiguity, as explained above. A 10-second audio clip of wind blowing represents a single instance of that class in its entirety. On the other hand, a 10-second clip with a gunshot in it includes one instance of a gunshot, but also an undefined number of instances of audio that are *not* gunshots. Yet another kind of ambiguity occur for semantic reasons – is a burst of gunshots a single "burst" event, or multiple "shot" events?

Given the ambiguity, we realize we cannot expect to work with the traditional "strong" label requirement. Instead we use the "training with weak labels" approach, originally introduced in our research group [19, 30], where each recording is considered to be a "bag" of instances, of arbitrary (and possibly even unknown) size. Any recording that includes the target class is viewed as a "positive" bag, where it is only known that at least one instance is positive. In maintaining the ambiguity, there is no further indication of which of the instances in the positive bag is positive, or even the *number* of positive instances in the bag. A recording that does not include the target class is a

"negative" bag, in which no instance is positive. The model must be learned from such "weak" labels.

Weak labelling also brings about another ambiguity: we must also consider that *multiple* sounds may occur in an audio recording. So we have a "multi-label" scenario where a single recording may be weakly positive for multiple sound classes, even though they are not coincident.

In order to classify a recording in this setting we use the following logic [19]: if a bag of instances is positive, then *at least* one of the instances in the bag must be positive. In order to apply this logic to audio recordings, they must be converted to bags, and the above logic must be appropriately formalized.

Consequently a recording $x(1:T)$ (of length T) is segmented into a collection of segments: $x_i((i-1)\Delta t : i\Delta t), \quad i = 1:N$, where Δt is the length of the segments and $N = T/\Delta t$. In practice, we set $\Delta t = 1$ *sec*. A separate feature vector – also called an "embedding" is derived for each of the N segments: $E[i] = f(x_i)$, where $f()$ represents the feature extraction network. The set of embeddings $\{E_i,\ i = 1:N\}$ now forms our bag.

For a "vocabulary" of L, potentially co-occurring classes, we build L classifiers, one per class. In our setup each classifier is a simple logistic classifier for reasons we explain later. Thus, given any embedding E the classifier for the k^{th} class computes the posterior probability of the class as $P(k|E) \approx \sigma(W_k E + b_k)$, where W_k is a vector of weights for the class, b_k is a bias term that models its prior probability, and $\sigma()$ is the logistic function.

Given a recording that is labelled as positive for class $\hat{k}$, we now know that *at least* one of the segments in the recording must be positive, i.e. $P(\hat{k}|E_i) \approx 1$ for at least one of the segment embeddings E_i. Alternately stated, the probability $\max_i P(\hat{k}|E_i) \approx 1$. Note that $\hat{P}(\hat{k}|X) = \max_i P(\hat{k}|E_i)$ is actually a lower bound on $P(\hat{k}|X)$, the true probability that the recording includes at least one instance of $\hat{k}$. We can now set up both training and classification.

For training, consider a recording $X = \{E_i\}$ (i.e. which has been converted to a bag of embeddings), which has been labelled positive for classes $k_1, k_2, k_3, \cdots$, and negative for classes $l_1, l_2, l_3, \cdots$. Training the model comprises estimating the parameters W_k such that the cross entropy $Xent(\hat{P}(k_i|X); 1)$ is minimized for all the classes k_i. In principle we must also minimize $Xent(\hat{P}(l_i|X); 0)$ for the classes that are not present in the data.

Thus training takes the form:

$$\hat{W}_1, \hat{W}_2, \cdots = \arg\max_{W_1, W_2, \cdots} \sum_X \left(\sum_{k \in \{k_1, k_2, \cdots\}} Xent(\hat{P}(l_i|X); 1) + \sum_{l \in \{l_1, l_2, \cdots\}} Xent(\hat{P}(l_i|X); 0) \right)$$

(we have not explicitly mentioned the bias terms b_i in the above equations, but they too are estimated along with the weights W_k).

In practice, one may ignore or deweight the second term to indicate uncertainty of the *absence* of untagged classes.

Data insufficiency and adaptation

When training a classifier for a forensic setting with specific classes, often for which we do not have sufficient data samples, we must make some additional modifications to the above approach. Instead of training models entirely on our small training set, we must instead train the classifier on a larger wider dataset such as AudioSet, and *adapt* it to our setting.

Also, note that in the previous section, classification was expected to be performed on embeddings E that were obtained by the state-of-art model trained on large amounts of data. The data that this network is trained on may not be representative of the data obtained within the forensic application. Hence we must also account for this mismatch.

Thus, we use the following approach. Training follows a two-step procedure. In the first step, we compose a classifier that uses the embeddings from the AudioSet model directly as follows.

1. We first train a multi-label classifier for all the classes within a large dataset. In our case we used AudioSet, with its 2 million recordings covering 537 classes. This results in 537 parameters $W_1, \cdots, W_{537}$ (and the corresponding biases which are not explicitly shown.

2. For each of the target classes $c_1, \cdots, c_L$ that are explicitly found among the 537 classes, we set the weight vector for the class to be the corresponding audioset weight vector, to obtain $W_{c_1}, W_{c_2}, \cdots$. These represent the classifiers for these classes.

3. For classes $\hat{c}_1, \hat{c}_2, \cdots, \hat{c}_L$ that are *not* represented in the Audioset classes, we create L new weight vectors $W_{\hat{c}_1}, W_{\hat{c}_2}, \cdots, W_{\hat{c}_L}$ which are initialized with the weights of closest classes to $\hat{c}_1, \hat{c}_2, \cdots, \hat{c}_L$ among the audioset classes.

4. All the weights are then *fine tuned* using the smaller training data available for the task.

In the second step, we now *adapt* the embeddings to the new domain. We assume that the domain mismatch between the universal (AudioSet) Embedding E and the optimal Embeddings $\hat{E}$ for the forensic data can be modelled *piecewise* by an Affine

transform. Specifically, we assume that the Embedding space can be partitioned into a number of convex voronoi regions. Within each region the domain mismatch can be modelled by an Affine transform.

Specifically, for any embedding that falls within the j^{th} Voronoi region we assume that the relation between a "universal" embedding E and its corresponding domain-mismatched forensic-data embedding $\hat{E}$ can be given by

$$\hat{E} = A_j E + B_j$$

where A_j and B_j are region-specific affine parameters. To deal with data insufficiency we may assume a diagonal or tridiagonal matrix, rather than a full matrix. In order to estimate A_j and B_j for each Voronoi region, we now use the following procedure:

1. We identify the set $\mathcal{X}_j$ of all training instances that fall into the j^{th} Voronoi region.

2. We estimate the affine parameters for the region through the following optimization:

$$\hat{A}_j, \hat{B}_j = \arg\max_{A_j, B_j} \sum_{X \in \mathcal{X}_j} \left(\sum_{k \in \{k_1, k_2, \cdots\}} Xent(\max_i P(\hat{k}|\hat{E}_i); 1) + \sum_{l \in \{l_1, l_2, \cdots\}} Xent(\max_i P(\hat{k}|\hat{E}_i); 0) \right) \tag{A.3}$$

B

Learning without labels

Acoustic scene analysis is fast becoming a standard technology, expected in devices such as smartphones. However, the latest solutions are limited by the availability of labeled training data.

B.1 Significance of the work

In this work, we propose to automatically label a large quantity of audio data to generate the largest dataset for the research community. This will, however, require the development of algorithms that can iterate over such large amounts of data and iteratively refine their automatically generated labels.

On traditional machine learning hardware such as Graphical Processing Units (GPUs), we expect our approach to take several weeks or more of compute time for a single pass through the dataset, leading to unreasonable latencies in research (and development) time. We believe that the neocortex system can reduce the iteration time by orders of magnitude and enable us to optimize our unsupervised inference algorithms and put out labeled data resources that will be of high value to us and the research community at large.

There is increasing expectation of the sensing and AI capabilities of modern personal devices. Among the many abilities we expect of them, if not immediately, but in the near future, is to be able to "listen" to their acoustic environment and make interpretations of

them. Indeed, several such capabilities are already present in your smartphone – some apps can record bird sounds and inform us what bird it is; you can record a snippet of a catchy song at the mall and find the actual song; other applications detect gunshots, recognize the background noise, sounds like police and ambulance sirens, and can even geolocate on the basis of sounds heard.

For complete "acoustic" capabilities, however, the "vocabulary" of sounds and sound scenes a system must be able to recognize is very large. Pattern recognition models, generally neural networks, must be trained to recognize these sounds, and that requires labelled training data, which is still a very scarce commodity. Labeling data requires copious manual labor. Although many attempts have been made at crowdsourcing the labeling task, it remains labor intensive, and the largest corpora still only include a limited vocabulary of sound classes. Additionally, current state-of-the-art approaches for acoustic scene understanding rely on the availability of *strongly labeled data*, i.e., where the labels provide not only information about *which* sound classes are present in a recording but also the timestamps between which they occur. Strong labels are subjective to human bias, and the timestamp annotations are error-prone.

Our project aims to build up a vast corpus of automatically labeled audio. Members of our research group have previously gathered (in collaboration with researchers from various institutions), an enormous corpus of nearly a million audio recordings, comprising many thousands of hours of audio, amounting to several terabytes [1]. The objective is to automatically annotate each of these recordings with all the sound types present in them at various temporal granularities.

B.2 Overall Approach

Our approach will consist of employing state-of-the-art sound classifiers to perform preliminary classification of the recordings at various temporal granularities, followed by iterative refinement of the labels through a combination of the following procedures:

- Verifying the consistency of the labels across acoustically similar segments of the corpus. This will require both the comparison of the features and classification outputs from each (of hundreds of millions of) segments of audio to those derived from hundreds of millions of other similar segments, to establish consistency and filter high-confidence labels.

- Imposing automatically derived (and refined) sequentiality constraints. Once again, these will be derived and refined through analysis of label tags of tens or hundreds of millions of audio recordings.

- Automatically verifying the accuracy of the automatically assigned labels by training models with them and testing their performance on curated datasets with known labels.

- Imposing multi-view consensus among diverse models trained from these automatically derived labels. This is achieved by training several independent models in parallel on these data, using as training objective, a combination of the conformance of the output of the model to automatically derived labels and mutual conformance between the models themselves.

It is expected that a single experiment on the entire dataset, using conventional AWS-based solutions (even using their highest-end hardware) would require several weeks of computing times. Our proposed project requires iterated updates and refinements. Furthermore, the algorithm is iterative, and it is expected that continued iterations will improve the accuracy of the extracted labels continually. This will not be feasible on conventional computing platforms such as GPUs as they are not optimized for deep learning applications.

B.3 Overall Objective

Our objective is to release the outcome of our work, including the automatically derived labels, along with all code, to researchers around the world, as a public resource. When released, this will be the single largest tagged corpus of "data in the wild" made available to the public. (Please note that the data are legally cleared to be released to the public, and the unlabelled raw data are already in the public domain).

It is expected that the data will spur research on technologies for acoustic scene analysis, on the use of automatically labeled data, and on the refinement of automatically derived labels when large amounts of such data are available.

B.4 Approach

Data Processing: Data-preprocessing and feature extraction for the large-scale dataset and feature extraction is complete. A pipeline is built in order to use AudioSet dataset in order to analyze the AudioSet dataset using the PASST model and AST model. Features for the AudioSet dataset are also extracted at various levels of granularity in the temporal domain.

B.5 Notation and Setup

- $X_l = \{x_1^{(l)}, x_2^{(l)}, \ldots, x_{N_l}^{(l)}\}$: A small labeled dataset with ground-truth labels $L = \{L_1, L_2, \ldots, L_{N_l}\}$.
- $X_u = \{x_1, x_2, \ldots, x_{N_u}\}$: A large unlabeled dataset (e.g., YFCC).
- $f_\theta(x)$: A model (e.g., a neural network) parameterized by θ, which outputs class probabilities or embeddings.
- $\hat{L}^{(t)} = \{\hat{L}_1^{(t)}, \hat{L}_2^{(t)}, \ldots, \hat{L}_{N_u}^{(t)}\}$: The set of pseudo-labels for the unlabeled data at iteration t.
- $L_{sup}(L, f_\theta(X_l))$: A supervised loss computed on the small labeled set (e.g., cross-entropy).
- $L_{unsup}(\hat{L}^{(t)}, f_\theta(X_u))$: An unsupervised loss that encourages the model predictions on X_u to match the current pseudo-labels $\hat{L}^{(t)}$.

We operate at multiple temporal granularities, denoted $h_1, h_2, \ldots, h_K$. For instance, we may obtain labels at a fine resolution of 0.5 s with 0.1 s overlap, and then aggregate these to coarser temporal scales (e.g., 1 s, 2 s).

B.6 Hierarchical and Multi-Model Constraints

B.6.1 Hierarchical Consistency

Define $H_k(x)$ as a function that aggregates finer-grained predictions into coarser segments at the temporal scale h_k. We impose a hierarchical consistency loss L_{hier} that ensures the model's predictions are consistent across different temporal granularities:

$$L_{hier}(\hat{L}^{(t)}, f_\theta(X_u)) = \sum_{k=1}^{K} \sum_{x \in X_u} \|H_k(f_\theta(x)) - f_\theta(x)\|^2. \tag{B.1}$$

B.6.2 Multi-View Consistency

To increase robustness, we train M models in parallel: $f_{\theta_1}, f_{\theta_2}, \ldots, f_{\theta_M}$. A multi-view consistency loss L_{mv} ensures these models produce similar predictions:

$$L_{mv} = \sum_{m=1}^{M} \sum_{m'=m+1}^{M} \sum_{x \in X_u} D\big(f_{\theta_m}(x), f_{\theta_{m'}}(x)\big), \tag{B.2}$$

where $D(\cdot, \cdot)$ measures divergence (e.g., KL divergence) between model outputs.

B.7 Overall Objective

At iteration t, we combine these objectives. Using the align environment helps break the equation across multiple lines:

$$\begin{aligned}\theta_{t+1} = \arg\min_{\theta} \big[& \lambda_{sup} L_{sup}(L, f_\theta(X_l)) \\ & + \lambda_{unsup} L_{unsup}(\hat{L}^{(t)}, f_\theta(X_u)) \\ & + \lambda_{hier} L_{hier}(\hat{L}^{(t)}, f_\theta(X_u)) \\ & + \lambda_{mv} L_{mv} \big]. \end{aligned} \tag{B.3}$$

Here, λ_{sup}, λ_{unsup}, λ_{hier}, and λ_{mv} are hyperparameters that balance the influence of each loss component.

B.8 Iterative Refinement Procedure

B.8.1 Initialization

Start with a baseline model f_{θ_0}, which may be pretrained or trained on a small labeled set (e.g., the balanced AudioSet subset). Generate initial pseudo-labels:

$$\hat{L}^{(0)} = f_{\theta_0}(X_u). \tag{B.4}$$

B.8.2 Refinement Steps

At each iteration t:

1. **Model Re-Training:**

$$\begin{aligned}\theta_{t+1} = \arg\min_{\theta} \big[& \lambda_{sup} L_{sup}(L, f_\theta(X_l)) \\ & + \lambda_{unsup} L_{unsup}(\hat{L}^{(t)}, f_\theta(X_u)) \\ & + \lambda_{hier} L_{hier}(\hat{L}^{(t)}, f_\theta(X_u)) \\ & + \lambda_{mv} L_{mv} \big]. \end{aligned} \tag{B.5}$$

2. **Pseudo-Label Update:**

$$\hat{L}^{(t+1)} = f_{\theta_{t+1}}(X_u). \tag{B.6}$$

B.8.3 Stopping Criterion

Evaluate f_{θ_t} on a holdout AudioSet test set. Let $mAP^{AS}(\theta_t)$ be the mean Average Precision on this test set, and let mAP^{AS}_{base} be the performance of a model trained solely on the balanced AudioSet subset.

We stop when:

$$mAP^{AS}(\theta_t) > mAP^{AS}_{base}. \tag{B.7}$$

If this condition is not met, we continue with another iteration.

B.8.4 Intuition and Outcome

This iterative procedure gradually improves the quality of pseudo-labels by enforcing consistency across time, multiple models, and leveraging hierarchical constraints. Over time, the unsupervised labels become more accurate, leading to models that surpass the baseline performance achieved by using only the balanced AudioSet subset. This large-scale labeling approach will enable the creation of massive, publicly available labeled audio corpora, spurring further advances in acoustic scene analysis.

B.9 Results

We built a pipeline by performing analysis on the AudioSet test set as the evaluation set. First, we will measure the performance of the model trained on the pseudo-labels for the YFCC dataset with the YFCC dataset as training and obtain a baseline evaluation performance. The labels will then be refined using the label consistency measure developed by obtaining labels at different levels of temporal granularity. The labels will also be refined based on the information obtained from multiple different types of models and the consistency of the predicted labels. Our current results can improve the performance of the 0.38 mAP score using the above approach.

C

Strengthening the labels: Semi-weak label learning

C.1 Strengthening of Labels

Weak label represents the presence/absence of an event in the data when dealing with weak labels for audio classification. Since weak labels are relatively easier to obtain than strong labels for audio understanding has aided the rise in popularity of approaches making use of weak labels. In weak label learning, the labels provided for the data used for training are not completely accurate or reliable. This can occur for a variety of reasons, such as incomplete or ambiguous labeling, or because the labels are generated automatically based on certain criteria. Despite the relative ease of collecting weak labels at a large scale, there still exists a significant performance gap between weak-label learning approaches and strongly supervised learning methods. To address this gap, we came up with semi-weak label learning as seen in Chapter 6. This approach uses the count of the events as additional information over the weakly labeled annotations, thereby making it possible to bridge the performance gap.

C.1.1 Speech Recognition as a weak label learning problem

Additional cues, however, still are unexplored in the community. For example, Speech recognition is a field of artificial intelligence that deals with the ability of a machine to understand and transcribe speech. For speech recognition, we are using the sequence of phonemes, also known as the phonetic transcription, involving the transcription of the spoken language into a sequence of phonemes, which are the smallest unit of sound that can distinguish one word from another in a particular language. Here, in this case, we know the weak labels are the phonemes that are present in the data but we also know the sequence of weak labels in the data. Such information is a key driver to building systems at a more granular level rather than using the sequence of words or characters.

There are several approaches that can be used to perform speech recognition using the sequence of phonemes. One approach is to use a phonetic dictionary, which maps words to their corresponding phonetic transcription. The speech recognition system can then use this dictionary to transcribe the spoken language into a sequence of phonemes. Another approach is to use a phonetic transcription model, which is trained to transcribe spoken language into a sequence of phonemes based on a dataset of labeled speech data. The model can be trained using supervised learning techniques, where the input is a spoken utterance and the output is the corresponding sequence of phonemes. Once the model is trained, it can be used to transcribe new spoken utterances into their corresponding phonetic transcriptions. Overall, speech recognition using the sequence of phonemes can be a useful approach for improving the accuracy of speech recognition systems, particularly for languages with a large number of homophones (words that are pronounced the same but have different meanings) or for tasks that require a high level of accuracy in transcribing spoken language.

C.2 Use of additional cues - Label Strengthening

The additional cues can provide information by strengthening the information provided by weak labels. For example, it is more informative to know that this picture contains ‘a car‘ and ‘a passenger was driving it‘ as opposed to only knowing that the picture contains ‘car‘ in it. Here, in the first scenario, it is more probable that the situation contains ‘a car with a person driving it‘ over ‘a toy car‘, thereby revealing contextual information about the environment. Such types of additional cues are relatively less exploited in the

Additional Cues can be used to strengthen the labels in the following ways.

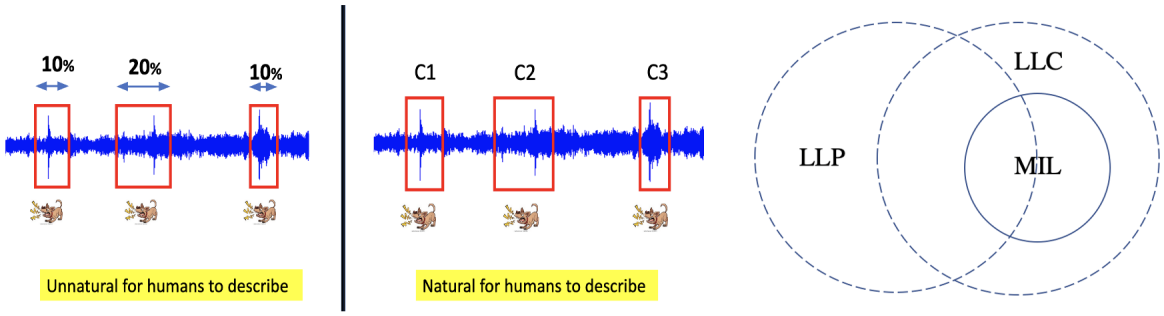


Figure C.1: a) Figure depicting whether it is natural for humans to describe with proportions or counts and b) Figure providing the comparison between learning with proportions and learning with counts.

C.2.1 Counts: Semi-weak Label learning

Additional cues along with the weakly labeled data can be used to strengthen the labels is in the form of counts information, proportion information, approximate counts, bounds on the counts and utilizing the combination of the strongly and weakly labeled data.

C.2.2 Learning using Proportions

Learning from Label Proportions (LLP) is a learning setting, where the training data is provided in groups, or "bags", and only the proportion of each class in each bag is known. The task is to learn a model to predict the class labels of the individual instances. In this case, the learning is enabled using the label proportions of the instances present in the bag. Learning with proportions may not be applicable in practical settings for case of audio data for instance if one considers audio recording as a bag of individual events since it is not natural to describe/label the data in the form of proportions. For instance, say a recording has gunshot, dog barking and man shouting, then it is much easier to label that there are two gunshots in the data than to mention the recording has 10

C.2.3 Comparison between different types of count of events in recording

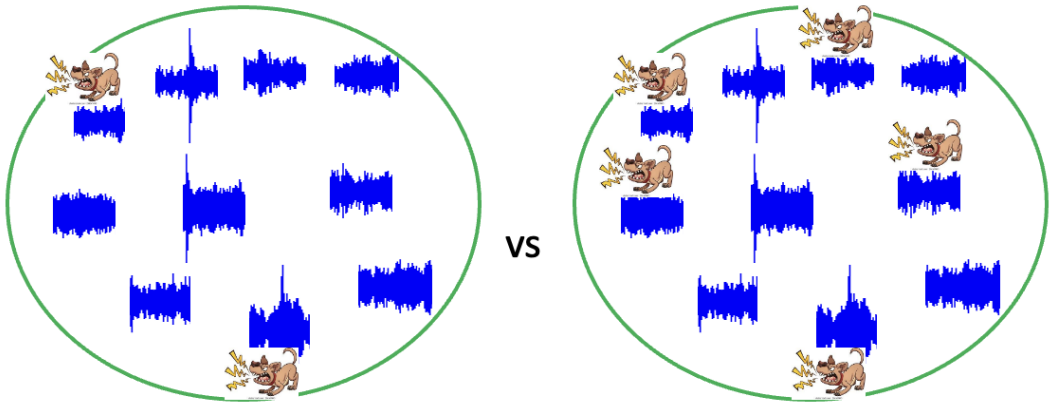


Figure C.2: Comparison between the different types of counts of events in the recording

Figure C.3: Figure depicting the following a) Approximate counts, b) Upper and Lower bounds on counts and c) Comparative counts

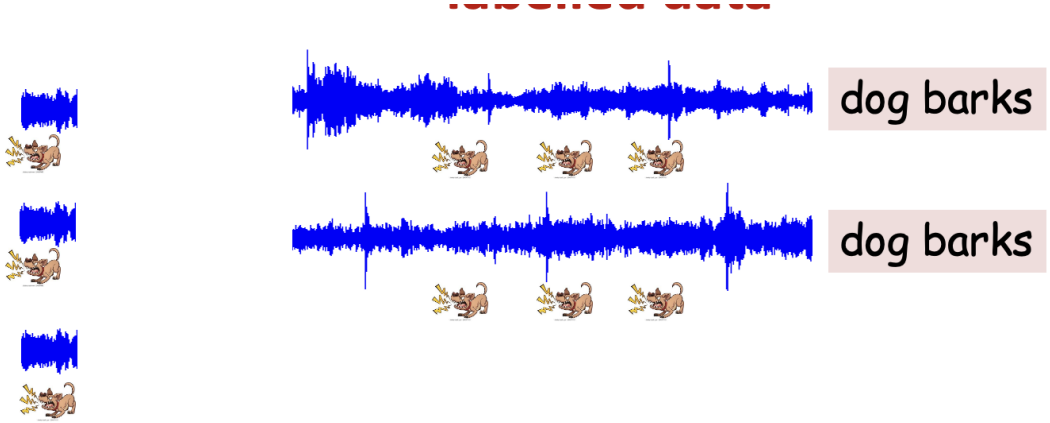


Figure C.4: Combining strongly and weakly labeled data

D

Appendix: Unified Formalism

D.1 Notation

We present the notation table for each symbol used in this appendix section in Table D.1.

D.2 Related Work

Many previous methods have been proposed for dealing with the specific types and some combinations of imprecise label configurations. We revisit the relevant work in this section, especially the state-of-the-art popular baselines for learning with individual and mixture imprecise label configurations.

Partial label learning (PLL). The prior arts can be roughly divided into identification-based for label disambiguation [268, 269, 270, 271] or average-based for utilizing all candidate labels [238, 180, 225]. The traditional average-based methods usually treat all candidate labels equally, which may involve the misleading false positive labels into training. To overcome these limitations, researchers have explored identification-based methods, viewing the ground-truth label as a latent variable. They seek to maximize its estimated probability using either the maximum margin criterion [272, 273] or the maximum likelihood criterion [274]. Deep learning techniques have recently been incorporated into identification-based methods, yielding promising results across multiple datasets. For example, PRODEN [222] proposed a self-training strategy

Table D.1: Notation Table

Notation	Definition
$\mathbf{x}$	A training instance
y	A class index label
$\{\mathbf{x}_i\}_{i\in[N]}$	A set of data instances $\mathbf{x}$ of size N
$\{y_i\}_{i\in[N]}$	A set of precise label indices y of size N
$[\iota]$	An imprecise label, which might contain multiple class indices
$\{[\iota]_i\}_{i\in[N]}$	A set of imprecise labels $[\iota]$ of size N
X	Random variable of training instance
$\mathcal{X}$	Input space where $\mathbf{x}$ is drawn from
Y	Random variable of ground-truth labels
$\mathcal{Y}$	Label space where y is drawn from
I	Random variable of imprecise labels
f	Model backbone
g	Model classifier
h	Model multi-layer perceptron
$f\circ g$	Model mapping $\mathcal{X}\to\mathcal{Y}$
θ	Learnable parameters of $f\circ g$
$\mathbf{p}(y\|\mathbf{x};\theta)$	Output probability from model $f\circ g$
$f\circ h$	Model mapping $\mathcal{X}\to\mathcal{Z}$, where Z is a projected feature space
$\mathcal{D}$	Dataset
$\mathcal{L}$	Loss function
$\mathcal{A}_\mathrm{w}$	Weak data augmentation, usually is HorizontalFlip
$\mathcal{A}_\mathrm{s}$	Strong data augmentation, usually is RandAugment [267]
$\mathbf{z}_\mathrm{w}$	Projected features from $f\circ h$ on weakly-augmented data
$\mathbf{z}_\mathrm{s}$	Projected features from $f\circ h$ on strongly-augmented data
$\mathcal{M}$	Memory queue in MoCo [236]
$\mathbf{s}$	A partial label, with ground-truth label contained
$\{\mathbf{s}_i\}_{i\in[N]}$	A set partial labels, with ground-truth label contained of size N
S	Random variable of partial label
$\mathbf{x}^\mathrm{l}$	A labeled training example
y^l	A labeled class index
$\mathbf{x}^\mathrm{u}$	A unlabeled training example
y^u	A unknown class index for unlabeled data
X^L	A set of labeled data instances
Y^L	A set of labels for labeled data instances
X^U	A set of unlabeled data instances
Y^U	A set of unknown labels for unlabeled data instances
$\hat{p}^\mathbf{u}$	The maximum predicted probability on unlabeled data $\max(\mathbf{p}(y\|\mathbf{x}^\mathrm{u};\theta))$
$\hat{y}^\mathrm{u}$	The pseudo-label from the predicted probability on unlabeled data $\arg\max(\mathbf{p}(y\|\mathbf{x}^\mathrm{u};\theta))$
τ	The threshold for confidence thresholding
$\hat{y}$	A corrupted/noisy label
$\hat{y}^\mathrm{oh}$	An one-hot version of the corrupted/noisy label
$\hat{Y}$	Random variable of noisy labels
$\mathbf{u},\mathbf{v},\mathbf{m}$	Noise model related parameters in SOP [4]
$\mathcal{T}(\hat{y}\|y;\omega)$	The simplified noise transition model in ILL
ω	The parameters in the simplified noise model

that disambiguates candidate labels using model outputs. CC [182] introduced classifier-consistent and risk-consistent algorithms, assuming uniform candidate label generation. LWS [184] relaxed this assumption and proposed a family of loss functions for label disambiguation. More recently, Wangt et al. [2] incorporated contrastive learning into PLL, enabling the model to learn discriminative representations and show promising results under various levels of ambiguity. RCR involves consistency regularization into PLL recently [185].

Semi-supervised learning (SSL). SSL is a paradigm for learning with a limited labeled dataset supplemented by a much larger unlabeled dataset. Consistency regularization and self-training, inspired by clusterness and smoothness assumptions, have been proposed to encourage the network to generate similar predictions for inputs under varying perturbations [275, 187, 276]. Self-training [186, 239, 3] is a widely-used approach for leveraging unlabeled data. Pseudo Label [186, 239], a well-known self-training technique, iteratively creates pseudo labels that are then used within the same model. Recent studies focus largely on generating high-quality pseudo-labels. MixMatch [188], for instance, generates pseudo labels by averaging predictions from multiple augmentations. Other methods like ReMixMatch [189], UDA [221], and FixMatch [3] adopt confidence thresholds to generate pseudo labels for weakly augmented samples, which are then used to annotate strongly augmented samples. Methods such as Dash [277], FlexMatch [191], and FreeMatch [193] dynamically adjust these thresholds following a curriculum learning approach. SoftMatch [194] introduces a novel utilization of pseudo-labels through Gaussian re-weighting. SSL has also seen improvements through the incorporation of label propagation, contrastive loss, and meta learning [278, 279, 251, 252, 5].

Noisy label learning (NLL). Overfitting to the noisy labels could result in poor generalization performance, even if the training error is optimized towards zero [114, 280]. Several strategies to address the noisy labels have been proposed [281]. Designing loss functions that are robust to noise is a well-explored strategy for tackling the label noise problem [199, 201, 282, 283]. Additionally, methods that re-weight loss [284] have also been explored for learning with noisy labels. Another common strategy to handle label noise involves assuming that the noisy label originates from a probability distribution that depends on the actual label. Early works [112] incorporated these transition probabilities into a noise adaptation layer that is stacked over a classification network and trained in an end-to-end fashion. More recent work, such as Forward [256], prefers to estimate these transition probabilities using separate procedures. However, the success of this method is contingent upon the availability of clean validation data [285] or additional assumptions about the data [286]. Noise correction has shown promising results in noisy label learning recently [287, 288, 289, 4]. During the early learning phase, the model can accurately predict a subset of the mislabeled examples [202]. This observation suggests a potential strategy of correcting the corresponding labels. This could be accomplished by generating new labels equivalent to soft or hard pseudo-labels estimated by the model [111, 290]. Co-Teaching uses multiple differently trained networks for correcting noisy labels [198]. SELFIE [291] corrects a subset of labels by replacing them based on past model outputs. Another study in [292] uses a two-component mixture model for sample selection, and then corrects labels using

a convex combination. Similarly, DivideMix [203] employs two networks for sample selection using a mixture model and Mixup [255].

Mixture imprecise label settings. Various previous works have explored dealing with distinct types of imprecise labels. However, they have yet to tackle a combination of partial labels, limited labels, and noisy labels, which is a highly realistic scenario. For instance, recent attention has been paid to the issue of partial noisy label learning. PiCO+ [259], an extended version of PiCO [2], is tailored specifically for partial noisy labels. IRNet [237] uses two modules: noisy sample detection and label correction, transforming the scenario of noisy PLL into a more traditional PLL. DALI [218] is another framework designed to reduce the negative impact of detection errors by creating a balance between the initial candidate set and model outputs, with theoretical assurances of its effectiveness. Additionally, some work has focused on semi-supervised partial label learning [219, 220]. No existing research can effectively address the challenge of handling a combination of partial, limited, and noisy labels simultaneously, which underscores the novelty and significance of our work.

Previous attempts towards unification of learning from imprecise labels. There are earlier attempts for the generalized solutions of different kinds of imprecise labels/observations. Denœux [227] proposed an EM algorithm for the likelihood estimation of fuzzy data and verified the algorithm on linear regression and uni-variate normal mixture estimation. Van Rooyen et al. [230] developed an abstract framework that generically tackles label corruption via the Markov transition. Quost et al. [229] further extended the EM algorithm of fuzzy data on the finite mixture of Gaussians. Gong et al. [231] proposed a general framework with centroid estimation for imprecise supervision. A unified partial AUC optimization approach was also proposed earlier [234]. Zhang et al. [213] and Wei. et al. [233] proposed generalized solutions for aggregate observations. A unified solution based on dynamic programming for count-based weak supervision was also proposed [293] While relating to these works on the surface, ILL does not require any assumption on the imprecise information and generalizes well to more practical settings with noisy labels. Some other works for individual settings also related EM framework, but usually involved the approximation on the EM [294, 196, 2].

D.3 Methods

D.3.1 Derivation of Variational Lower Bound

Evidence lower bound (ELBO), or equivalently variational lower bound [226], is the core quantity in EM. From Equation (9.27), to model $\log P(X, I; \theta)$, we have:

$$\begin{aligned}
\log P(X, I; \theta) &= \int Q(Y) \log P(X, I; \theta) dY \\
&= \int Q(Y) \log P(X, I; \theta) \frac{P(Y|X, I; \theta)}{P(Y|X, I; \theta)} dY \\
&= \int Q(Y) \log \frac{P(X, I, Y; \theta) Q(Y)}{P(Y|X, I; \theta) Q(Y)} dY \\
&= \int Q(Y) \log \frac{P(X, I, Y; \theta)}{Q(Y)} dY - \int Q(Y) \log \frac{P(Y|X, I; \theta)}{Q(Y)} dY
\end{aligned} \tag{D.1}$$

where the first term is the ELBO and the second term is the KL divergence $\mathcal{D}_{KL}(Q(Y)||P(Y|X, I; \theta))$. Replacing $Q(Y)$ with $P(Y|X, I; \theta^t)$ at each iteration will obtain Equation (9.27).

D.3.2 Instantiations to Partial Label Learning

The imprecise label I for partial labels is defined as the label candidate sets S with $\{\mathbf{s}_i\}_{i \in [N]}$ containing the true labels. Now we can derive Equation (9.28) by replacing I with S in Equation (9.27):

$$\begin{aligned}
&\mathbb{E}_{Y|X,I;\theta^t} \left[\log P(Y|X; \theta) + \log P(I|X, Y; \theta)\right] \\
&= \mathbb{E}_{Y|X,S;\theta^t} \left[\log P(Y|X; \theta) + \log P(I|X, Y; \theta)\right] \\
&= \sum_Y P(Y|X, S; \theta^t) \left[\log P(Y|X; \theta) + \log P(I|X, Y; \theta)\right] \\
&= \sum_Y P(Y|X, S; \theta^t) \left[\log P(Y|X; \theta)\right] + \log P(I|X, Y; \theta)
\end{aligned} \tag{D.2}$$

Note that $P(I|Y, X; \theta)$ can be moved out of the expectation because it is a fixed quantity to any Y. Now we replace Y, X, and S to y, $\mathbf{x}$, and $\mathbf{s}$ for each instance, and converting the maximization problem to negative log-likelihood minimization problem to drive the loss function:

$$\mathcal{L}_{\text{ILL}}^{\text{PLL}} = -\frac{1}{N} \sum_i^N \mathbf{p}(y_i|\mathbf{x}_i, \mathbf{s}_i; \theta^t) \log \mathbf{p}(y_i|\mathbf{x}_i; \theta) - \frac{1}{N} \sum_i^N \log \mathbf{p}(\mathbf{s}_i|\mathbf{x}_i, y_i; \theta). \tag{D.3}$$

The first term is the Cross-Entropy loss we derived in Equation (9.28). If S is not instance-dependent, then knowing Y also knows S, the second term thus can be ignored in Equation (9.28). If S becomes instance-dependent, the second term can be maintained as a supervised term as in [185] to optimize θ.

D.3.3 Instantiations to Semi-Supervised Learning

In SSL, the input X consists of the labeled data X^{L} and the unlabeled data X^{U}. The imprecise label for SSL is realized as the limited number of full labels Y^{L} for X^{L}. The labels Y^{U} for unlabeled X^{U} are unknown and become the latent variable. Thus we can write:

$$\begin{aligned}
&\mathbb{E}_{Y|X,I;\theta^t}\left[\log P(Y|X;\theta) + \log P(I|X,Y;\theta)\right] \\
&= \mathbb{E}_{Y^{\mathrm{U}}|X^{\mathrm{U}},X^{\mathrm{L}},Y^{\mathrm{L}};\theta^t}\left[\log P(Y^{\mathrm{U}}|X^{\mathrm{U}},X^{\mathrm{L}};\theta) + \log P(Y^{\mathrm{L}}|X^{\mathrm{L}},X^{\mathrm{U}},Y^{\mathrm{U}};\theta)\right] \\
&= \sum_{Y^{\mathrm{U}}} P(Y^{\mathrm{U}}|X^{\mathrm{U}};\theta^t)\left[\log P(Y^{\mathrm{U}}|X^{\mathrm{U}};\theta)\right] + \log P(Y^{\mathrm{L}}|X^{\mathrm{L}};\theta).
\end{aligned} \tag{D.4}$$

The negative log-likelihood loss function for $\{\mathbf{x}_i^l, y_i^l\}_{i\in[N^L]}$ and $\{\mathbf{x}^u\}_{i\in[N^U]}$ thus becomes:

$$\mathcal{L}_{\mathrm{ILL}}^{\mathrm{SSL}} = \mathcal{L}_{\mathrm{CE}}\left(\mathbf{p}(y|\mathbf{x}^{\mathrm{u}};\theta), \mathbf{p}(y|\mathbf{x}^{\mathrm{u}};\theta^t)\right) + \mathcal{L}_{\mathrm{CE}}\left(\mathbf{p}(y|\mathbf{x}^{\mathrm{L}};\theta), y^{\mathrm{L}}\right) \tag{D.5}$$

D.3.4 Instantiations to Noisy Label Learning

We denote the given noisy labels as $\hat{Y}$. For noisy label learning, our method naturally supports a noise transition model $\mathcal{T}(\hat{Y}|Y;\omega)$ with learnable parameter ω, as we will show in the following:

$$\begin{aligned}
&\mathbb{E}_{Y|X,I;\theta^t}\left[\log P(Y|X;\theta) + \log P(I|X,Y;\theta)\right] \\
&= \mathbb{E}_{Y|X,\hat{Y};\theta^t}\left[\log P(Y,\hat{Y}|X;\theta)\right] \\
&= \mathbb{E}_{Y|X,\hat{Y};\theta^t}\left[\log P(Y|\hat{Y},X;\theta) + \log P(\hat{Y}|X;\theta)\right] \\
&= \sum_{Y} P(Y|\hat{Y},X;\theta^t)\log P(Y|\hat{Y},X;\theta) + \log P(\hat{Y}|X;\theta).
\end{aligned} \tag{D.6}$$

The loss function is:

$$\mathcal{L}_{\mathrm{ILL}}^{\mathrm{NLL}} = \mathcal{L}_{\mathrm{CE}}\left(\mathbf{p}(y|\mathbf{x},\hat{y};\theta,\omega^t), \mathbf{p}(y|\mathbf{x},\hat{y};\theta^t,\omega^t)\right) + \mathcal{L}_{\mathrm{CE}}\left(\mathbf{p}(\hat{y}|\mathbf{x};\theta,\omega), \hat{y}\right) \tag{D.7}$$

Note that both term is computed from the noise transition matrix as mentioned in Equation (9.31).

D.3.5 Instantiations to Mixed Imprecise Label Learning

In this setting, we have both labeled data and unlabeled data, where the labels for the labeled data are both partial and noisy. On the unlabeled data, the unsupervised objective is the same as the unsupervised consistency regularization of semi-supervised learning shown in Equation (9.29). On the labeled data, it mainly follows the Equation (9.31) of noisy label learning, with the noisy single label becoming the noisy partial labels $\hat{\mathbf{s}}$. For noisy partial labels, the noisy supervised objective in Eq. 8 becomes the supervised consistency regularization as in Eq. 6 of partial label setting to train the noise transition model, and the noisy unsupervised objective becomes the consistency

regularization of the prediction conditioned on noisy partial labels:

$$\mathcal{L}_{\mathrm{CE}}\left(\mathbf{p}\left(y \mid \mathcal{A}_{\mathrm{s}}(\mathbf{x}), \hat{\mathbf{s}}; \theta, \omega^{t}\right), \mathbf{p}\left(y \mid \mathcal{A}_{\mathrm{w}}(\mathbf{x}), \hat{y}; \theta^{t}, \omega^{t}\right)\right) + \mathcal{L}_{\mathrm{CE}}\left(\mathbf{p}\left(\hat{y} \mid \mathcal{A}_{\mathrm{w}}(\mathbf{x}); \theta, \omega\right), \hat{\mathbf{s}}\right) \quad \text{(D.8)}$$

We can compute both quantity through the noise transition model:

$$\mathbf{p}(y|\mathbf{x}, \hat{\mathbf{s}}; \theta, \omega^{t}) \propto \mathbf{p}(y|\mathbf{x}; \theta) \prod_{\hat{y} \in \hat{\mathbf{s}}} \mathcal{T}(y|\hat{y}; \omega^{t}), \text{and } \mathbf{p}(\hat{y}|\mathbf{x}; \theta, \omega) = \sum_{y \in [C]} \mathbf{p}(y|\mathbf{x}; \theta) \mathcal{T}(\hat{y}|y; \omega). \quad \text{(D.9)}$$

D.4 Experiments

D.4.1 Additional Training Details

We adopt two additional training strategies for the ILL framework. The first is the "strong-weak" augmentation strategy [221]. Since there is a consistency regularization term in each imprecise label formulation of ILL, we use the soft pseudo-targets of the weakly-augmented data to train the strongly-augmented data. The second is the entropy loss [241] for class balancing, which is also adopted in SOP [4] and FreeMatch [193]. We set the loss weight for the entropy loss uniformly for all experiments as 0.1.

D.4.2 Partial Label Learning

D.4.2.1 Setup

Following previous work [295, 184, 2], we evaluate our method on partial label learning setting using CIFAR-10, CIFAR-100, and CUB-200 [244]. We generate partially labeled datasets by flipping negative labels to false positive labels with a probability q, which is also denoted as a partial ratio. Specifically, the $C-1$ negative labels are uniformly aggregated into the ground truth label to form a set of label candidates. We consider $q \in \{0.1, 0.3, 0.5\}$ for CIFAR-10, $q \in \{0.01, 0.05, 0.1\}$ for CIFAR-100, and $q = 0.05$ for CUB-200. For CIFAR-10 and CIFAR-100, we use ResNet-18 [174] as backbone. We use SGD as an optimizer with a learning rate of 0.01, a momentum of 0.9, and a weight decay of 1e−3. For CUB-200, we initialize the ResNet-18 [174] with ImageNet-1K [296] pre-trained weights. We train 800 epochs for CIFAR-10 and CIFAR-100 [243], and 300 epochs for CUB-200, with a cosine learning rate scheduler. For CIFAR-10 and CIFAR-100, we use an input image size of 32. For CUB-200, we use an input image size of 224. A batch size of 256 is used for all datasets. The choice of these parameters mainly follows PiCO [2]. We present the full hyper-parameters systematically in Table D.2.

D.4.2.2 Discussion

We additionally compare our method with R-CR [185], which uses a different architecture as the results in Table 9.1. R-CR uses Wide-ResNet34x10 as backbone, and adopts

Table D.2: Hyper-parameters for **partial label learning** used in experiments.

Hyper-parameter	CIFAR-10	CIFAR-100	CUB-200
Image Size	32	32	224
Model	ResNet-18	ResNet-18	ResNet-18 (ImageNet-1K Pretrained)
Batch Size	256	256	256
Learning Rate	0.01	0.01	0.01
Weight Decay	1e-3	1e-3	1e-5
LR Scheduler	Cosine	Cosine	Cosine
Training Epochs	800	800	300
Classes	10	100	200

multiple strong data augmentations. It also adjusts the loss weight along training. For fair comparison, we use the same architecture without multiple augmentation and the curriculum adjust on loss. The results are shown in Table D.3, where our method outperforms R-CR on CIFAR-10 and is comparable on CIFAR-100.

Table D.3: Comparison with R-CR in partial label learning

Method	CIFAR-10		CIFAR-100	
	0.3	0.5	0.05	0.10
R-CR	$97.28_{\pm 0.02}$	$97.05_{\pm 0.05}$	$82.77_{\pm 0.10}$	$82.24_{\pm 0.07}$
Ours	$97.55_{\pm 0.07}$	$97.17_{\pm 0.11}$	$82.46_{\pm 0.08}$	$82.22_{\pm 0.05}$

We also provide the comparison of our method on instance-dependent partial label learning as proposed by Xu et al. [247, 295]. Due to the nature of instance-dependence, we maintain the term $P(S|Y, X; \theta)$ from Equation (9.27) as a supervised term for optimization. We compare our method with VALEN [247], RCR [185], PiCO [2], and POP [295] on MNIST, Kuzushiji-MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100, with synthetic instance-dependent partial labels generated according to Xu et al. [295]. From the results in Table D.4, we proposed method demonstrate the best performance across different datasets evaluated.

Table D.4: Comparison on instance-dependent partial label learning

	MNIST	Kuzushiji-MNIST	Fashion-MNIST	CIFAR-10	CIFAR-100
VALEN [247]	99.03	90.15	96.31	92.01	71.48
RCR [185]	98.81	90.62	96.64	86.11	71.07
PiCO [2]	98.76	88.87	94.83	89.35	66.30
POP [295]	**99.28**	91.09	96.93	93.00	71.82
Ours	99.19	**91.35**	**97.01**	**93.86**	**72.43**

A recent work on PLL discussed and analyzed the robustness performance of different loss functions, especially the average-based methods [225]. We perform a similar analysis here for the derived loss function in ILL. Following the notation in [225], let $\mathbf{s}$ denote the candidate label set, $\mathbf{x}$ as the training instance, g as the probability score from the

model, and f as the classifier $f(\boldsymbol{x}) = \arg\max_{i \in \mathcal{Y}} g_i(\boldsymbol{x})$, the average-based PLL can be formulated as:

$$\mathcal{L}_{avg-PLL}(f(\boldsymbol{x}), \mathbf{s}) = \frac{1}{|\mathbf{s}|} \sum_{i \in \mathbf{s}} \ell(f(\boldsymbol{x}), i) \tag{D.10}$$

Lv et al. [225] compared different loss functions ℓ on both noise-free and noisy PLL settings, where they find both theoretically and empirically that average-based PLL with *bounded* loss are robust under mild assumptions. Empirical study in [225] suggests that both *Mean Absolute Error* and *Generalized Cross-Entropy* loss [199] that proposed for noisy label learning achieves the best performance and robustness for average-based PLL.

Our solution for PLL can be viewed as an instantiation of the average-based PLL as in [225] with:

$$\ell(f(\mathbf{x}), i) = -\bar{g}_i(\mathbf{x}) \log g_i(\mathbf{x}) \tag{D.11}$$

where $\bar{g}$ is normalized probability over $\mathbf{s}$ with detached gradient. We can further show that the above loss function is bounded for $0 < \ell \leq \frac{1}{e}$ and thus bounded for summation of all classes, which demonstrates robustness, as we show in Table 9.4.

D.4.3 Semi-Supervised Learning

D.4.3.1 Setup

For experiments of SSL, we follow the training and evaluation protocols of USB [5] on image and text classification. To construct the labeled dataset for semi-supervised learning, we uniformly select l/C samples from each class and treat the remaining samples as the unlabeled dataset. For image classification tasks, ImageNet-1K [296] Vision Transformers [177] are used, including CIFAR-100 [243], EuroSAT [297], STL-10 [298], TissueMNIST [299, 300], Semi-Aves [301]. For text classification tasks, we adopt BERT [176] as backbone, including IMDB [248], Amazon Review [249], Yelp Review [302], AG News [303] , Yahoo Answer [304]. The hyper-parameters strictly follow USB, and are shown in Table D.5 and Table D.6.

D.4.3.2 Results

In the chapter, we only provide the comparison on CIFAR-100, STL-10, IMDB, and Amazon Review. Here we provide the full comparison in Table D.7 and Table D.8. From the full results, similar conclusion can be drawn as in the chapter. Our ILL framework demonstrates comparable performance as previous methods.

Table D.5: Hyper-parameters of **semi-supervised learning** used in vision experiments of USB.

Hyper-parameter	CIFAR-100	STL-10	Euro-SAT	TissueMNIST	Semi-Aves
Image Size	32	96	32	32	224
Model	ViT-S-P4-32	ViT-B-P16-96	ViT-S-P4-32	ViT-T-P4-32	ViT-S-P16-224
Labeled Batch size	16				
Unlabeled Batch size	16				
Learning Rate	5e-4	1e-4	5e-5	5e-5	1e-3
Weight Decay	5e-4				
Layer Decay Rate	0.5	0.95	1.0	0.95	0.65
LR Scheduler	$\eta = \eta_0 \cos(\frac{7\pi k}{16K})$				
Training epochs	20				
Classes	100	10	10	10	200
Model EMA Momentum	0.0				
Prediction EMA Momentum	0.999				
Weak Augmentation	Random Crop, Random Horizontal Flip				
Strong Augmentation	RandAugment [267]				

Table D.6: Hyper-parameters of **semi-supervised learning** NLP experiments in USB.

Hyper-parameter	AG News	Yahoo! Answer	IMDB	Amazom-5	Yelp-5
Max Length	512				
Model	Bert-Base				
Labeled Batch size	4				
Unlabeled Batch size	4				
Learning Rate	5e-5	1e-4	5e-5	1e-5	5e-5
Weight Decay	1e-4				
Layer Decay Rate	0.65	0.65	0.75	0.75	0.75
LR Scheduler	$\eta = \eta_0 \cos(\frac{7\pi k}{16K})$				
Training epochs	10				
Classes	4	10	2	5	5
Model EMA Momentum	0.0				
Prediction EMA Momentum	0.999				
Weak Augmentation	None				
Strong Augmentation	Back-Translation [221]				

Table D.7: Error rate comparison of different number of labels on CIFAR-100, STL-10, EuroSAT, TissueMNIST, and SemiAves for **semi-supervised learning**. We use USB [5] image classification task results. The best results are indicated in bold. Our results are averaged over 3 independent runs.

Datasets	CIFAR-100		STL-10		EuroSat		TissueMNIST		SemiAves
# Labels	200	400	40	100	20	40	80	400	3959
Pseudo-Label [186]	33.99±0.95	25.32±0.29	19.14±1.33	10.77±0.60	25.46±1.36	15.70±2.12	56.92±4.54	50.86±1.79	40.35±0.30
Mean-Teacher [275]	35.47±0.40	26.03±0.30	18.67±1.69	24.19±10.15	26.83±1.46	15.85±1.66	62.06±3.43	55.12±2.53	38.55±0.21
VAT [276]	31.49±1.33	21.34±0.50	18.45±1.47	10.69±0.51	26.16±0.96	10.09±0.94	57.49±5.47	51.30±1.73	38.82±0.04
MixMatch [188]	38.22±0.71	26.72±0.72	58.77±1.98	36.74±1.24	24.85±4.85	17.28±2.67	55.53±1.51	49.64±2.28	37.25±0.08
ReMixMatch [189]	22.21±2.21	16.86±0.57	13.08±3.34	**7.21±0.39**	**5.05±1.05**	5.07±0.56	58.77±4.43	49.82±1.18	**30.20±0.03**
AdaMatch [250]	22.32±1.73	16.66±0.62	13.64±2.49	7.62±1.90	7.02±0.79	**4.75±1.10**	58.35±4.87	52.40±2.08	31.75±0.13
FixMatch [3]	29.60±0.90	19.56±0.52	16.15±1.89	8.11±0.68	13.44±3.53	5.91±2.02	**55.37±4.50**	51.24±1.56	31.90±0.06
FlexMatch [191]	26.76±1.12	18.24±0.36	14.40±3.11	8.17±0.78	5.17±0.57	5.58±0.81	58.36±3.80	51.89±3.21	32.48±0.15
Dash [277]	30.61±0.98	19.38±0.10	16.22±5.95	7.85±0.74	11.19±0.90	6.96±0.87	56.98±2.93	51.97±1.55	32.38±0.16
CoMatch [251]	35.08±0.69	25.35±0.50	15.12±1.88	9.56±1.35	5.75±0.43	4.81±1.05	59.04±4.90	52.92±1.04	38.65±0.18
SimMatch [252]	23.78±1.08	17.06±0.78	11.77±3.20	7.55±1.86	7.66±0.60	5.27±0.89	60.88±4.31	52.93±1.56	33.85±0.08
FreeMatch [193]	**21.40±0.30**	**15.65±0.26**	12.73±3.22	8.52±0.53	6.50±0.78	5.78±0.51	58.24±3.08	52.19±1.35	32.85±0.31
SoftMatch [194]	22.67±1.32	16.84±0.66	13.55±3.16	7.84±1.72	5.75±0.62	5.90±1.42	57.98±3.66	51.73±2.84	31.80±0.22
Ours	22.06±1.06	17.40±1.04	**11.09±0.71**	8.10±1.02	5.86±1.06	5.74±1.13	57.99±2.16	**50.95±2.03**	33.08±0.26

Table D.8: Error rate comparison of different number of labels on IMDB, AG News, Amazon Review, Yahoo Answers, and Yelp Review for **semi-supervised learning**. We use USB [5] text classification task results. Best results are indicated in bold. Our results are averaged over 3 independent runs.

Datasets	IMDB		AG News		Amazon Review		Yahoo Answers		Yelp Review	
# Labels	20	100	40	200	250	1000	500	2000	250	1000
Pseudo-Label [186]	45.45±4.43	19.67±1.01	19.49±3.07	14.69±1.88	53.45±1.9	47.00±0.79	37.70±0.65	32.72±0.31	54.51±0.82	47.33±0.20
Mean-Teacher [275]	20.06±2.51	13.97±1.49	15.17±1.21	13.93±0.65	52.14±0.52	47.66±0.84	37.09±0.18	33.43±0.28	50.60±0.62	47.21±0.31
VAT [276]	25.93±2.58	11.61±1.79	14.70±1.19	11.71±0.84	49.83±0.46	46.54±0.31	34.87±0.41	31.50±0.35	52.97±1.41	45.30±0.32
MixMatch [188]	26.12±6.13	15.47±0.65	13.50±1.51	11.75±0.60	59.54±0.67	61.69±3.32	35.75±0.71	33.62±0.14	53.98±0.59	51.70±0.68
AdaMatch [250]	8.09±0.99	7.11±0.20	**11.73±0.17**	**11.22±0.95**	46.72±0.72	42.27±0.25	32.75±0.35	30.44±0.31	45.40±0.96	40.16±0.49
FixMatch [3]	7.72±0.33	7.33±0.13	30.17±1.87	11.71±1.95	47.61±0.83	43.05±0.54	**33.03±0.49**	30.51±0.53	46.52±0.94	40.65±0.46
FlexMatch [191]	7.82±0.77	7.41±0.38	16.38±3.94	12.08±0.73	45.73±1.60	42.25±0.33	35.61±1.08	31.13±0.18	**43.35±0.69**	40.51±0.34
Dash [277]	8.34±0.86	7.55±0.35	17.67±3.19	13.76±1.67	47.10±0.74	43.09±0.60	35.26±0.33	31.19±0.29	45.24±2.02	40.14±0.79
CoMatch [251]	7.44±0.30	7.72±1.14	11.95±0.76	10.75±0.35	48.76±0.90	43.36±0.21	33.48±0.51	30.25±0.35	45.40±1.12	40.27±0.51
SimMatch [252]	7.93±0.55	**7.08±0.33**	14.26±1.51	12.45±1.37	45.91±0.95	42.21±0.30	33.06±0.20	30.16±0.21	46.12±0.48	40.26±0.62
FreeMatch [193]	8.94±0.21	7.95±0.45	12.98±0.58	11.73±0.63	46.41±0.60	42.64±0.06	32.77±0.26	30.32±0.18	47.95±1.45	40.37±1.00
SoftMatch [194]	7.76±0.58	7.97±0.72	11.90±0.27	11.72±1.58	45.29±0.95	**42.21±0.20**	33.07±0.31	30.44±0.62	44.09±0.50	39.76±0.13
Ours	**7.32±0.12**	7.64±0.67	14.77±1.59	12.21±0.82	**43.96±0.32**	42.32±0.02	33.80±0.25	30.86±0.17	44.82±0.17	**39.67±0.71**

D.4.4 Noisy Label Learning

D.4.4.1 Setup

We conduct experiments of noisy label learning following SOP [4]. We evaluate the proposed method on both synthetic symmetric/asymmetric noise on CIFAR-10 and CIFAR-100, and more realistic and larger-scale instance noise on Clothing1M and WebVision. To introduce the synthetic symmetric noise to CIFAR-10 and CIFAR-100, we uniformly flip labels for a probability η into other classes. For asymmetric noise, we only randomly flip the labels for particular pairs of classes. For CIFAR-10 and CIFAR-100, we train PreAct-ResNet-18 with SGD using a learning rate of 0.02, a weight decay of $1e-3$, and a momentum of 0.9. We train for 300 epochs with a cosine learning rate schedule and a batch size of 128. For WebVision, we use InceptionResNet-v2 as

the backbone and set the batch size to 32. Other settings are similar to CIFAR-10. For Clothing1M, we use ImageNet-1K pre trained ResNet-50 as the backbone. We train it using SGD with an initial learning rate of 2e-3 for a total of 10 epochs, where the learning rate is reduced by 10 after 5 epochs. In addition, we also conducted experiments on CIFAR-10N and CIFAR-100N. We present the detailed hyper-parameters in Table D.9.

Table D.9: Hyper-parameters for **noisy label learning** used in experiments.

Hyper-parameter	CIFAR-10 (CIFAR-10N)	CIFAR-100 (CIFAR-100N)	Clothing1M	WebVision
Image Size	32	32	224	299
Model	PreAct-ResNet-18 (ResNet-34)	PreAct-ResNet-18 (ResNet-34)	ResNet-50 (ImageNet-1K Pretrained)	Inception-ResNet-v2
Batch Size	128	128	64	32
Learning Rate	0.02	0.02	0.002	0.02
Weight Decay	1e-3	1e-3	1e-3	5e-4
LR Scheduler	Cosine	Cosine	MultiStep	MultiStep
Training Epochs	300	300	10	100
Classes	10	100	14	50
Noisy Matrix Scale	1.0	2.0	0.5	2.5

D.4.4.2 Results

In addition to the results regarding noisy label learning provided in Chapter 9, we also present comparison results on CIFAR-10N and CIFAR-100N [257] in Table D.10. We include a full comparison on Clothing1M and WebVision, incorporating methods like Co-Teaching, Forward, and CORES, in Table D.11. As shown in Table D.10, the proposed ILL framework achieves performance comparable to the previous best method, SOP [4]. On CIFAR-10N, our method yields results very close to SOP in the Random and Aggregate case noise scenarios and surpasses SOP in the Worst case noise scenario. However, on CIFAR-100N, our method slightly underperforms previous methods, possibly due to the oversimplified noise model utilized in ILL. We believe that a more realistic noise transition model and further tuning of our method could lead to improved performance.

Table D.10: Test accuracy comparison of instance independent label noise on CIFAR-10N and CIFAR-100N for **noisy label learning**. The best results are indicated in **bold**, and the second best results are indicated in underline. Our results are averaged over three independent runs with ResNet34 as the backbone.

Dataset	CIFAR-10N						CIFAR-100N	
Noisy Type	Clean	Random 1	Random 2	Random 3	Aggregate	Worst	Clean	Noisy
CE	92.92±0.11	85.02±0.65	86.46±1.79	85.16±0.61	87.77±0.38	77.69±1.55	76.70±0.74	55.50±0.66
Forward [256]	93.02±0.12	86.88±0.50	86.14±0.24	87.04±0.35	88.24±0.22	79.79±0.46	76.18±0.37	57.01±1.03
Co-teaching [198]	93.35±0.14	90.33±0.13	90.30±0.17	90.15±0.18	91.20±0.13	83.83±0.13	73.46±0.09	60.37±0.27
DivideMix [203]	-	95.16±0.19	95.23±0.07	95.21±0.14	95.01±0.71	92.56±0.42	-	71.13±0.48
ELR [202]	95.39±0.05	94.43±0.41	94.20±0.24	94.34±0.22	94.83±0.10	91.09±1.60	78.57±0.12	66.72±0.07
CORES [305]	94.16±0.11	94.45±0.14	94.88±0.31	94.74±0.03	95.25±0.09	91.66±0.09	73.87±0.16	55.72±0.42
SOP [4]	**96.38±0.31**	95.28±0.13	95.31±0.10	95.39±0.11	95.61±0.13	93.24±0.21	**78.91±0.43**	67.81±0.23
Ours	96.21±0.29	**96.06±0.07**	**95.98±0.12**	**96.10±0.05**	**96.40±0.03**	**93.55±0.14**	78.53±0.21	**68.07±0.33**

Table D.11: Test accuracy comparison of realistic noisy labels on Clothing1M and WebVision for **noisy label learning**. The best results are indicated in **bold** and the second best results are indicated in underline. Our results are averaged over 3 independent runs. For Clothing1M, we use ImageNet-1K pre trained ResNet50 as the backbone. For WebVision, InceptionResNetv2 is used as the backbone.

Dataset	Clothing1M	WebVision
CE	69.10	-
Forward [256]	69.80	61.10
MentorNet [258]	66.17	63.00
Co-Teaching [198]	69.20	63.60
DivideMix [203]	**74.76**	77.32
ELR [202]	72.90	76.20
CORES [305]	73.20	-
SOP [4]	73.50	76.60
Ours	74.02±0.12	**79.37±0.09**

D.4.5 Mixed Imprecise Label Learning

D.4.5.1 Setup

To create a mixture of various imprecise label configurations, we select CIFAR-10 and CIFAR-100 as base datasets. We first uniformly sample l/C labeled samples from each class to form the labeled dataset and treat the remaining samples as the unlabeled dataset. Based on the labeled dataset, we generate partially labeled datasets by flipping negative labels to false positive labels with the partial ratio q. After obtaining the partial labels, we randomly select η percentage of samples from each class, and recreate the partial labels for them by flipping the ground truth label uniformly to another class. In this setting, unlabeled data, partially labeled data, and noisy labeled data exist simultaneously, which is very challenging and more closely resembles realistic situations. For CIFAR-10, we set $l \in \{1000, 5000, 50000\}$, and for CIFAR-100, we set $l \in \{5000, 10000, 50000\}$. Similarly in the partial label setting, we set $q \in \{0.1, 0.3, 0.5\}$ for CIFAR-10, and $q \in \{0.01, 0.05, 0.1\}$ for CIFAR-100. For noisy labels, we set $\eta \in \{0.1, 0.2, 0.3\}$ for both datasets.

D.4.5.2 Results

We provide a more complete version of Table 9.4 in Table D.12. On partial noisy labels of CIFAR-10 with partial ratio 0.5 and of CIFAR-100 with partial ratio 0.1, most baseline methods are more robust or even fail to perform. However, our ILL still shows very robust performance with minor performance degradation as increase of noise ratios.

Table D.12: Test accuracy comparison of **mixture of different imprecise labels**. We report results of full labels, partial ratio q of $\{0.1, 0.3, 0.5\}$ for CIFAR-10 and $\{0.01, 0.05, 0.1\}$ for CIFAR-100, and noise ratio η of $\{0.1, 0.2, 0.3\}$ for CIFAR-10 and CIFAR-100.

Dataset	# Labels	Partial Ratio q	Noise Ratio η	0	0.1	0.2	0.3
CIFAR-10	50,000	0.1	PiCO+ [259]	95.99±0.03	93.64	93.13	92.18
			IRNet [237]	-	93.44	92.57	92.38
			DALI [218]	-	94.15	94.04	93.77
			PiCO+ w/ Mixup [218]	-	94.58	94.74	94.43
			DALI w/ Mixup [218]	-	95.83	95.86	95.75
			Ours	**96.55**±0.08	**96.47**±0.11	**96.09**±0.20	**95.83**±0.05
		0.3	PiCO+ [259]	95.73±0.10	92.32	92.22	89.95
			IRNet [237]	-	92.81	92.18	91.35
			DALI [218]	-	93.44	93.25	92.42
			PiCO+ w/ Mixup [218]	-	94.02	94.03	92.94
			DALI w/ Mixup [218]	-	95.52	95.41	94.67
			Ours	**96.52**±0.12	**96.2**±0.02	**95.87**±0.14	**95.22**±0.06
		0.5	PiCO+ [259]	95.33±0.06	91.07	89.68	84.08
			IRNet [237]	-	91.51	90.76	86.19
			DALI [218]	-	92.67	91.83	89.8
			PiCO+ w/ Mixup [218]	-	93.56	92.65	88.21
			DALI w/ Mixup [218]	-	95.19	93.89	92.26
			Ours	**96.28**±0.13	**95.82**±0.07	**95.28**±0.08	**94.35**±0.08
CIFAR-100	50,000	0.01	PiCO+ [259]	76.29±0.42	71.42	70.22	66.14
			IRNet [237]	-	71.17	70.10	68.77
			DALI [218]	-	72.26	71.98	71.04
			PiCO+ w/ Mixup [218]	-	75.04	74.31	71.79
			DALI w/ Mixup [218]	-	76.52	76.55	76.09
			Ours	**78.08**±0.26	**77.53**±0.24	**76.96**±0.02	**76.43**±0.27
		0.05	PiCO+ [259]	76.17±0.18	69.40	66.67	62.24
			IRNet [237]	-	70.73	69.33	68.09
			DALI [218]	-	72.28	71.35	70.05
			PiCO+ w/ Mixup [218]	-	73.06	71.37	67.56
			DALI w/ Mixup [218]	-	76.87	75.23	74.49
			Ours	**76.95**±0.46	**77.07**±0.16	**76.34**±0.08	**75.13**±0.63
		0.1	PiCO+ [259]	75.55±0.21	-	-	-
			IRNet [237]	-	-	-	-
			DALI [218]	-	-	-	-
			PiCO+ w/ Mixup [218]	-	-	-	-
			DALI w/ Mixup [218]	-	-	-	-
			Ours	**76.41**±1.02	**75.50**±0.54	**74.67**±0.30	**73.88**±0.60

D.4.6 Ablation on Strong-Augmentation and Entropy Loss

We provide the ablation study on the strong-augmentation and entropy loss components here, which are common techniques in each setting [3, 2, 4]. For example, in SSL, strong-weak augmentation is an important strategy for SSL algorithms widely used in existing works such as FixMatch [3] and FlexMatch [191]. Thus, it is important to adopt strong-weak augmentation to achieve better performance in SSL [193, 194, 5]. This is similar in PLL settings [2, 185]. PiCO [2, 185] also used strong augmentation). Strong-weak augmentation and entropy loss are also adopted in SOP [4] of NLL. However, we found these techniques are less important for our formulation of NLL. We provide an

ablation study on the entropy loss of SSL, and both techniques for NLL and PLL here to demonstrate our discussions.

Table D.13: SSL ablation

	CIFAR100 l=200	STL10 l=40
Ours	22.06	11.09
Ours w/o ent.	22.41	11.23

Table D.14: PLL ablation

	CIFAR10 $q = 0.5$	CIFAR100 $q = 0.1$
PiCO	93.58	69.91
Ours	95.91	74.00
PiCO w/o s. a.	91.78	66.43
Ours w/o s. a.	94.53	72.69
Ours w/o ent.	95.87	73.75

Table D.15: NLL ablation

	CIFAR10 $\eta = 0.5$	CIFAR100 $\eta = 0.1$
SOP	94.00	63.30
Ours	94.31	66.46
SOP w/o s. a.	66.85	36.60
Ours w/o s. a.	93.56	65.89
SOP w/o ent.	93.04	62.85
Ours w/o ent.	94.16	66.12

Table D.16: Runtime Analysis on CIFAR-100

Setting	Algorithm	CIFAR-100 Avg. Runtime (s/iter)
SSL	FreeMatch	0.2157
SSL	Ours	0.1146
PLL	PiCO	0.3249
PLL	Ours	0.2919
NLL	SOP	0.1176
NLL	Ours	0.1021

D.4.7 Runtime Analysis

We provide the runtime analysis on CIFAR-100 of our method in different settings, compared with the SOTA baselines. We compute the average runtime from all training iterations on CIFAR-100. The results are shown in Table D.16. Our method in general present faster runtime without complex design such as contrastive loss.

Bibliography

[1] Irene Martín-Morató, Toni Heittola, Annamaria Mesaros, and Tuomas Virtanen. Low-complexity acoustic scene classification for multi-device audio: Analysis of dcase 2021 challenge systems. *arXiv preprint arXiv:2105.13734*, 2021.

[2] Haobo Wang, Ruixuan Xiao, Yixuan Li, Lei Feng, Gang Niu, Gang Chen, and Junbo Zhao. PiCO: Contrastive label disambiguation for partial label learning. In *International Conference on Learning Representations (ICLR)*, 2022.

[3] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

[4] Sheng Liu, Zhihui Zhu, Qing Qu, and Chong You. Robust training under label noise by over-parameterization. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 14153–14172. PMLR, 17–23 Jul 2022.

[5] Yidong Wang, Hao Chen, Yue Fan, Wang Sun, Ran Tao, Wenxin Hou, Renjie Wang, Linyi Yang, Zhi Zhou, Lan-Zhe Guo, Heli Qi, Zhen Wu, Yu-Feng Li, Satoshi Nakamura, Wei Ye, Marios Savvides, Bhiksha Raj, Takahiro Shinozaki, Bernt Schiele, Jindong Wang, Xing Xie, and Yue Zhang. Usb: A unified semi-supervised learning benchmark. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[6] J. P. Bello, C. Silva, O. Nov, R. L. DuBois, A. Arora, J. Salamon, C. Mydlarz, and H. Doraiswamy. SONYC: A system for the monitoring, analysis and mitigation of urban noise pollution. *Communications of the ACM*, In press, 2018.

[7] Juan Pablo Bello, Charlie Mydlarz, and Justin Salamon. Sound analysis in smart cities. In *Computational Analysis of Sound Scenes and Events*, pages 373–397. Springer, 2018.

[8] Regunathan Radhakrishnan, Ajay Divakaran, and A Smaragdis. Audio analysis for surveillance applications. In *Proc. WASPAA*, pages 158–161. IEEE, 2005.

[9] Erling Wold, Thom Blum, Douglas Keislar, and James Wheaten. Content-based classification, search, and retrieval of audio. *IEEE multimedia*, 3(3):27–36, 1996.

[10] Qin Jin, Peter Schulam, Shourabh Rawat, Susanne Burger, Duo Ding, and Florian Metze. Event-based video retrieval using audio. In *Proc. Interspeech*, 2012.

[11] Romain Serizel, Nicolas Turpault, Hamid Eghbal-Zadeh, and Ankit Parag Shah. Large-scale weakly labeled semi-supervised sound event detection in domestic environments. *arXiv preprint arXiv:1807.10501*, 2018.

[12] Christian Debes, Andreas Merentitis, Sergey Sukhanov, Maria Niessen, Nikolaos Frangiadakis, and Alexander Bauer. Monitoring activities of daily living in smart homes: Understanding human behavior. *IEEE Signal Processing Magazine*, 33(2):81–94, 2016.

[13] Yaniv Zigel, Dima Litvak, and Israel Gannot. A method for automatic fall detection of elderly people using floor vibrations and sound—proof of concept on human mimicking doll falls. *IEEE Transactions on Biomedical Engineering*, 56(12):2858–2867, 2009.

[14] Maximin Coavoux, Shashi Narayan, and Shay B. Cohen. Privacy-preserving neural representations of text, 2018.

[15] Thore Graepel, Kristin Lauter, and Michael Naehrig. Ml confidential: Machine learning on encrypted data. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *Information Security and Cryptology – ICISC 2012*, pages 1–21, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[16] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[17] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark, 2016.

[18] Lu Jiang, Di Huang, Mason Liu, and Weilong Yang. Beyond synthetic noise: Deep learning on controlled noisy labels, 2020.

[19] Sourish Chaudhuri. *Structured Models for Semantic Analysis of Audio Content*. PhD thesis, PhD thesis, Carnegie Mellon University. 46, 47, 2013.

[20] Anurag Kumar and Bhiksha Raj. Weakly supervised scalable audio content analysis. In *Multimedia and Expo (ICME), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[21] Ankit Shah, Anurag Kumar, Alexander G Hauptmann, and Bhiksha Raj. A closer look at weak label learning for audio events. *arXiv preprint arXiv:1804.09288*, 2018.

[22] Cisco Systems Inc. Cisco visual networking index: Forecast and methodology, 2016–2021. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html, –.

[23] Dimitrios Giannoulis, Emmanouil Benetos, Dan Stowell, Mathias Rossignol, Mathieu Lagrange, and Mark D Plumbley. Detection and classification of acoustic scenes and events: an IEEE AASP challenge. In *2013 IEEE WASPAA*, pages 1–4. IEEE, 2013.

[24] Tuomas Virtanen, Annamaria Mesaros, Toni Heittola, Mark D. Plumbley, Peter Foster, Emmanouil Benetos, and Mathieu Lagrange. *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*. Tampere University of Technology. Department of Signal Processing, 2016.

[25] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen. DCASE 2017 challenge setup: Tasks, datasets and baseline system. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, pages 85–92, November 2017.

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

[27] Tom M Mitchell, William W Cohen, Estevam R Hruschka Jr, Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, et al. Never ending learning. In *AAAI*, pages 2302–2310, 2015.

[28] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Neil: Extracting visual knowledge from web data. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[29] Richard F Lyon. Machine hearing: An emerging field [exploratory dsp]. *Ieee signal processing magazine*, 27(5):131–139, 2010.

[30] Anurag Kumar and Bhiksha Raj. Audio event detection using weakly labeled data. In *24th ACM International Conference on Multimedia*. ACM Multimedia, 2016.

[31] Benjamin Elizalde, Guan-Lin Chao, Ming Zeng, and Ian Lane. City-identification of flickr videos using semantic acoustic features. In *Multimedia Big Data (BigMM), 2016 IEEE Second International Conference on*, pages 303–306. IEEE, 2016.

[32] Sebastian Sager, Damian Borth, Benjamin Elizalde, Christian Schulze, Bhiksha Raj, Ian Lane, and Andreas Dengel. Audiosentibank: Large-scale semantic ontology of acoustic concepts for audio content analysis. *arXiv preprint arXiv:1607.03766*, 2016.

[33] Anurag Kumar, Bhiksha Raj, and Ndapandula Nakashole. Discovering sound concepts and acoustic relations in text. *arXiv preprint arXiv:1609.07384*, 2016.

[34] PAFY: retrieve YouTube content and metadata. https://pypi.python.org/pypi/pafy.

[35] Dan Ellis, Tuomas Virtanen, Mark D. Plumbley, and Bhiksha Raj. *Future Perspective*. Springer International Publishing, 2018.

[36] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. TUT database for acoustic scene classification and sound event detection. In *24th European Signal Processing Conference 2016 (EUSIPCO 2016)*, Budapest, Hungary, 2016.

[37] Karol J Piczak. Environmental sound classification with convolutional neural networks. In *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*, pages 1–6. IEEE, 2015.

[38] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[39] Wenjing Han, Eduardo Coutinho, Huabin Ruan, Haifeng Li, Björn Schuller, Xiaojie Yu, and Xuan Zhu. Semi-supervised active learning for sound classification in hybrid learning environments. *PloS one*, 11(9):e0162075, 2016.

[40] Ankit Shah, Rohan Badlani, Anurag Kumar, Benjamin Elizalde, and Bhiksha Raj. An approach for self-training audio event detectors using web data. *arXiv preprint arXiv:1609.06026*, 2016.

[41] Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. Clap learning audio concepts from natural language supervision. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[42] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[43] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[44] Justin Salamon, Duncan MacConnell, Mark Cartwright, Peter Li, and Juan Pablo Bello. Scaper: A library for soundscape synthesis and augmentation. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA). New Paltz, NY, USA*, 2017.

[45] Rohan Badlani, Ankit Shah, Benjamin Elizalde, Anurag Kumar, and Bhiksha Raj. Framework for evaluation of sound event detection in web videos. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3096–3100. IEEE, 2018.

[46] Benjamin Elizalde, Anurag Kumar, Ankit Shah, Rohan Badlani, Emmanuel Vincent, Bhiksha Raj, and Ian Lane. Experiments on the DCASE Challenge 2016: Acoustic scene classification and sound event detection in real life recording. In *DCASE2016 Workshop on Detection and Classification of Acoustic Scenes and Events*, 2016.

[47] Romain Serizel, Nicolas Turpault, Hamid Eghbal-Zadeh, and Ankit Parag Shah. Large-scale weakly labeled semi-supervised sound event detection in domestic environments. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, pages 19–23, November 2018.

[48] Nicolas Turpault, Romain Serizel, Ankit Parag Shah, and Justin Salamon. Sound event detection in domestic environments with weakly labeled data and soundscape synthesis. Technical report, Université de Lorraine, CNRS, Inria, Loria, France, June 2019. working paper or preprint.

[49] Ines Nolasco, Shubhr Singh, E Vidana-Villa, Emily Grout, J Morford, M Emmerson, F Jensens, Helen Whitehead, Ivan Kiskin, Ariana Strandburg-Peshkin, et al. Few-shot bioacoustic event detection at the dcase 2022 challenge. *arXiv preprint arXiv:2207.07911*, 2022.

[50] John Martinsson, Martin Willbo, Aleksis Pirinen, Olof Mogren, and Maria Sandsten. Few-shot bioacoustic event detection using an event-length adapted ensemble of prototypical networks. In *DCASE*, 2022.

[51] Haohe Liu, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Wenwu Wang, and Mark D Plumbley. Surrey system for dcase 2022 task 5: Few-shot bioacoustic event detection with segment-level metric learning. *arXiv preprint arXiv:2207.10547*, 2022.

[52] Fei-Fei Li and P. Perona. The perceived position of moving objects: Transcranial magnetic stimulation of area MT+ reduces the flash-lag effect. In *IEEE CVPR*, volume 2, 2005.

[53] J Gauvain and C Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *Speech and audio processing, IEEE Trans. on*, 1994.

[54] Benjamin Elizalde and Gerald Friedland. Lost in segmentation: Three approaches for speech/non-speech detection in consumer-produced videos. In *2013 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2013.

[55] Benjamin Elizalde, Gerald Friedland, Howard Lei, and Ajay Divakaran. There is No Data Like Less Data: Percepts for Video Concept Detection on Consumer-Produced Media. In *ACM International Workshop on Audio and Multimedia Methods for Large-Scale Video Analysis at ACM Multimedia*, 2012.

[56] Randal S. Olson, Ryan J. Urbanowicz, Peter C. Andrews, et al. Automating biomedical data science through tree-based pipeline optimization. In *Proceedings of the 18th European Conference on the Applications of Evolutionary and Bio-inspired Computation*, Lecture Notes in Computer Science. Springer-Verlag, 2016.

[57] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, and others. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[58] Jitong Chen, Yuxuan Wang, and DeLiang Wang. Noise perturbation improves supervised speech separation. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 83–90. Springer, 2015.

[59] Romain Serizel, Nicolas Turpault, Hamid Eghbal-Zadeh, and Ankit Parag Shah. Large-Scale Weakly Labeled Semi-Supervised Sound Event Detection in Domestic Environments. Proc. DCASE Workshop, July 2018.

[60] Romain Serizel and Nicolas Turpault. Sound Event Detection from Partially Annotated Data: Trends and Challenges. In *IcETRAN conference*, Srebrno Jezero, Serbia, June 2019.

[61] Frederic Font, Gerard Roma, and Xavier Serra. Freesound technical demo. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 411–412. ACM, 2013.

[62] Eduardo Fonseca, Jordi Pons Puig, Xavier Favory, Frederic Font Corbera, Dmitry Bogdanov, Andres Ferraro, Sergio Oramas, Alastair Porter, and Xavier Serra. Freesound datasets: a platform for the creation of open audio datasets. In *Hu X, Cunningham SJ, Turnbull D, Duan Z, editors. Proceedings of the 18th ISMIR Conference; 2017 oct 23-27; Suzhou, China.[Canada]: International Society for Music Information Retrieval; 2017. p. 486-93.*, 2017.

[63] Gert Dekkers, Steven Lauwereins, Bart Thoen, Mulu Weldegebreal Adhana, Henk Brouckxon, Toon van Waterschoot, Bart Vanrumste, Marian Verhelst, and Peter Karsmakers. The SINS database for detection of daily activities in a home environment using an acoustic sensor network. In *Proc. DCASE Workshop*, pages 32–36, November 2017.

[64] E. J. Humphrey, J. Salamon, O. Nieto, J. Forsyth, R. Bittner, and J. P. Bello. JAMS: A JSON annotated music specification for reproducible MIR research. In *15th Int. Soc. for Music Info. Retrieval Conf.*, pages 591–596, Taipei, Taiwan, Oct. 2014.

[65] David Snyder, Guoguo Chen, and Daniel Povey. MUSAN: A Music, Speech, and Noise Corpus, 2015. arXiv:1510.08484v1.

[66] Matthias Mauch and Sebastian Ewert. The audio degradation toolbox and its application to robustness evaluation. In *Proc. ISMIR*, pages 83–88, 2013.

[67] Lu JiaKai. Mean teacher convolution system for dcase 2018 task 4. Technical report, DCASE2018 Challenge, September 2018.

[68] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proc. NeurIPS*, page 10, 2017.

[69] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Metrics for polyphonic sound event detection. *Applied Sciences*, 6(6):162, May 2016.

[70] Liwei Lin and Xiangdong Wang. Guided learning convolution system for dcase 2019 task 4. Technical report, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, June 2019.

[71] Lionel Delphin-Poulat and Cyril Plapous. Mean teacher with data augmentation for dcase 2019 task 4. Technical report, Orange Labs Lannion, France, June 2019.

[72] Ziqiang Shi. Hodgepodge: Sound event detection based on ensemble of semi-supervised learning methods. Technical report, Fujitsu Research and Development Center, Beijing, China, June 2019.

[73] Liwei Lin, Xiangdong Wang, Hong Liu, and Yueliang Qian. What you need is a more professional teacher. *arXiv preprint arXiv:1906.02517*, 2019.

[74] Léo Cances, Thomas Pellegrini, and Patrice Guyot. Multi task learning and post processing optimization for sound event detection. Technical report, IRIT, Université de Toulouse, CNRS, Toulouse, France, June 2019.

[75] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[76] Sandeep Kothinti, Gregory Sell, Shinji Watanabe, and Mounya Elhilali. Integrated bottom-up and top-down inference for sound event detection. Technical report, Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, USA, June 2019.

[77] Justin Salamon and Juan Pablo Bello. Feature learning with deep scattering for urban sound analysis. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 724–728. IEEE, 2015.

[78] Anthony Agnone and Umair Altaf. Virtual adversarial training system for dcase 2019 task 4. Technical report, Pindrop, Audio Research department, Atlanta, United States, June 2019.

[79] Teck Kai Chan, Cheng Siong Chin, and Ye Li. Non-negative matrix factorization-convolution neural network (nmf-cnn) for sound event detection. Technical report, Newcastle University, Singapore, June 2019.

[80] Yu Kiyokawa, Sakiko Mishima, Takahiro Toizumi, Kazutoshi Sagi, Reishi Kondo, and Toshiyuki Nomura. Sound event detection with resnet and self-mask module for dcase 2019 task 4. Technical report, Data Science Research Laboratories, NEC Corporation, Japan, June 2019.

[81] Wootaek Lim, Sangwon Suh, Sooyoung Park, and Youngho Jeong. Sound event detection in domestic environments using ensemble of convolutional recurrent neural networks. Technical report, Realistic AV Research Group, Electronics and Telecommunications Research Institute, Daejeon, Korea, June 2019.

[82] Jie Yan and Yan Song. Weakly labeled sound event detection with resdual crnn using semi-supervised method. Technical report, University of Science and Technology of China, National Engineering Laboratory for Speech and Language Information Processing, Hefei, China, June 2019.

[83] Zhenyuan Zhang Zhang, Mingxue Yang, and Li Liu. An improved system for dcase 2019 challenge task 4. Technical report, University of Electronic Science and Technology of China ence and Technology of China, School of Information and Communication Engineering, Chengdu, China, June 2019.

[84] Nicolas Turpault, Romain Serizel, Ankit Parag Shah, and Justin Salamon. Sound event detection in domestic environments with weakly labeled data and soundscape synthesis. working paper or preprint, July 2019.

[85] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Tut database for acoustic scene classification and sound event detection. In *Signal Processing Conference (EUSIPCO), 2016 24th European*, pages 1128–1132. IEEE, 2016.

[86] Christian Zieger and Maurizio Omologo. Acoustic event detection-itc-irst aed database. *Internal ITC report*, 2005.

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

[88] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1041–1044. ACM, 2014.

[89] Anurag Kumar and Bhiksha Raj. Audio event detection using weakly labeled data. In *Proceedings of the 2016 ACM on Multimedia Conference*, MM '16, pages 1038–1047, New York, NY, USA, 2016. ACM.

[90] Zhiwu Lu, Zhenyong Fu, Tao Xiang, Peng Han, Liwei Wang, and Xin Gao. Learning from weak and noisy labels for semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(3):486–500, 2017.

[91] Jinhui Tang, Shuicheng Yan, Richang Hong, Guo-Jun Qi, and Tat-Seng Chua. Inferring semantic concepts from community-contributed images and noisy tags. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 223–232. ACM, 2009.

[92] Zheyun Feng, Songhe Feng, Rong Jin, and Anil K Jain. Image tag completion by noisy matrix recovery. In *European Conference on Computer Vision*, pages 424–438. Springer, 2014.

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

[94] Szu-Yu Chou, Jyh-Shing Roger Jang, and Yi-Hsuan Yang. Learning to recognize transient sound events using attentional supervision. In *IJCAI*, pages 3336–3342, 2018.

[95] Anurag Kumar, Maksim Khadkevich, and Christian Fügen. Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 326–330. IEEE, 2018.

[96] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 131–135. IEEE, 2017.

[97] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[98] C. Buckley and E. Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 25–32. ACM, 2004.

[99] Tom Fawcett. Roc graphs: Notes and practical considerations for researchers. *Machine learning*, 31(1):1–38, 2004.

[100] Y. Wu and T. Lee. Reducing Model Complexity for DNN Based Large-Scale Audio Classification. *ArXiv e-prints*, November 2017.

[101] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017.

[102] Dawei Zhu, Xiaoyu Shen, Marius Mosbach, Andreas Stephan, and Dietrich Klakow. Weaker than you think: A critical look at weakly supervised learning. *arXiv preprint arXiv:2305.17442*, 2023.

[103] Xinlei Chen and Abhinav Gupta. Webly supervised learning of convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[104] Junwei Liang, Lu Jiang, Deyu Meng, and Alexander Hauptmann. Learning to detect concepts from webly-labeled video data. IJCAI, 2016.

[105] Santosh K Divvala, Ali Farhadi, and Carlos Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3270–3277, 2014.

[106] Rob Fergus, Li Fei-Fei, Pietro Perona, and Andrew Zisserman. Learning object categories from internet image searches. *Proceedings of the IEEE*, 98(8):1453–1466, 2010.

[107] Yan Xia, Xudong Cao, Fang Wen, and Jian Sun. Well begun is half done: Generating high-quality seeds for automatic image dataset construction from web. In *European Conference on Computer Vision*, pages 387–400. Springer, 2014.

[108] Anurag Kumar. *Acoustic Intelligence in Machines*. PhD thesis, Carnegie Mellon University, 2018.

[109] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 2014.

[110] Anurag Kumar, Bhiksha Raj, and Ndapandula Nakashole. Discovering sound concepts and acoustic relations in text. *submitted IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[111] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. *arXiv preprint arXiv:1803.11364*, 2018.

[112] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *International Conference on Learning Representations (ICLR)*, 2016.

[113] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[114] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2016.

[115] Eran Malach and Shai Shalev-Shwartz. Decoupling" when to update" from" how to update". In *Advances in Neural Information Processing Systems*, pages 960–970, 2017.

[116] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: robust training deep neural networks with extremely noisy labels. 2018.

[117] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[118] Shiliang Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7-8):2031–2038, 2013.

[119] Anurag Kumar, M. Khadkevich, and C. Fugen. Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes. In *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE, 2018.

[120] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[121] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. Clustering with bregman divergences. *Journal of machine learning research*, 6(Oct), 2005.

[122] Han Zhang, Tao Xu, Mohamed Elhoseiny, Xiaolei Huang, Shaoting Zhang, Ahmed Elgammal, and Dimitris Metaxas. Spda-cnn: Unifying semantic part detection and abstraction for fine-grained recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1143–1152, 2016.

[123] Yong Xu, Qiuqiang Kong, Qiang Huang, Wenwu Wang, and Mark D Plumbley. Attention and localization based on a deep convolutional recurrent model for weakly supervised audio tagging. *arXiv preprint arXiv:1703.06052*, 2017.

[124] Marc-André Carbonneau, Veronika Cheplygina, Eric Granger, and Ghyslain Gagnon. Multiple instance learning: A survey of problem characteristics and applications. *Pattern Recognition*, 77:329–353, 2018.

[125] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997.

[126] James Richard Foulds and Eibe Frank. A review of multi-instance learning assumptions. 2010.

[127] Qingping Tao, Stephen Scott, NV Vinodchandran, Thomas Takeo Osugi, and Brandon Mueller. An extended kernel for generalized multiple-instance learning. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 272–277. IEEE, 2004.

[128] Qingping Tao, Stephen Scott, NV Vinodchandran, and Thomas Takeo Osugi. Svm-based generalized multiple-instance learning via approximate box counting. In *Proceedings of the twenty-first international conference on Machine learning*, page 101, 2004.

[129] Nikolaos Pappas and Andrei Popescu-Belis. Explaining the stars: Weighted multiple-instance learning for aspect-based sentiment analysis. In *Proceedings of the 2014 Conference on Empirical Methods In Natural Language Processing (EMNLP)*, pages 455–466, 2014.

[130] Kiri L Wagstaff and Terran Lane. Salience assignment for multiple-instance regression. 2007.

[131] Nikolaos Pappas and Andrei Popescu-Belis. Explicit document modeling through weighted multiple-instance learning. *Journal of Artificial Intelligence Research*, 58:591–626, 2017.

[132] Saravanan Subramanian, Santu Rana, Sunil Gupta, P Bagavathi Sivakumar, C Shunmuga Velayutham, and Svetha Venkateshc. Bayesian nonparametric multiple instance regression. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3661–3666. IEEE, 2016.

[133] Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. *Advances in neural information processing systems*, 10:570–576, 1997.

[134] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *Advances in neural information processing systems*, pages 577–584, 2003.

[135] Boris Babenko, Piotr Dollár, Zhuowen Tu, and Serge Belongie. Simultaneous learning and alignment: Multi-instance and multi-pose learning. 2008.

[136] Gary Doran and Soumya Ray. A theoretical and empirical analysis of support vector machine methods for multiple-instance classification. *Machine learning*, 97(1-2):79–102, 2014.

[137] Gitte Vanwinckelen, Daan Fierens, Hendrik Blockeel, et al. Instance-level accuracy versus bag-level accuracy in multi-instance learning. *Data mining and knowledge discovery*, 30(2):313–341, 2016.

[138] Gabriel Dulac-Arnold, Neil Zeghidour, Marco Cuturi, Lucas Beyer, and Jean-Philippe Vert. Deep multi-class learning from label proportions. *arXiv preprint arXiv:1905.12909*, 2019.

[139] Felix X Yu, Dong Liu, Sanjiv Kumar, Tony Jebara, and Shih-Fu Chang. Svm for learning with label proportions. *arXiv preprint arXiv:1306.0886*, 2013.

[140] David R Musicant, Janara M Christensen, and Jamie F Olson. Supervised learning by training on aggregate outputs. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 252–261. IEEE, 2007.

[141] Novi Quadrianto, Alex J Smola, Tiberio S Caetano, and Quoc V Le. Estimating labels from label proportions. *Journal of Machine Learning Research*, 10(10), 2009.

[142] Kai Fan, Hongyi Zhang, Songbai Yan, Liwei Wang, Wensheng Zhang, and Jufu Feng. Learning a generative classifier from label proportions. *Neurocomputing*, 139:47–55, 2014.

[143] Giorgio Patrini, Richard Nock, Paul Rivera, and Tiberio Caetano. (almost) no label no cry. In *Advances in Neural Information Processing Systems*, pages 190–198, 2014.

[144] Felix X Yu, Liangliang Cao, Michele Merler, Noel Codella, Tao Chen, John R Smith, and Shih-Fu Chang. Modeling attributes from category-attribute proportions. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 977–980, 2014.

[145] Ehsan Mohammady Ardehaly and Aron Culotta. Co-training for demographic classification using deep learning from label proportions. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 1017–1024. IEEE, 2017.

[146] Gerda Bortsova, Florian Dubost, Silas Ørting, Ioannis Katramados, Laurens Hogeweg, Laura Thomsen, Mathilde Wille, and Marleen de Bruijne. Deep learning from label proportions for emphysema quantification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 768–776. Springer, 2018.

[147] Jerzy Letkowski. Applications of the poisson probability distribution. In *Proc. Acad. Business Res. Inst. Conf*, pages 1–11, 2012.

[148] Kuen Han Tsai and Hsuan Tien Lin. Learning from Label Proportions with Consistency Regularization. *arXiv*, pages 1–16, 2019.

[149] Sanath Narayan, Hisham Cholakkal, Fahad Shahbaz Khan, and Ling Shao. 3c-net: Category count and center loss for weakly-supervised action localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8679–8687, 2019.

[150] D. Crouse. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52:1679–1696, 2016.

[151] Alireza Ghasemi, Hamid R Rabiee, Mohsen Fadaee, Mohammad T Manzuri, and Mohammad H Rohban. Active learning from positive and unlabeled data. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 244–250. IEEE, 2011.

[152] Shuang Ma, Zhaoyang Zeng, Daniel McDuff, and Yale Song. Active contrastive learning of audio-visual video representations. *arXiv preprint arXiv:2009.09805*, 2020.

[153] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*, 2019.

[154] Yukun Chen and Subramani Mani. Active learning for unbalanced data in the challenge with multiple models and biasing. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pages 113–126. JMLR Workshop and Conference Proceedings, 2011.

[155] Weishi Shi and Qi Yu. Integrating bayesian and discriminative sparse kernel machines for multi-class active learning. *Advances in neural information processing systems*, 2019.

[156] Jingyu Shao, Qing Wang, and Fangbing Liu. Learning to sample: an active learning framework. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 538–547. IEEE, 2019.

[157] Anxiang Zhang, Ankit Shah, and Bhiksha Raj. Training image classifiers using semi-weak label data, 2021.

[158] Yongqi Zhang, Quanming Yao, Yingxia Shao, and Lei Chen. Nscaching: simple and efficient negative sampling for knowledge graph embedding. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 614–625. IEEE, 2019.

[159] Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. Understanding negative sampling in graph representation learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1666–1676, 2020.

[160] Kian Ahrabian, Aarash Feizi, Yasmin Salehi, William L Hamilton, and Avishek Joey Bose. Structure aware negative sampling in knowledge graphs. *arXiv preprint arXiv:2009.11355*, 2020.

[161] Yuan Gong, Yu-An Chung, and James Glass. Psla: Improving audio event classification with pretraining, sampling, labeling, and aggregation. *arXiv preprint arXiv:2102.01243*, 2021.

[162] Konstantinos Sikelis, George E. Tsekouras, and Konstantinos Kotis. Ontology-based feature selection: A survey. *Future Internet*, 13(6), 2021.

[163] Yuxia Geng, Jiaoyan Chen, Zhuo Chen 007, Jeff Z. Pan, Zhiquan Ye, Zonggang Yuan, Yantao Jia, and Huajun Chen. Ontozsl: Ontology-enhanced zero-shot learning. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021.* ACM / IW3C2, 2021.

[164] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[165] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.

[166] M. Sheraz Anjum Kamran Munir. The use of ontologies for effective knowledge modelling and information retrieval. *Applied Computing and Informatics 14 (2018)*, 2018.

[167] Gordon Wichern, Brandon Mechtley, Alex Fink, Harvey Thornburg, and Andreas Spanias. An ontological framework for retrieving environmental sounds using semantics and acoustic content. *EURASIP J. Audio Speech Music Process.*, 2010(1), December 2010.

[168] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[169] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson. CNN architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[170] Bhiksha Raj Abelino Jimenez, Benjamin Elizalde. Sound event classification using ontology-based neural networks. *NIPS 2018 Workshop*, 2018.

[171] Harsh Shrivastava, Yfang Yin, Rajiv Ratn Shah, and Roger Zimmermann. Mt-gcn for multi-label audio-tagging with noisy labels. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 136–140, 2020.

[172] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. Multi-label image recognition with graph convolutional networks, 2019.

[173] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[174] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[175] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. NeurIPS*, pages 5998–6008, December 2017.

[176] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[177] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[178] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021.

[179] OpenAI. Gpt-4 technical report. 2023.

[180] Timothee Cour, Ben Sapp, and Ben Taskar. Learning from partial labels. *The Journal of Machine Learning Research*, 12:1501–1536, 2011.

[181] Jie Luo and Francesco Orabona. Learning from candidate labeling sets. *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

[182] Lei Feng, Jiaqi Lv, Bo Han, Miao Xu, Gang Niu, Xin Geng, Bo An, and Masashi Sugiyama. Provably consistent partial-label learning. *ArXiv*, abs/2007.08929, 2020.

[183] Dengbao Wang, Min-Ling Zhang, and Li Li. Adaptive graph guided disambiguation for partial label learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:8796–8811, 2019.

[184] Hongwei Wen, Jingyi Cui, Hanyuan Hang, Jiabin Liu, Yisen Wang, and Zhouchen Lin. Leveraged weighted loss for partial label learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 11091–11100. PMLR, 2021.

[185] Dong-Dong Wu, Deng-Bao Wang, and Min-Ling Zhang. Revisiting consistency regularization for deep partial label learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 24212–24225. PMLR, 17–23 Jul 2022.

[186] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, page 896, 2013.

[187] Laine Samuli and Aila Timo. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations (ICLR)*, page 6, 2017.

[188] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[189] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *International Conference on Learning Representations (ICLR)*, 2019.

[190] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 10687–10698, 2020.

[191] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

[192] Xudong Wang, Zhirong Wu, Long Lian, and Stella X Yu. Debiased learning from naturally imbalanced pseudo-labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pages 14647–14657, 2022.

[193] Yidong Wang, Hao Chen, Qiang Heng, Wenxin Hou, Yue Fan, , Zhen Wu, Jindong Wang, Marios Savvides, Takahiro Shinozaki, Bhiksha Raj, Bernt Schiele, and Xing Xie. Freematch: Self-adaptive thresholding for semi-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2023.

[194] Hao Chen, Ran Tao, Yue Fan, Yidong Wang, Jindong Wang, Bernt Schiele, Xing Xie, Bhiksha Raj, and Marios Savvides. Softmatch: Addressing the quantity-quality trade-off in semi-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2023.

[195] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2691–2699, 2015.

[196] Alan Joseph Bekker and Jacob Goldberger. Training deep neural-networks based on unreliable labels. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2682–2686, 2016.

[197] Aritra Ghosh, Himanshu Kumar, and P. Shanti Sastry. Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[198] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Wai-Hung Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[199] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[200] Junnan Li, Yongkang Wong, Qi Zhao, and M. Kankanhalli. Learning to learn from noisy labeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5046–5054, 2018.

[201] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 322–330, 2019.

[202] Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

[203] Junnan Li, Richard Socher, and Steven C.H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2020.

[204] Xingjun Ma, Hanxun Huang, Yisen Wang, Simone Romano, Sarah Monazam Erfani, and James Bailey. Normalized loss functions for deep learning with noisy labels. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[205] Yivan Zhang, Gang Niu, and Masashi Sugiyama. Learning noise transition matrix from only noisy labels via total variation regularization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[206] Shahana Ibrahim, Tri Nguyen, and Xiao Fu. Deep learning from crowdsourced labels: Coupled cross-entropy minimization, identifiability, and regularization. In *International Conference on Learning Representations (ICLR)*, 2023.

[207] Jiaheng Wei, Zhaowei Zhu, Tianyi Luo, Ehsan Amid, Abhishek Kumar, and Yang Liu. To aggregate or not? learning with separate noisy labels. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Aug 2023.

[208] Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. A survey on programmatic weak supervision, 2022.

[209] Renzhi Wu, Shen-En Chen, Jieyu Zhang, and Xu Chu. Learning hyper label model for programmatic weak supervision. In *International Conference on Learning Representations (ICLR)*, 2023.

[210] Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *International Conference on Machine Learning (ICML)*, pages 2127–2136. PMLR, 2018.

[211] Nan Lu, Gang Niu, Aditya Krishna Menon, and Masashi Sugiyama. On the minimal supervision for training any binary classifier from only unlabeled data. *arXiv preprint arXiv:1808.10585*, 2018.

[212] Clayton Scott and Jianxin Zhang. Learning from label proportions: A mutual contamination framework. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:22256–22267, 2020.

[213] Y. Zhang, N. Charoenphakdee, Z. Wu, and M. Sugiyama. Learning from aggregate observations. pages 7993–8005, 2020.

[214] Saurabh Garg, Yifan Wu, Alex Smola, Sivaraman Balakrishnan, and Zachary C. Lipton. Mixture proportion estimation and pu learning: A modern approach, 2021.

[215] Lei Feng, Senlin Shu, Nan Lu, Bo Han, Miao Xu, Gang Niu, Bo An, and Masashi Sugiyama. Pointwise binary classification with pairwise confidence comparisons. In *International Conference on Machine Learning*, pages 3252–3262. PMLR, 2021.

[216] Andrea Campagner. Learnability in "learning from fuzzy labels". In *2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE, 2021.

[217] Zheng Lian, Mingyu Xu, Lan Chen, Licai Sun, Bin Liu, and Jianhua Tao. Arnet: Automatic refinement network for noisy partial label learning. *arXiv preprint arXiv:2211.04774*, 2022.

[218] Mingyu Xu, Zheng Lian, Lei Feng, Bin Liu, and Jianhua Tao. Dali: Dynamically adjusted label importance for noisy partial label learning, 2023.

[219] Qian-Wei Wang, Yu-Feng Li, and Zhi-Hua Zhou. Partial label learning with unlabeled data. In *IJCAI*, pages 3755–3761, 2019.

[220] Wei Wang and Min-Ling Zhang. Semi-supervised partial label learning via confidence-rated margin maximization. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6982–6993, 2020.

[221] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

[222] Jiaqi Lv, Miao Xu, Lei Feng, Gang Niu, Xin Geng, and Masashi Sugiyama. Progressive identification of true labels for partial-label learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 6500–6510. PMLR, 2020.

[223] Chidubem Arachie and Bert Huang. Constrained labeling for weakly supervised learning. In *Uncertainty in Artificial Intelligence*, pages 236–246. PMLR, 2021.

[224] Eyke Hüllermeier and Weiwei Cheng. Superset learning based on generalized loss minimization. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part II 15*, pages 260–275. Springer, 2015.

[225] Jiaqi Lv, Biao Liu, Lei Feng, Ning Xu, Miao Xu, Bo An, Gang Niu, Xin Geng, and Masashi Sugiyama. On the robustness of average losses for partial-label learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–15, 2023.

[226] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.

[227] Thierry Denœux. Maximum likelihood estimation from fuzzy data using the em algorithm. *Fuzzy sets and systems*, 183(1):72–91, 2011.

[228] Eyke Hüllermeier. Learning from imprecise and fuzzy observations: Data disambiguation through generalized loss minimization. *International Journal of Approximate Reasoning*, 55(7):1519–1534, 2014.

[229] Benjamin Quost and Thierry Denoeux. Clustering and classification of fuzzy data using the fuzzy em algorithm. *Fuzzy Sets and Systems*, 286:134–156, 2016.

[230] Brendan Van Rooyen and Robert C Williamson. A theory of learning with corrupted labels. *J. Mach. Learn. Res.*, 18(1):8501–8550, 2017.

[231] Chen Gong, Jian Yang, Jane You, and Masashi Sugiyama. Centroid estimation with guaranteed efficiency: A general framework for weakly supervised learning.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(6):2841–2855, 2020.

[232] Chao-Kai Chiang and Masashi Sugiyama. Unified risk analysis for weakly supervised learning. *arXiv preprint arXiv:2309.08216*, 2023.

[233] Zixi Wei, Lei Feng, Bo Han, Tongliang Liu, Gang Niu, Xiaofeng Zhu, and Heng Tao Shen. A universal unbiased method for classification from aggregate observations. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 36804–36820. PMLR, 23–29 Jul 2023.

[234] Zheng Xie, Yu Liu, Hao-Yuan He, Ming Li, and Zhi-Hua Zhou. Weakly supervised auc optimization: A unified partial auc approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[235] Andrea Campagner. Learning from fuzzy labels: Theoretical issues and algorithmic solutions. *International Journal of Approximate Reasoning*, page 108969, 2023.

[236] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020.

[237] Zheng Lian, Mingyu Xu, Lan Chen, Licai Sun, Bin Liu, and Jianhua Tao. Irnet: Iterative refinement network for noisy partial label learning, 2022.

[238] Eyke Hüllermeier and Jürgen Beringer. Learning from ambiguously labeled examples. *Intelligent Data Analysis*, 10(5):419–439, 2006.

[239] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[240] Baixu Chen, Junguang Jiang, Ximei Wang, Jianmin Wang, and Mingsheng Long. Debiased pseudo labeling in self-training. *arXiv preprint arXiv:2202.07136*, 2022.

[241] David MacKay John Bridle, Anthony Heading. Unsupervised classifiers, mutual information and 'phantom targets. *Advances in Neural Information Processing Systems (NeurIPS)*, 1991.

[242] Armand Joulin and Francis Bach. A convex relaxation for weakly supervised classifiers. In *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2012.

[243] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[244] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 2010.

[245] Lei Feng, Takuo Kaneko, Bo Han, Gang Niu, Bo An, and Masashi Sugiyama. Learning with multiple complementary labels. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3072–3081. PMLR, 2020.

[246] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*. British Machine Vision Association, 2016.

[247] Ning Xu, Congyu Qiao, Xin Geng, and Min-Ling Zhang. Instance-dependent partial label learning. *Advances in Neural Information Processing Systems*, 34:27119–27130, 2021.

[248] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[249] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172, 2013.

[250] David Berthelot, Rebecca Roelofs, Kihyuk Sohn, Nicholas Carlini, and Alex Kurakin. Adamatch: A unified approach to semi-supervised learning and domain adaptation. *International Conference on Learning Representations (ICLR)*, 2021.

[251] Junnan Li, Caiming Xiong, and Steven CH Hoi. Comatch: Semi-supervised learning with contrastive graph regularization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pages 9475–9484, 2021.

[252] Mingkai Zheng, Shan You, Lang Huang, Fei Wang, Chen Qian, and Chang Xu. Simmatch: Semi-supervised learning with similarity matching. *arXiv preprint arXiv:2203.06915*, 2022.

[253] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2691–2699, 2015.

[254] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data, 2017.

[255] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[256] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2233–2241, 2016.

[257] Jiaheng Wei, Zhaowei Zhu, Hao Cheng, Tongliang Liu, Gang Niu, and Yang Liu. Learning with noisy labels revisited: A study using real-world human annotations. *arXiv preprint arXiv:2110.12088*, 2021.

[258] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International conference on machine learning*, pages 2304–2313. PMLR, 2018.

[259] Haobo Wang, Ruixuan Xiao, Yixuan Li, Lei Feng, Gang Niu, Gang Chen, and Junbo Zhao. Pico+: Contrastive label disambiguation for robust partial label learning, 2022.

[260] Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer. *arXiv preprint arXiv:2104.01778*, 2021.

[261] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pages 3122–3130. PMLR, 2018.

[262] Thomas Varsavsky, Mauricio Orbes-Arteaga, Carole H Sudre, Mark S Graham, Parashkev Nachev, and M Jorge Cardoso. Test-time unsupervised domain adaptation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part I 23*, pages 428–436. Springer, 2020.

[263] Chen Wang, Zhenjiang Miao, and Xiao Meng. Differential mfcc and vector quantization used for real-time speaker recognition system. In *Image and Signal Processing, 2008. CISP'08. Congress on*, volume 5, pages 319–323. IEEE, 2008.

[264] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[265] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. *Advances in neural information processing systems*, 34:14200–14213, 2021.

[266] Sanyuan Chen, Yu Wu, Chengyi Wang, Shujie Liu, Daniel Tompkins, Zhuo Chen, and Furu Wei. Beats: Audio pre-training with acoustic tokenizers. *arXiv preprint arXiv:2212.09058*, 2022.

[267] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, pages 702–703, 2020.

[268] Min-Ling Zhang and Fei Yu. Solving the partial label learning problem: An instance-based approach. In *IJCAI*, pages 4048–4054, 2015.

[269] Chen Gong, Tongliang Liu, Yuanyan Tang, Jian Yang, Jie Yang, and Dacheng Tao. A regularization approach for instance-based superset label learning. *IEEE transactions on cybernetics*, 48(3):967–978, 2017.

[270] Ning Xu, Jiaqi Lv, and Xin Geng. Partial label learning via label enhancement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5557–5564, 2019.

[271] Zhenguo Wu, Jiaqi Lv, and Masashi Sugiyama. Learning with proper partial labels. *Neural Computation*, 35(1):58–81, Jan 2023.

[272] Nam Nguyen and Rich Caruana. Classification with partial labels. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 551–559, 2008.

[273] Min-Ling Zhang, Bin-Bin Zhou, and Xu-Ying Liu. Partial label learning via feature-aware disambiguation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1335–1344, 2016.

[274] Liping Liu and Thomas Dietterich. A conditional multinomial mixture model for superset label learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 25, 2012.

[275] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

[276] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2018.

[277] Yi Xu, Lei Shang, Jinxing Ye, Qi Qian, Yu-Feng Li, Baigui Sun, Hao Li, and Rong Jin. Dash: Semi-supervised learning with dynamic thresholding. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 11525–11536. PMLR, 2021.

[278] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, 2019.

[279] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V Le. Meta pseudo labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11557–11568, 2021.

[280] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[281] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–19, 2022.

[282] Xingjun Ma, Hanxun Huang, Yisen Wang, Simone Romano, Sarah Erfani, and James Bailey. Normalized loss functions for deep learning with noisy labels. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[283] Yaodong Yu, Kwan Ho Ryan Chan, Chong You, Chaobing Song, and Yi Ma. Learning diverse and discriminative representations via the principle of maximal coding rate reduction, 2020.

[284] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. In *IEEE Transactions on pattern analysis and machine intelligence*, pages 447–461, 2016.

[285] Curtis Northcutt, Lu Jiang, and Isaac Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411, Apr 2021.

[286] Yivan Zhang, Gang Niu, and Masashi Sugiyama. Learning noise transition matrix from only noisy labels via total variation regularization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 12501–12512. PMLR, 2021.

[287] Yingbin Bai, Erkun Yang, Bo Han, Yanhua Yang, Jiatong Li, Yinian Mao, Gang Niu, and Tongliang Liu. Understanding and improving early stopping for learning with noisy labels, 2021.

[288] Junnan Li, Caiming Xiong, and Steven CH Hoi. Learning from noisy data with robust representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pages 9485–9494, 2021.

[289] Shikun Li, Xiaobo Xia, Shiming Ge, and Tongliang Liu. Selective-supervised contrastive learning with noisy labels. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2022.

[290] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[291] Hwanjun Song, Minseok Kim, and Jae-Gil Lee. SELFIE: Refurbishing unclean samples for robust deep learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 5907–5915. PMLR, 09–15 Jun 2019.

[292] Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction, 2019.

[293] Vinay Shukla, Zhe Zeng, Kareem Ahmed, and Guy Van den Broeck. A unified approach to count-based weakly-supervised learning. In *ICML 2023 Workshop on Differentiable Almost Everything: Differentiable Relaxations, Algorithms, Operators, and Simulators*, jul 2023.

[294] Massih-Reza Amini and Patrick Gallinari. Semi-supervised logistic regression. In *ECAI*, volume 2, page 11, 2002.

[295] Ning Xu, Jiaqi Lv, Biao Liu, Congyu Qiao, and Xin Geng. Progressive purification for instance-dependent partial label learning, 2022.

[296] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[297] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

[298] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[299] Jiancheng Yang, Rui Shi, and Bingbing Ni. Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis. In *IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pages 191–195, 2021.

[300] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2: A large-scale lightweight benchmark for 2d and 3d biomedical image classification. *arXiv preprint arXiv:2110.14795*, 2021.

[301] Jong-Chyi Su and Subhransu Maji. The semi-supervised inaturalist-aves challenge at fgvc7 workshop. *arXiv preprint arXiv:2103.06937*, 2021.

[302] Yelp dataset: http://www.yelp.com/dataset_challenge.

[303] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems (NeurIPS)*, 28:649–657, 2015.

[304] Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. Importance of semantic representation: Dataless classification. In *AAAI*, volume 2, pages 830–835, 2008.

[305] Hao Cheng, Zhaowei Zhu, Xingyu Li, Yifei Gong, Xing Sun, and Yang Liu. Learning with instance-dependent label noise: A sample sieve approach. *arXiv preprint arXiv:2010.02347*, 2020.