Université de BLIDA 1
Faculté des Sciences
Département d'Informatique

Master Thesis
Option : Ingénierie des Logiciels

SEMI-SUPERVISED LEARNING FOR MULTI-LABEL AUDIO TAGGING

By:
AMROUCHE Djamel Eddine
ALLALI Adil

In front of a jury composed of:

Mr. CHIKHI Nasim fateh	President
Ms. HADJ HENNI Malika	Examiner
Ms. YKHLEF Hadjer	Supervisor
Ms. DIFFALLAH Zhor	Supervisor

2020/2021

Abstract

Audio Tagging is concerned with the development of systems that are able to recognize sound events. A growing interest is geared towards audio tagging for various applications such as acoustic surveillance, tagging video content and environmental scene recognition. Our goal is to design an audio tagging system capable of recognizing a wide range of sound events. The development process usually requires a large set of labeled sound data. However, most existing datasets are unlabeled since hand-labeling is a very costly and a time-consuming process, and it involves a lot of manual labor. To mend with this, we have built our audio tagging system following the Semi-Supervised Learning (SSL) paradigm. Specifically, we have chosen the pseudo-labeling strategy to learn from weakly labeled data. In addition, our system trains a ResNet deep learning model on log-mel spectrograms, along with augmentation techniques to increase the dataset size. The training uses the cyclic cosine annealing technique for the learning rate. We have carried out our experiments on a huge dataset made of sound recordings; we have investigated the impact of the sharpening temperature (a hyperparameter of our system) on the distribution of the pseudo-labels, and have tested ensembling various variants of our approach. The results demonstrate the efficacy of pseudo-labeling SSL strategy. Furthermore, ensembling various systems significantly boosts the overall performance.

Keywords: Audio Tagging, Semi-Supervised Learning, Feature Extraction, Deep Learning, Ensemble Learning, Statistical Tests.

Résumé

L’étiquetage audio est concerné par le développement de systèmes capables de reconnaître des événements de son. Un intérêt croissant est porté à l’étiquetage audio pour diverses applications telles que la surveillance acoustique, l’étiquetage de contenu vidéo et la reconnaissance de scènes environnementales. Notre objectif est de concevoir un système d’étiquetage audio capable de reconnaître un large ensemble d'événements sonores. Le processus de développement nécessite généralement un grand ensemble de données sonores étiquetées. Cependant, la plupart des ensembles de données existants ne sont pas étiquetés, car l'étiquetage manuel est un processus très coûteux et très long, et il implique beaucoup de travail manuel. Pour remédier à cela, nous avons construit notre système de l’étiquetage audio en suivant le paradigme de l'apprentissage semi-supervisé (SSL). Plus précisément, nous avons choisi la stratégie de pseudo-étiquetage pour apprendre à partir de données faiblement étiquetées. En outre, notre système entraîne un modèle d'apprentissage profond ResNet sur log mel spectrogrammes, ainsi que des techniques d'augmentation pour augmenter la taille de l'ensemble de données. L'apprentissage utilise la technique de recuit cosinus cyclique pour déterminer le taux d'apprentissage approprié. Nous avons réalisé nos expériences sur un ensemble énorme de données constitué d'enregistrements de sonores ; nous avons étudié l'impact de la température d'affûtage (un hyperparamètre de notre système) sur la distribution des pseudo-étiquettes, et nous avons testé de combiner de diverses variantes de notre approche. Les résultats démontrent l'efficacité de la stratégie SSL de pseudo-étiquetage. De plus, la combinaison de plusieurs systèmes augmente la performance globale.

Mots clé: L’étiquetage audio, Apprentissage semi-supervisé, Extractions des caractéristiques, Apprentissage approfondi, Apprentissage d'ensemble, Tests statistiques

الملخص

تهدف أنظمة وضع العلامات الصوتية إلى التعرف على الأصوات الموجودة في المقاطع صوتية. وفي الوقت الحالي يزداد الإهتمام بهذا المجال لتطبيقاته المختلفة مثل المراقبة الصوتية و التعرف على الأصوات الموجودة في الفديوهات و تحديد الأصوات الموجودة في البيئة. هدفنا هو تصميم نظام للتعرف على مجموعة واسعة من الأحداث الصوتية. تتطلب عملية التطوير عادةً مجموعة كبيرة من البيانات الصوتية ذات العلامات. لكن معظم مجموعات البيانات الحالية غير مصنفة نظرًا لأن وضع العلامات يدويا عملية مكلفة للغاية وتستغرق وقتًا طويلاً ، ولتدارك هذه المشاكل ، لقد قمنا بإنشاء نظامنا الخاص بإتباع نموذج التعلم شبه الخاضع للإشراف (SSL) . على وجه التحديد ، اخترنا استراتيجية وضع العلامات الزائفة (pseudo-labeling) للتعلم من البيانات ذات العلامات الضعيفة. وبالإضافة إلى ذلك , يقوم نظامنا بتدريب نموذج التعلم العميق ResNet على مخططات طيفية لوغاريتمية "Log-Mel Spectrogram"، إلى جانب تقنيات التعزيز لزيادة حجم مجموعة البيانات الصوتية. يستخدم التدريب تقنية (cyclic cosine anealing) لتغير معدل التعلم. لقد أجرينا تجاربنا على مجموعة بيانات ضخمة من التسجيلات صوتية ، ولقد قمنا بدراسة تأثير الشحذ على توزيع العلامات الزائفة ، وإختبرنا تجميع المتغيرات المختلفة من نهجنا. و النتائج تظهر مدى فعالية إستراتيجية وضع العلامات الزائفة (pseudo-labeling) من التعلم شبه الخاضع للإشراف (SSL) . علاوة على ذلك ، فإن تجميع أنظمة وضع العلامات المختلفة يعزز الأداء العام بشكل كبير.

الكلمات المفتاحية : العلامات الصوتية، التعلم شبه الخاضع للإشراف، إستخراج الخصائص، التعلم العميق، فرقة التعلم، الاختبارات الإحصائية.

ACKNOWLEDGMENTS

First and foremost, we have to thank our research supervisors, Ms. YKHLEF Hadjer and Ms. DIFFALLAH Zhor. Without their assistance and dedicated involvement in every step throughout the process, this thesis would have never been accomplished. We would like to thank you very much for your support and understanding over these past few months.

Finally, many thanks to all the people who have contributed to the completion of our research work directly or indirectly.

Contents

List of Figures

List of Tables

INTRODUCTION

On a daily basis, humans rely on sounds to know their surroundings and enhance their scene understanding (e.g. streets, factory, car passing by, car horn, …etc.). More broadly, sound complements visual information such as videos and images. There are many types of sounds, and almost always in environments there are multiple sources producing sounds simultaneously. The task of recognizing sounds is not considered difficult for humans since we are able to discern and classify audio without conscious effort [1]. Machines in various environments have the ability to hear, such as smartphones, autonomous robots, or security systems. However, enabling devices to make sense of their environment through the analysis of sounds is a complex task, but achieving the automation of sound recognition can benefit humans greatly. In machine learning, audio tagging systems are capable of recognizing and discerning a wide range of acoustic events and audio scenes, these systems are trained by using a large amount of audio data to achieve high accuracy when recognizing and discerning a wide range of acoustic events and audio scenes.

The prospect of human-like sound understanding or audio tagging could open up a range of applications, including intelligent monitoring systems of equipment using acoustic information, acoustic surveillance, cataloging, search in audio archives, tag video content or recognize sound events happening in real time. Audio tagging has been implemented in many applications such as audio information retrieval [2], audio classification [3], acoustic scene recognition [4], industry sound [5] and music tagging [6].

Machine learning requires a large number of labeled data to achieve great performance. However, these datasets have to be hand-labeled by specialists, which is a very costly and time-consuming process since it involves a lot of manual labor [7]. This consideration causes a difficulty in collecting enough amounts of audio training data for building tagging systems. Furthermore, a large amount of user-generated audio content is available on the web, which can be resourceful for audio tagging research. Nevertheless, because these resources are either poorly labeled or unlabeled, this can decrease the

audio tagging system performance when used in a supervised fashion. To take advantage of plenty of web audio resources, the semi-supervised learning paradigm can use both labeled and unlabeled data to achieve great audio tagging performance.

Audio Tagging using semi-supervised learning has been addressed before. Eric Bouteillon [8] has presented a novel data augmentation technique for multi-label audio tagging named **SpecMix**. In the paper, the new augmentation technique **SpecMix** is an extension of SpecAugment [9] inspired by Mixup [10]. The author presented a semi-supervised **warm-up pipeline** by filtering unreliable samples in a multi-stage process by using a self-training technique. This latter refers to retraining a model based on its own predictions on unlabeled data. Xiaofeng Hong and Gang Liu [11] used a semi-supervised learning method called Interpolation Consistency Training (ICT) [12]. ICT encourages the prediction at an interpolation of unlabeled points to be consistent with the interpolation of the predictions at those points. Other researchers [13][14] used well-known semi-supervised techniques in their papers like pseudo-labeling and MixMatch [15].

Many researchers have experimented using different techniques and concepts to build audio tagging systems. The design and evaluation of such systems is actually a more complicated task, and should be conducted properly in order to ensure significance of results (i.e. avoid deriving conclusions affected by chance). In our case, we have used the pseudo-labeling technique on a large dataset, and we have used a sharpening function proposed in MixMatch [16] to sharpen the pseudo-labels distribution. Furthermore, we have employed the concept of thresholding to produce better distribution of pseudo-labels that better suits multi-label classification. However, further research is still needed to properly assess the effects of the latter techniques and hyperparameters on the performance of audio tagging systems. Motivated by these needs, we have designed an audio tagging system that uses the pseudo-labeling strategy. Additionally, we have conducted extensive experiments and analyzed the behavior of multiple variants of our audio tagging system. We can summarize our contributions as follows:

- We have carried out our experiments using a recent dataset FSDKaggle2019, and we have backed our conducted experimental comparisons using well-known statistical tests.
- We have designed our audio tagging systems using a well-known deep neural network architecture, the ResNet-34 [17], which has been used successfully in audio-related tasks [18][19].
- We have analyzed the impact of the temperature hyperparameter of sharpening on the distribution of pseudo-labels.
- We have used Ensemble learning by averaging the scores of our audio tagging systems.
- We have studied the behavior of the ResNet neural network model and cosine annealing, by increasing the number of training epochs.

The rest of this thesis is structured as follows. **Chapter 1** reviews the multi-label audio tagging pipeline, starting from data preparation to model evaluation. **Chapter 2** introduces some semi-supervised learning methodologies that have been widely invoked for audio tagging. **Chapter 3** presents the process of our audio tagging system. It summarizes the steps that we have followed to design our model, including preprocessing, feature extraction, data augmentation and the training stages. **Chapter 4** reports the obtained results through performance tables and plots. Finally, Conclusion summarizes the contributions of this thesis, the lines of limitations and future work.

Chapter 1 : Overview of Multi-Label Audio Tagging

1.1. Introduction

In recent years, the popularity of sound recognition has increased and gained a lot of attention due to the incredible potential and adoption of machine learning in many fields. Sound recognition encompasses tasks such as acoustic scene classification, sound event detection and **audio tagging** [20]. The latter is becoming a popular task with its great impact on many real-life applications such as (acoustic monitoring, hearing aids, virtual reality, videoconferencing, video games, automated sound recognition, etc...). The popularity of audio tagging is growing by virtue of its increasing performance with the advances of machine learning.

In this chapter, we provide some basic notions on audio tagging (also known as multi-label audio classification) that are necessary for understanding the remaining of this thesis. Specifically, we discuss the major steps for building an audio tagging system, namely **sound representation** i.e. **features extraction** and **model learning**. We also present some scientific and technical challenges that researchers face every day in the field of audio tagging.

1.2. Multi-label Audio Tagging

Audio tagging is a task where the main goal aims at performing **multi-label** classification on audio recordings by assigning one or more labels to it [20]. Tagging is equivalent to multi-label classification in machine learning terminology. Audio Tagging has drawn a lot of attention due to its remarkable application in many different fields like multimedia sharing sites (e.g., YouTube, … etc.) [21], emotion detection in music [22] and music genres classification [23].

The audio tagging process consists of two main steps: **audio signal processing** and **multi-label classification**. First, signal processing is the step where the audio data undergoes a **pre-processing** stage to enhance its property by removing silence, making audio files the same duration and reducing noises to make audio files fairly equal [24]. In

addition, this step involves **feature extraction**, where the audio signal goes through a transformation process into different representations of its physical properties (i.e. **time**, **frequency** and loudness). These representations provide helpful information about those properties to use in the next step, it is worth mentioning that using a suitable representation for some tasks can improve the accuracy because the representations carry the acoustic content [25][26]. Next is **audio classification** using a **multi-label** approach, the previous step provides the system with the necessary inputs to train the learning model. The goal is to make accurate predictions on unseen data. The overall audio tagging process is depicted in Figure 1.1.

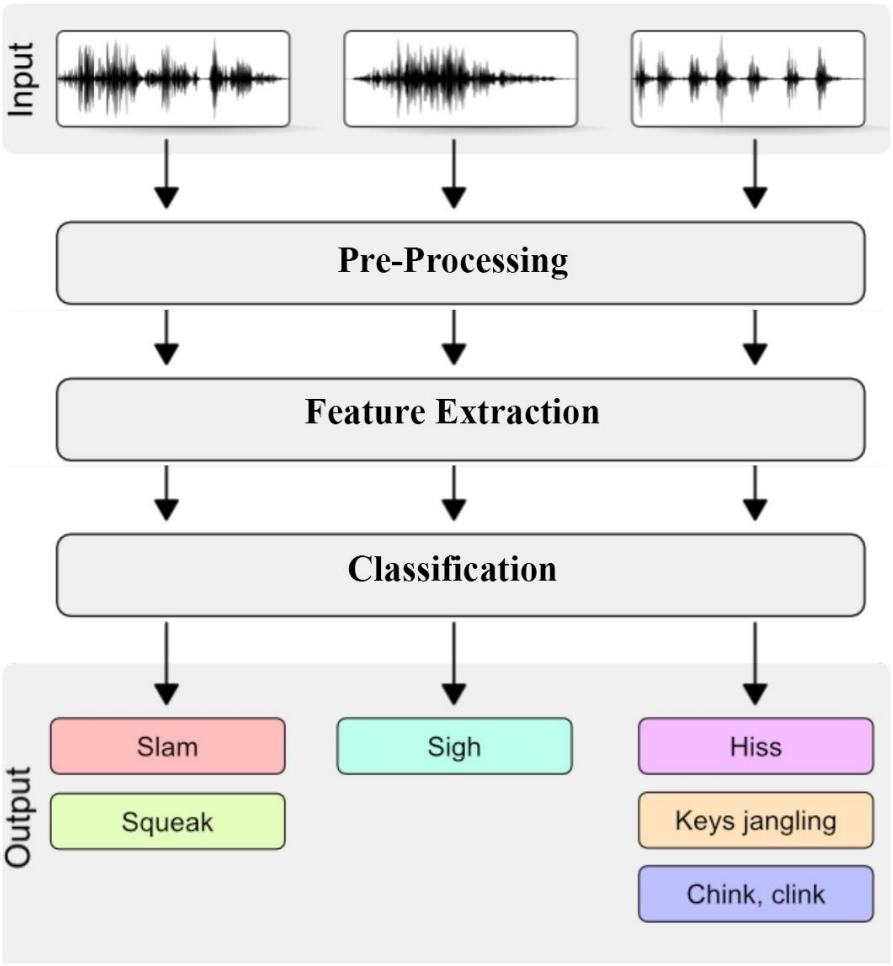


Figure 1.1: Audio Tagging process [27].

1.3. Audio Signal Representation

The physical representation of sound is a pressure wave (i.e., **vibration**) in the air which can be measured with a **mechanical** device. The arrival of electronic technology has introduced the ability to convert the pressure waves into a voltage reading that can be transferred onto a variety of storage media. The pressure waves (i.e. sounds) are converted from a **continuous** status to a **discrete** status (i.e. digitized signals) for the machines to process it [28]. The basic representation of sound is by the **amplitude** $\boldsymbol{A}$ of its vibration over **time** as a **waveform**, where sound is as a changing function of **time** $\boldsymbol{t}$ that is denoted as $\boldsymbol{x(t)}$ as shown in Figure 1.2.

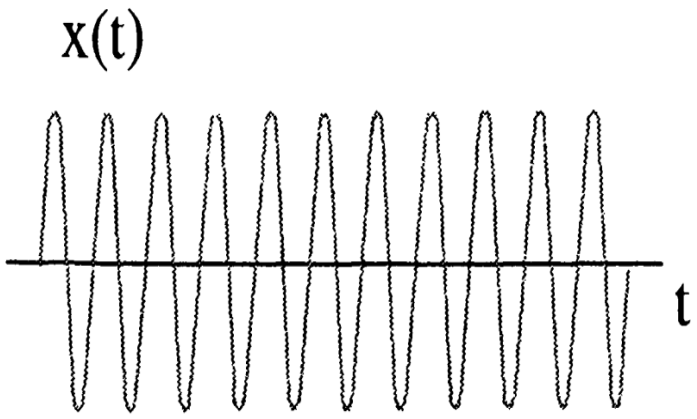


Figure 1.2: Time-Domain representation of a sound tone [29].

One of the main characteristics of sound that the human ear can distinct is the **tonal content** that can be high or low pitched, this is the effect of **frequencies** stored in the sound [29]. Humans perceive sounds in terms of their tonal content, in many situations it is appropriate to describe audio signals in the **frequency** domain, where the representation that corresponds to the frequency domain shows how much **frequency** $\boldsymbol{f}$ is present in the signal. Figure 1.3 depicts the frequency representation of a sound tone.

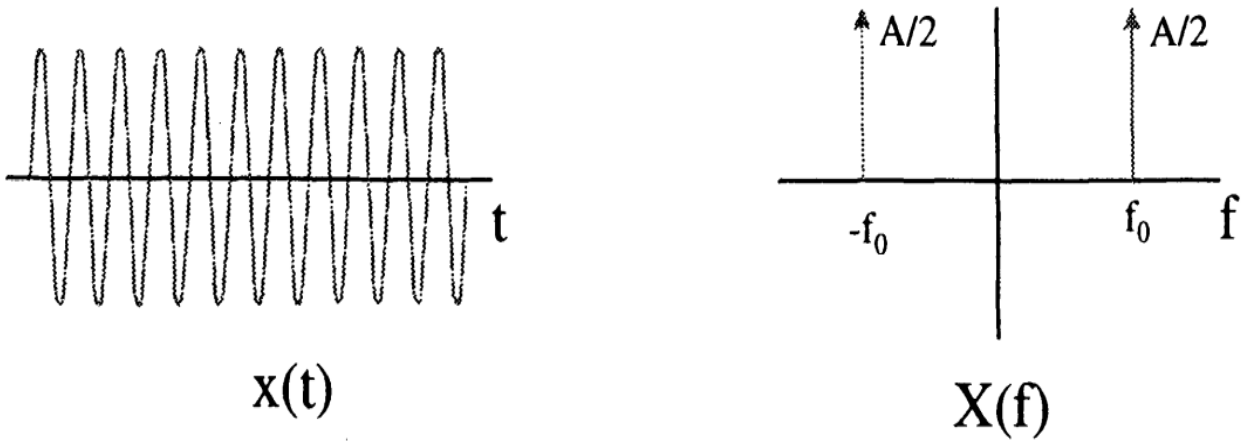


Figure 1.3: Time-domain versus frequency representation of a sound tone of amplitude A at frequency f_0 [29].

1.4. Feature Extraction Methods

For audio tagging, a significant amount of information is contained in the relative distribution of energy in **frequency** of an audio signal [20]. The information stored in the frequencies allows for making comparisons between audio files while paying attention to the most relevant **characteristics** of the audio. **Feature extraction** is a crucial process in the development of the audio tagging system, the audio signal gets transformed from the default representation (i.e. **waveform**) into representations that maximizes the sound recognition performance of the audio tagging system [20].

For the essential audio tagging features to be extracted, the audio signal must be converted to the **frequency-domain** representation. To do that, the mathematical function **Fourier Transform** allows the passage of the audio signal from the **time-domain** representation to the **frequency** representation. The Fourier Transform is the basic tool for converting a signal from its representation in time **x(t)** into a corresponding representation in frequency **X(f)**. The Fourier Transform is defined as:

$$X(f) \equiv \int_{-\infty}^{\infty} x(t) e^{-j2\pi f t} dt \tag{1.1}$$

where the notation is as follow:

-j to denote the square root of -1 (the imaginary number).

-d is an infinitesimal or a differential “Delta”.

The default Fourier transform is used for **continuous** signals, but because the audio signal is stored as **discrete** and periodic values (i.e. digitized signals), the **Discrete Fourier Transform** (i.e. **DFT**) is used with the digitized audio signal. Given the sequence $x_0, \ldots, x_N$ of N complex-valued measurements, the **DFT** is defined by the formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \qquad k = 0, \ldots, N-1 \tag{1.2}$$

where $e^{i2\pi/N}$ is a primitive N^{th} root of 1.

The **Fast Fourier transform** (i.e. **FFT**) is a more efficient and much faster implementation of DFT [30]. Figure 1.4 shows the result of applying the Fast Fourier Transform (i.e. **FFT**) on an audio signal.

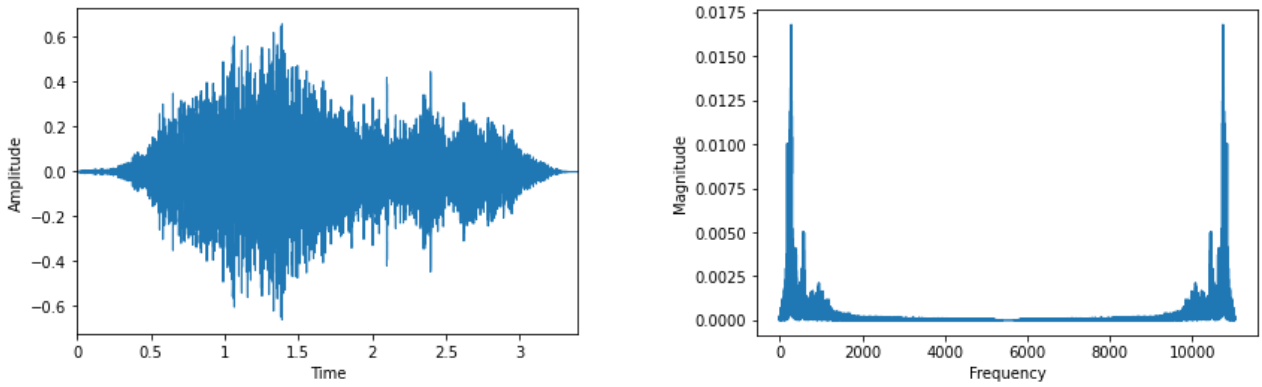


Figure 1.4: (a) Time-domain versus (b) Frequency-domain representation of a "Applause, Crowd, Cheering" audio file.

The **time-domain** and the **frequency-domain** representations do not really carry the information that is suitable for audio tagging, but the **time-frequency** representation (i.e. change of frequency over time) is more suitable for audio tagging. One of the most popular time-frequency representations are **spectrograms**, the Fourier transform does convert the audio from time-domain to frequency-domain but it does not cover the change of the frequency over time. That is why the concept of **Short-Time Fourier transform** (i.e. **STFT**) is used to solve the problem. the **STFT** considers only

specific time segments (i.e. **frames**) of the signal $\boldsymbol{x(t)}$, which are obtained by applying a **Window** function $\boldsymbol{w(t)}$ to $\boldsymbol{x(t)}$:

$$x_w(t_0, t) = x(t)w(t - t_0) \tag{1.3}$$

A common **Window** function is hamming window, plotted in Figure 1.5.

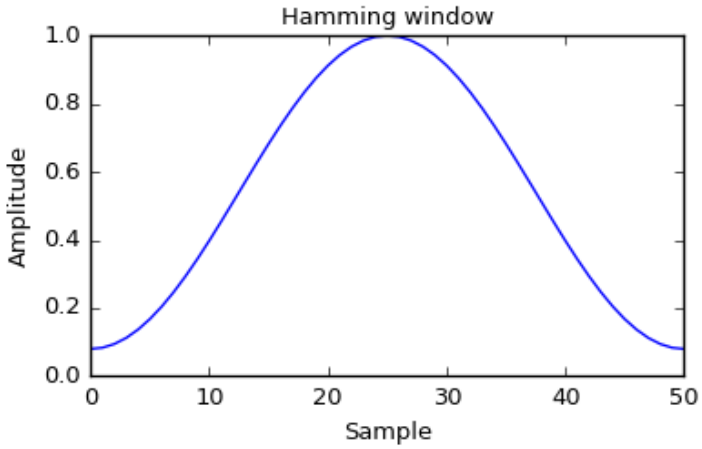


Figure 1.5: Hamming Window [31].

The mathematical STFT representation is as follows:

$$\boldsymbol{STFT}\{x(t)\}(m, \omega) \equiv X(m, \omega) = \sum_{-\infty}^{\infty} x(n)w(n - m)e^{-j\omega n} \tag{1.4}$$

Where the notation is as follow

-j = square root of -1.

-ω = frequency.

-$x(n)$ = input signal at time n.

-$w(n)$ = length M window function (e.g., Hamming).

-$X(m, \omega)$ is basically the Fourier transform of $x_w(t_0, t) = x(t)w(t - t_0)$.

STFT is applying **FFT** to small **overlapped** audio signal segments weighted by a **hamming Window**, this process is called **Frame Blocking**. The reason to apply windowing function to **time segments** is to avoid the **discontinuities** caused by Frame blocking. **STFT** process can be depicted in Figure 1.6.

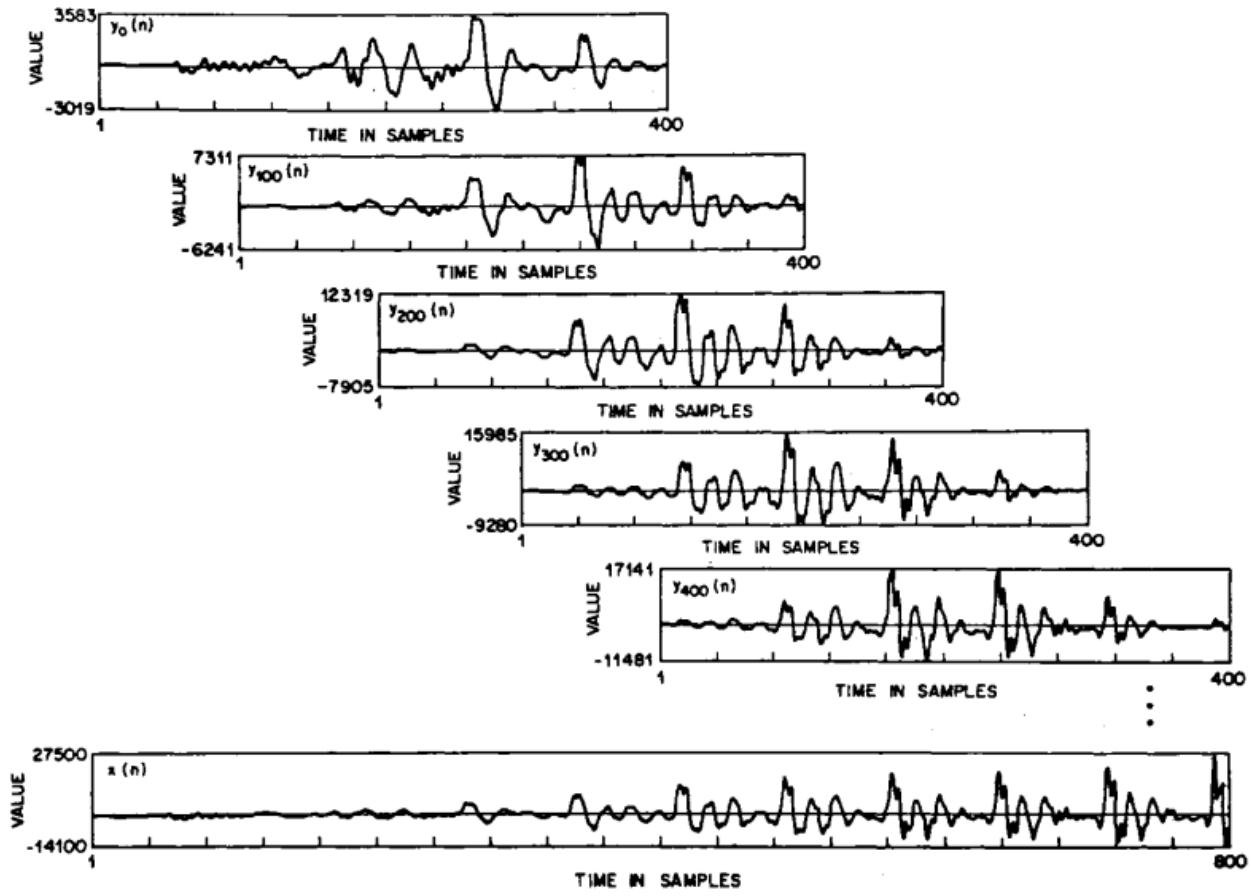


Figure 1.6: Graphical interpretation of the overlap-add synthesis method showing the overlapping sections (weighted by a Hamming Window Figure 1.5) and the result summation [32].

The overall process of producing a spectrogram from an audio file can be illustrated in Figure 1.7.

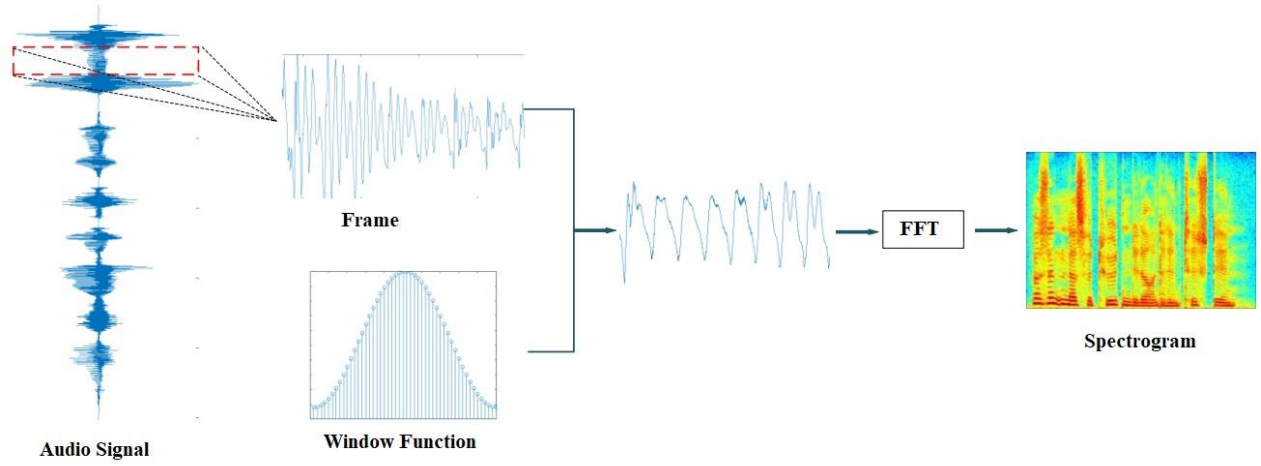


Figure 1.7: Spectrogram making process [33].

Two of the most used spectrograms in audio tagging are **Mel spectrogram** and **Log-Mel spectrogram**, which will be explained in the next subsections.

1.4.1. Mel Spectrograms

A spectrogram is a **visual** depiction of an audio signal in the **Time-Frequency** region [34]. The **Mel scale** is based on a unit of **PITCH** proposed by Stevens, Volkmann and Newmann in 1937. The Mel scale provides a **linear scale** below 1000 Hz and **logarithmic scale** beyond that for the human auditory system [35], and is related to Hertz by the following formula, where m represents Mels and f represents Hertz [34]:

$$m = 2595 log_{10}(1 + \frac{f}{700\ Hz}) \tag{1.5}$$

The Mel spectrogram is used to provide us with sound information similar to what a human would perceive [34]. Figure 1.8 shows what a Mel Spectrogram looks like where the color represents the power spectrum.

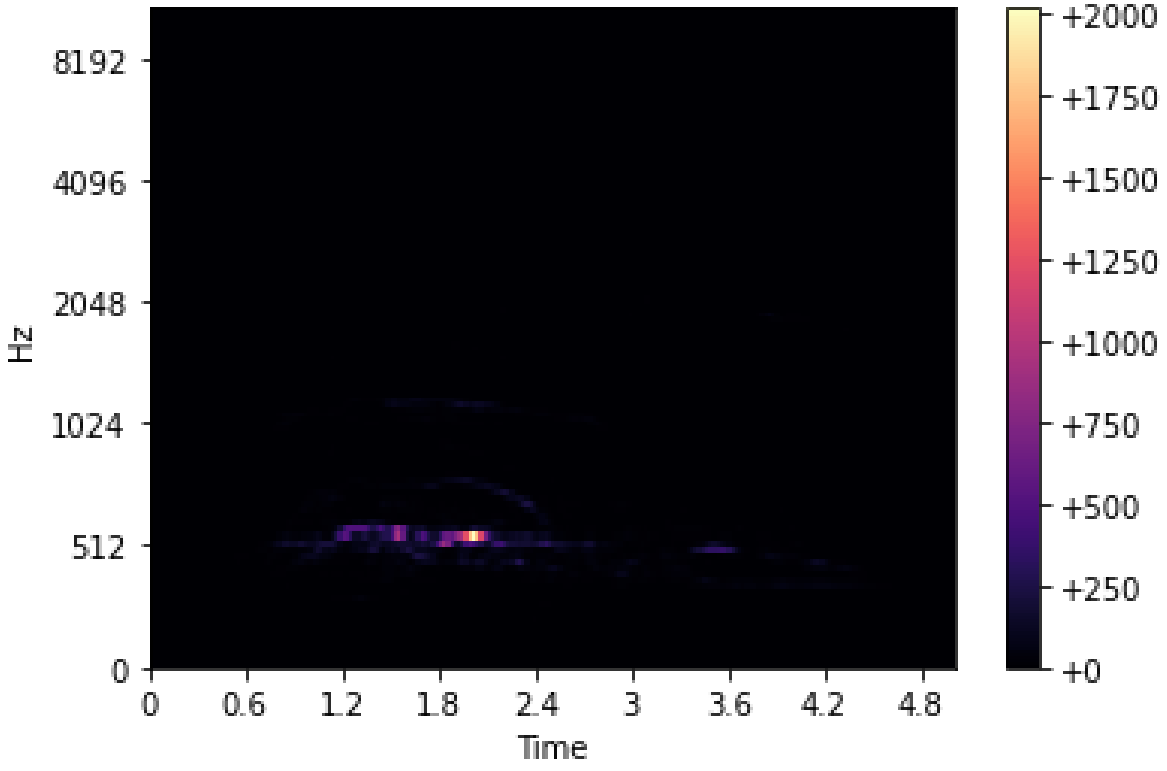


Figure 1.8: Mel Spectrogram of a "Applause, Crowd, Cheering" audio file.

1.4.2. Log-Mel Spectrograms

The Log-Mel Spectrogram is another visual representation of Mel Spectrograms where it is **converted** from **power** to **decibels**, which are log-scaled. The Figure 1.9 is the same as Figure 1.8 but the data is converted from power to decibels which results in different visuals. The massive experiments show that the Log-Mel Spectrogram is not only visually different from the Mel Spectrogram, but is a more suitable approximation of the human's auditory system [3][36].

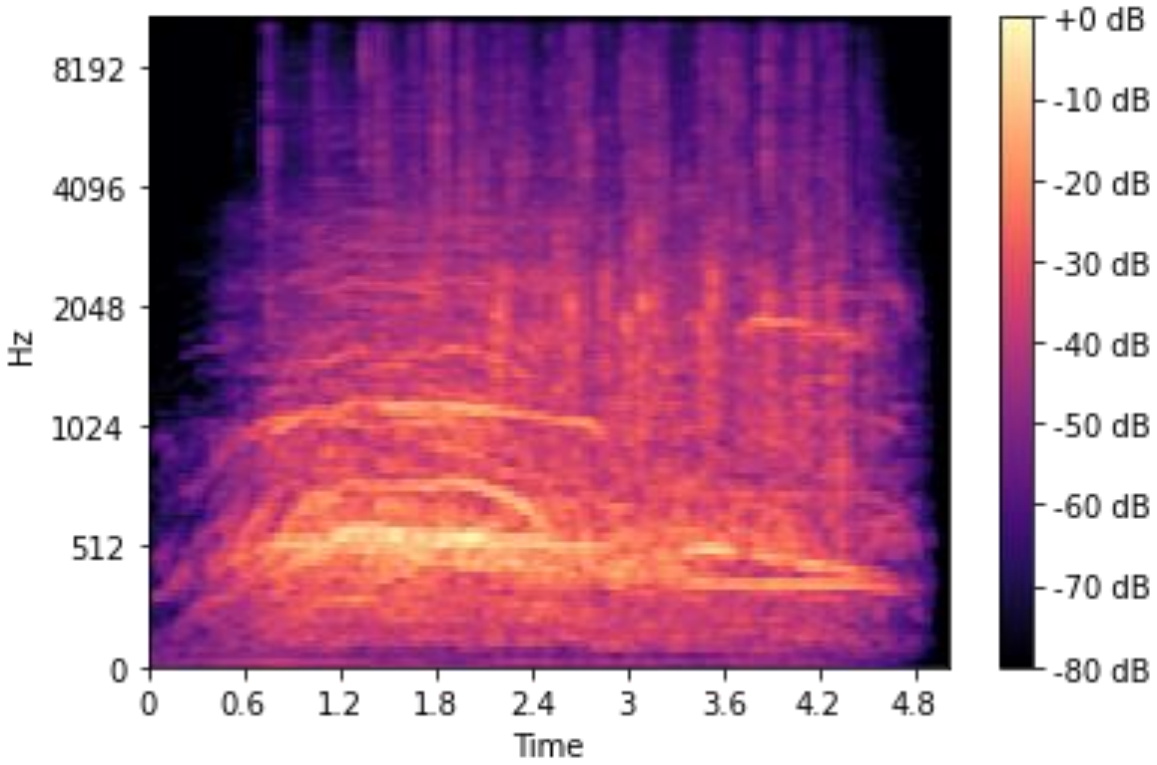


Figure 1.9: Log-Mel Spectrogram of a "Applause, Crowd, Cheering" audio file.

To date, Log-Mel Spectrograms are considered as one of the best variants of the visual features that could be used as an input feature to convolutional neural networks [37].

1.5. Fundamentals of Classification

Classification refers to a process where a class label is predicted for a given example of input data [24]. Through the use of many training examples, the classification model learns and calculates how to best map the examples to a specific class label or set of labels. A segment of audio is classified into a single predefined class

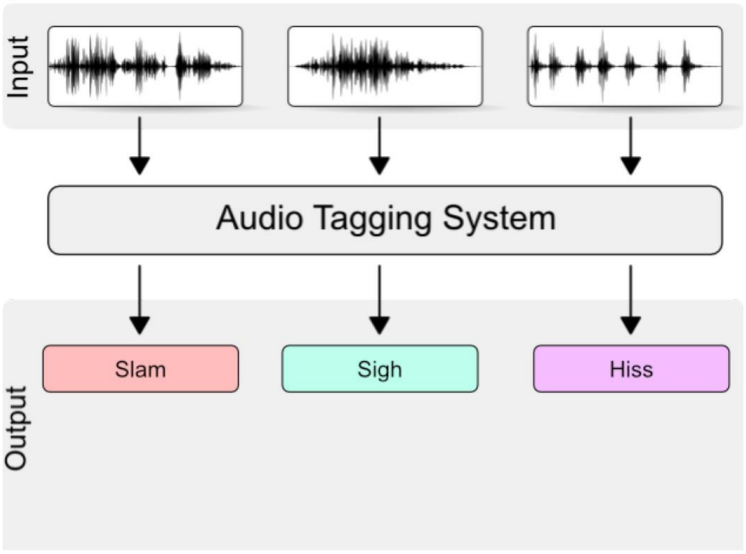
in single-label classification, or into multiple predefined classes in multi-label classification, the type of classification depends on the target application [20]. Figure 1.10 depicts both classifications methods.

1.5.1. Single-Label Classification

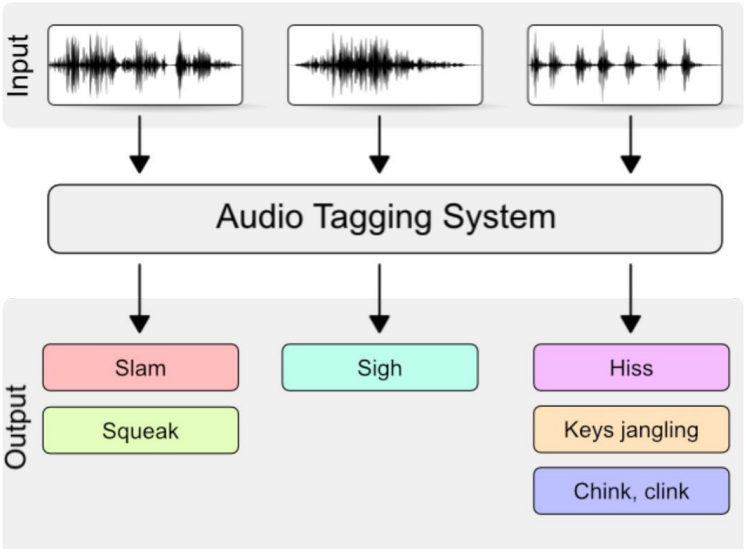
Commonly, a classification task involves predicting or associating a **single** label for each sample. More formally, given a set of $\boldsymbol{n}$ labels $L = \{l1, l2, \dots, ln\}$ and a set of $\boldsymbol{m}$ items $I = \{i1, i2, \dots, im\}$, in single-label classification the goal is to associate one label $\boldsymbol{l}$ to every item $\boldsymbol{i}$.

1.5.2. Multi-Label Classification

Multi-label classification refers to the tasks where each sample can be assigned simultaneously into **multiple** classes [38]. More formally, given a set of $\boldsymbol{n}$ labels $L = \{l1, l2, \dots, ln\}$ and a set of $\boldsymbol{m}$ items $I = \{i1, i2, \dots, im\}$, the multi-label classification task aims to associate a set $\boldsymbol{c}$ of $\boldsymbol{l}$ labels to every item in $\boldsymbol{i}$, where $\boldsymbol{c} \boldsymbol{\epsilon} [\boldsymbol{1}, \boldsymbol{n}]$ and varies for every item. In audio analysis systems, **audio tagging** systems are referred to as **multi-label** classification [20]. Moreover, in many real-world applications, multi-label classification is considered a more challenging task than single-label classification [39].



(a) Single-Label



(b) Multi-Label

Figure 1.10: Overview of (a) Single-Label and (b) Multi-Label audio tagging systems.

1.6. Performance Evaluation

The evaluation metric is a crucial element in achieving the optimal classifier during the training process. The metrics differs from task to task, where the type of classification (i.e. single-label, multi-label) makes a difference in the choice of an evaluation metric. For performance evaluation of an audio tagging system, the metrics have to be suited for **multi-label** classification. One of the metrics used for audio tagging

tasks is the **Multi-Label Ranking Metrics**, where the **order** of labels matters. Samples can have any number of ground truth labels associated with it, the goal of label ranking is to give better rank to labels according to their scores to focus on the first relevant labels, and avoids constructing binary classifiers that distinguish individual labels from the other labels. In simpler words, multi-label ranking aims to order all the relevant labels at a higher rank than the irrelevant ones.

1.6.1. Label Ranking Average Precision (LRAP)

A **label ranking average** precision score function is often used as an evaluation metric for audio tagging where it implements **ranking average** precision and is based on the notion of **label ranking** instead of **precision** and **recall**. Label ranking average precision (i.e. **LRAP**) averages over the samples, this performance measure will be higher if the system is able to give better rank to the labels associated with each sample. The obtained **score** is always strictly greater than **0**, and the best value is **1**. Formally, given a binary indicator matrix of the ground truth labels $y \epsilon \{0,1\}^{n_{sample} \times n_{labels}}$ and the score associated with each label $\hat{f} \epsilon \mathbb{R}^{n_{sample} \times n_{labels}}$, the average precision is defined as:

$$LRAP(y, \hat{f}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} \frac{1}{||y_i||} \sum_{j:y_{ij}=1} \frac{|L_{ij}|}{rank_{ij}} \tag{1.6}$$

where $L_{ij} = \{k: y_{ik} = 1, \hat{f}_{ik} \geq \hat{f}_{ij}\}, rank_{ij} = \{k: \hat{f}_{ik} \geq \hat{f}_{ij}\}$, $|.|$ computes the cardinality of the set (i.e. the number of elements in the set), and $||.||_0$ is the ℓ_0 "norm" (which computes the number of nonzero elements in a vector).

1.6.2. Label-Weighted Label-Ranking Average Precision (LWLRAP)

The **label-weighted label-ranking average** precision is abbreviated as **lwlrap** and pronounced "**lol wrap**" [27]. **lwlrap** measures the average precision of retrieving a ranked list of relevant labels for each audio clip (i.e., the system ranks all the available labels, then the precisions of the ranked lists down to each true label are averaged). This is a generalization of the mean reciprocal rank measure for the case where there can be multiple true labels per test item. The novel "**label-weighted**" part means that the overall score is the average over all the labels, where **each label** receives equal weight (by

contrast, plain **lrap** gives **each item** equal weight, thereby discounting the contribution of individual labels when they appear on the same item as multiple other labels). The **label weighting** is used because it allows per-class values to be calculated, and still have the overall metric be expressed as a simple average of the per-class metrics (weighted by each label's prior in the test set). Formally, let $\boldsymbol{Lab(s,r)}$ be the class label at **rank** $\boldsymbol{r}$ (starting from 1) in test **sample** $\boldsymbol{s}$, and $\boldsymbol{Rank(s,c)}$ be the rank of class label **c** in that list, i.e. $\boldsymbol{Lab(s,Rank(s,c)) = c}$. Then, if the set of ground-truth classes for sample $\boldsymbol{s}$ is $\boldsymbol{C(s)}$, the label-ranking precision for the list of labels up to class $\boldsymbol{c}$ (assumed to be $C(s)$ the number of true class labels for sample $\boldsymbol{s}$) is:

$$Prec(s,c) = \frac{1}{Rank(s,c)} \sum_{r=1}^{Rank(s,c)} I[Lab(s,r) \epsilon C(s)] \tag{1.7}$$

where $I[Lab(s,r) \epsilon C(s)]$ evaluates to I if the argument is true, else zero. $\boldsymbol{Prec(s,c)}$ is equal to 1 if all the **top-ranked** labels down to $\boldsymbol{c}$ are part of $\boldsymbol{C(s)}$, and at worst case equals $\frac{\boldsymbol{1}}{\boldsymbol{Rank(s,c)}}$ if none of the **higher-ranked** labels are correct. In contrast to plain **lrap**, which averages precisions within a sample then across samples, thereby **downweighting** labels that occur on samples with many labels, **lwlrap** calculates the precision for each label in the test set, and gives them all equal contribution to the final metric:

$$l\omega lrap = \frac{1}{\sum_s |C(s)|} \sum_s \sum_{c \epsilon C(s)} Prec(s,c) \tag{1.8}$$

where $|C(s)|$ is the number of true class labels for sample $\boldsymbol{s}$.

- **Calculate lwlrap for a label**

  In practice, the score of a label is calculated like this:

$$Score_{label} = \frac{number\ of\ correct\ answers\ from\ 1\ to\ predicted\ label\ rank}{predicted\ label\ rank}$$

  Assuming we labels A, B, C and D, if the correct answers are (A and C) and the predictions are (A: 0.4, B: 0.1, C:0.2, D:0.3) then it will be ranked like this (A:1,

B:4, C:3, D:2) (i.e. lower is better), the order in a descending will be (A, D, C, B). The calculation of the score of (A and C) will be like this:

For A:

A_{rank} = 1 so the number of correct answers from 1 to A_{rank} => 1 to 1 = 1 (i.e. Only A is correct and A is Ranked 1), so the score of $A = \frac{1}{1} = 1$.

For C:

C_{rank} = 3 so the number of correct answers from 1 to C_{rank} => 1 to 3 = 2 (i.e. Only A and C are correct and D have higher rank than C, so only 2 are correct answer out of the first 3 Ranked labels), so the score of $C = \frac{2}{3} = 0.66$.

- **Calculate lwlrap for all samples**

For a set of two samples if the following is assumed:

Sample 1: Correct answer label = A, C; prediction = (A: 0.1, B: 0.7, C: 0.2)

Sample 2: Correct answer label = B, C; prediction = (A: 0.1, B: 0.7, C: 0.2)

First, calculate the score for each class.

$$Score_{class} = \frac{total\ score\ for\ a\ class}{number\ of\ correct\ labels\ for\ a\ class}$$

Using $Score_{label}$ equation we find:

Sample 1 score = A: 0.6667, C: 0.5

Sample 2 score = B: 1.0, C: 1.0

So, the score of each class is as follow:

$$Score_A = \frac{0.6667}{1} = 0.6667$$

$$Score_B = \frac{1.0}{1} = 1.0$$

$$Score_C = \frac{(0.5 + 1.0)}{2} = 0.75$$

When calculating scores for all classes, averaging the scores of each class does not take into account the bias in the number of correct labels for each class. For **lrap**, frequent classes have less impact on the final score of one label, infrequently occurring classes have a greater effect on the final score of one label. Therefore, a weighted average is taken with the number of occurrences of each class as a weight (i.e. lwlrap) is calculated in the following way.

$$Weight_{class} = \frac{number\ of\ label\ occurrence}{total\ number\ of\ correct\ labels}$$

$$Weight_{class} = \frac{(A{:}\,1, B{:}\,1, C{:}\,2)}{4} = (A{:}\,0.25, B{:}\,0.25, C{:}\,0.5)$$

$$Score_{lwlrap} = A \times A_{weight} + B \times B_{weight} + C \times C_{weight}$$

$$Score_{lwlrap} = (0.6667 \times 0.25) + (1.0 \times 0.25) + (0.75 \times 0.5) = 0.7917$$

This is ultimately equal to the average of the scores for each label.

$$Average\ Score = \frac{correct\ labels\ scores}{total\ number\ of\ correct\ labels}$$

$$Average\ Score = \frac{A_{scores} + B_{scores} + C_{scores}}{4} = 0.7917$$

$$Average\ Score = \frac{0.6667 + 1.0 + (0.5 + 1.0)}{4} = 0.7917$$

- **lwlrap summary**:

-Firstly, the range of the probability score is between 0 and 1, where it is always greater than 0 and the higher being the better.

-Secondly, the score is calculated on the basis of the **relative ranking** of the label and not the actual probability scores, and a score of 1 means if there are ***k*** ground-truth labels for a given clip, then if the predicted probabilities are sorted in **descending** order (i.e. for each label of all labels), the first ***k*** labels are exactly the same labels that are present in the ground truth. The **non-ground-truth** labels can

decrease the score when they have a higher probability than the ground-truth labels (i.e. they are ranked higher than the ground truth labels).

-Finally, for **LWLRAP** specifically, the relative **occurrences** of the labels are taken into account and a weighted average is performed (**label-weighted**) instead of a simple average (**lrap**), which results in assigning appropriate weights based on the frequency of **occurrence** so that the less frequently appearing labels do not get an undue advantage.

1.6.3. Statistical tests

Given multiple learning algorithms, model evaluation aims at identifying which algorithm produces the most accurate classifiers. This concern is one among the fundamental issues in machine learning [39]. Various researchers adopt different statistical and common-sense techniques to decide whether the differences between the algorithms are real or random. In this regard, Demšar [40], García et al. [41], and Japkowicz et al. [42] introduced several statistical tests such as Friedman, Nemenyi, Bonferroni-Dunn and Wilcoxon for performance comparison.

- **Friedman test**

  The Friedman test is useful for comparing several algorithms over multiple domains. It first ranks the techniques for each dataset separately according to the generalization measure in descending order. The best performing technique gets the rank 1, the second best gets rank 2... etc. In case of ties, average ranks are assigned. Let r_i^j be the rank attributed to the j^{th} algorithm on the i^{th} dataset; and let R_j denote the average rank of algorithm $j \in \{1, \dots, t\}$ over N datasets. Under the null hypothesis, it is assumed that all techniques are equivalent; hence, their average ranks should be equal.

$$R_j = \frac{1}{N}\sum_{i=1}^{N} r_i^j \tag{1.9}$$

$$x_F^2 = \frac{12N}{t(t+1)}\left[\sum_{j=1}^{k} R_i^j - \frac{t(t+1)^2}{4}\right] \quad (1.10)$$

The test statistic is given in equations above chi-squared distribution with $t - 1$ degrees of freedom for sufficiently large N and t (Usually $N > 10$ and $t > 5$). This test provides only an assessment whether the observed differences in the performances are statistically significant.

- **Nemenyi test**

  This test is invoked when all techniques are compared with each other. The performance of two methods is significantly different if their corresponding average ranks differ by at least the critical difference CD.

$$CD = q_\alpha \sqrt{\frac{t(t+1)}{6N}} \quad (1.11)$$

Where the critical value q_α is defined based on the Studentized range statistic divided by $\sqrt{2}$.

- **Bonferroni-Dunn test**

  In general, the Bonferroni-Dunn test is undesirably conservative and has little power; nevertheless, this test is useful when the main interest is the comparison of all techniques against a control algorithm. In this specific case, Bonferroni-Dunn test is more powerful than Nemenyi test because this latter adjusts the critical value for making $t(t-1)$ comparisons, whereas when comparing with a control method, only $t - 1$ comparisons are made. This test is basically defined similarly to Nemenyi test except that we estimate the critical value for $\frac{\alpha}{(t-1)}$ significance level.

- **Wilcoxon signed-ranks test**

  Wilcoxon signed-ranks test is a non-parametric test and is considered the best strategy to compare two algorithms over multiple domains [43]. The formulation of this test is the following. We designate by di the difference between the performance scores of two techniques on N datasets. $i \in \{1, \dots, N\}$. We first rank these differences according to their absolute values; in case of ties, average ranks

are attributed. Then, we compute the sum of ranks for the positive and the negative differences, which are denoted as R^+ and R^-, respectively. Their formal definitions are given by:

$$R^+ = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{1.12}$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{1.13}$$

Notice that the ranks of $di = 0$ are split evenly between R^+ and R^+. Finally, the statistics T_w is computed as $T_w = \min(R^+, R^-)$. For small N, the critical value for Tw can be found in any textbook on general statistics [41], whereas for larger N, the statistics:

$$z = \frac{T_w - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{1.14}$$

1.7. Scientific and Technical Challenges in Audio Tagging

Many audio tagging systems achieve relatively high accuracies when using data produced in **controlled** laboratory conditions [20].

However, sounds that are produced in **realistic environments** tend to create **challenges** for the tagging systems which result in **poor** performance. These challenges can vary and depend on many factors.

Firstly, sounds come in many different types and the characteristics of the class that the sound falls under can be highly diverse, some of those characteristics can be very similar to other classes which can make recognizing the class of sound accurately more difficult.

Secondly, an audio signal captured by a **microphone** is affected by the channel coupling (impulse response) between the source and microphone, which may alter the signal sufficiently to prevent the matching of models developed to recognize the sound

[20]. The microphones that are used to capture audio are often significantly **further** away from target sources, which increases the effect of impulse responses from the source to the microphone as well as other sources in the environment.

Thirdly, in realistic environments there are almost always **multiple** sources producing sound simultaneously. The captured audio is a superposition of all the sources present, which again **distorts** the signal captured [20].

Finally, the most common challenge is that it may become difficult to collect enough samples for audio tagging systems due to different tasks requires different task-specific datasets, hence the number of recordings available may be limited. Also, the datasets have to be hand-labeled either by a machine learning engineer or a data scientist, this is a very costly and time-consuming process that involves a lot of manual labor [7], especially when dealing with large volumes of data. To counter these disadvantages, the concept of Semi-Supervised Learning was introduced.

1.8. Conclusion

In this Chapter, we have reviewed an outline of the basic concepts of audio tagging that are essential to understand the ideas treated in this work. We have provided some important concepts of classification in general, and audio feature extraction methods. In Addition, to performance evaluation metrics used in audio tagging and some common challenges of sound recognition research field. In the next chapter, we will present the fundamental notion of learning from weakly labeled data, including several well-known semi-supervised techniques.

Chapter 2 : Learning from Weakly Labeled Data

2.1. Introduction

Traditionally, machine learning has been studied in two fundamentally different types of tasks, either in the supervised learning paradigm where all the data is labeled, or in the unsupervised learning paradigm where all the data is unlabeled [15]. The goal of semi-supervised learning is to understand how combining labeled and unlabeled data may change the learning behavior, and design algorithms that take advantage of such combinations. The success of semi-supervised learning depends critically on some underlying assumptions [44].

In this chapter, we introduce the **semi-supervised** paradigm which is halfway between supervised and unsupervised learning paradigms, and the techniques used within and with the semi-supervised learning paradigm to deal with the lack of training data or the weakly labeled data.

2.2. Motivation

Current machine learning techniques require large and varied datasets in order to provide good performance and generalization [20]. On one side, supervised learning is the more commonly used form of machine learning, and it has proven to be a reliable solution for most problems by using large datasets to produce good results. However, manually labeling a dataset is expensive and time-consuming, which limits its size [20]. On the opposite side, unsupervised learning does not require labeled data to work, which can learn from the unlabeled data with a minimum human interaction; however, it is best used when there is no prior knowledge of what the output values for the samples should be, so its goal is to lean the inherent structure present within a set of data points without using explicitly-provided labels, but this approach does not really suit the audio tagging task where the output values are known. Both supervised and unsupervised learning comes with their **drawbacks** in the audio tagging task from lack of data, complexity and intensive human interaction. As a result, an in-between solution exists, **semi-supervised** learning uses both labeled and unlabeled data (i.e. weakly labeled data) to create an

audio tagging system. Many websites host large volumes of user-contributed audio and metadata, and labels can be inferred **automatically** from the **metadata** and/or predicted with **pre-trained** models. Nevertheless, these automatically inferred labels might include a substantial level of label **noise**, making them unreliable for the supervised usage. The main research question in learning from weakly labeled data is how to adequately exploit a small amount of reliable manually-labeled data, and a larger quantity of noisy web audio data (i.e. weakly labeled) in a multi-label audio tagging task.

2.3. Semi-Supervised Learning Assumptions

Semi-supervised learning methods have to make strong assumptions about the nature of the training data and thus, the performance of the predictor is highly dependent on these assumptions [44]. A Semi-Supervised algorithm assumes the following about the data:

- **Smoothness assumption**

  If two points $x1$ and $x2$ in a **high-density** region are close (e.g., if they belong to the same cluster and are close), then so should be their corresponding label sets $y1$, $y2$. This assumption is the main assumption made by semi-supervised learning algorithms [15], which implies that if two points are linked by a path of high-density then their outputs are likely to be close. If, on the other hand, they are separated by a **low-density** region then their outputs do not need to be close. This assumption applies to both regression and classification.

- **Cluster Assumption**

  If points are in the same **cluster**, they are likely to be of the same class. This assumption does not imply that each class forms a single compact cluster: it only means that usually, objects of two distinct classes are not observed in the same cluster [15]. The cluster assumption can easily be seen as a special case of smoothness assumption, considering that clusters are frequently defined as being sets of points that can be connected by short curves which traverse only high-density regions.

- **Low-Density Assumption**

  The **low-density** assumption implies that the **decision boundary** of a classifier should preferably pass through low-density regions in the input space. In other words, the decision boundary should not pass through high-density regions which is illustrated in Figure 2.1. For example, when considering a limited set of samples from the true distribution of the input data, it means that the decision boundary should lie in an area where few data points are observed. For the underlying data distribution, the low-density assumption is closely related to the smoothness assumption where it can be considered the counterpart of the smoothness assumption. In that point, supposing that a low-density area exists, if the decision boundary is placed in this low-density area, it will only concern pairs of similar data which does not violate the smoothness assumption. On the other hand, placing the decision boundary in a high-density area would indicate that the predicted labels are dissimilar for similar data points which does violate the smoothness assumption.

- **Manifold Assumption**

  In machine learning problems where the data can be represented in Euclidean space (i.e. 2- or 3-dimensional space), the observed data points in the **high-dimensional** input space $\mathbb{R}^d$ are usually concentrated along **lower-dimensional** substructures. These substructures are known as **manifolds**: topological spaces that are locally Euclidean. The manifold assumption in semi-supervised learning states that the input space is composed of **multiple lower-dimensional** manifolds on which all data points lie and data points lying on the same manifold have the same label. For instance, for a 3-dimensional input space where all points lie on the surface of a sphere, the data can be said to lie on a 2-dimensional manifold, which means that the data lie approximately on a manifold of much lower dimension than the input space (e.g. from 3-dimensional to 2-dimensional). Consequently, if it is possible to determine which manifolds exist and which data points lie on which manifold, the class assignments of unlabeled data points can be inferred from the labelled data points on the same manifold. This assumption is

different from the other assumption but it forms the basis of several semi-supervised learning methods [15].

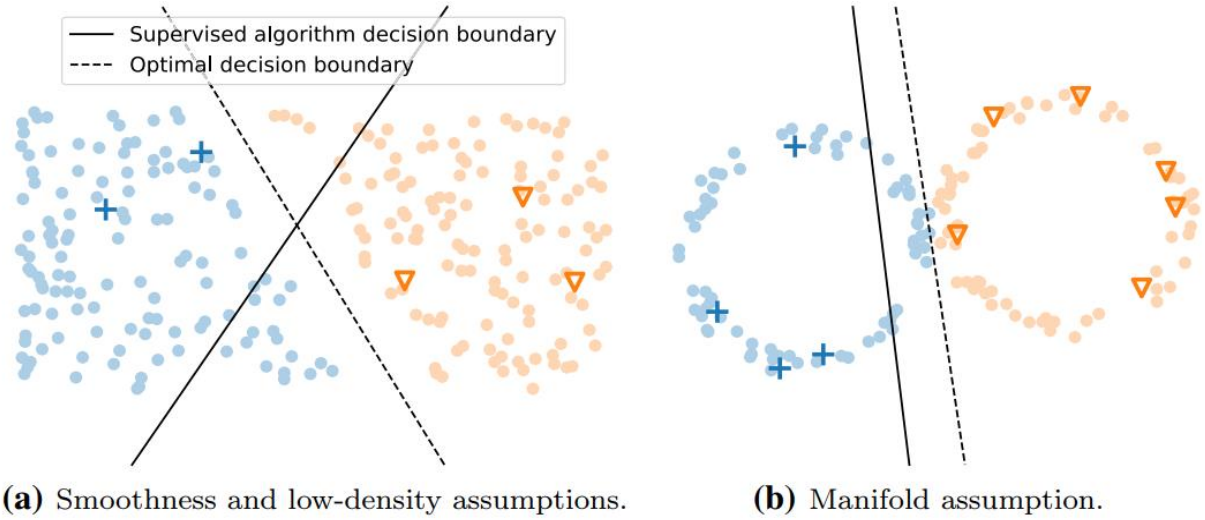


Figure 2.1: Illustrations of (a) Smoothness and low-density assumptions (b) Manifold assumption, and Cluster assumption depicted as the colors [45].

The figure above, illustrates all four assumptions. The cluster assumption is represented by the different colors; each cluster has a different color where the dots with the same color e.g. blue, belong to the same cluster. In (a) and (b), a reasonable supervised decision boundary is depicted, as well as the optimal decision boundary, which could be closely approximated by a semi-supervised learning algorithm relying on the respective assumption [45].

2.4. Semi-Supervised Learning Approaches

In Machine Learning, the semi-supervised learning paradigm is of great interest because it provides many techniques and approaches to handle the lack of labeled data. Some of the techniques incorporate ideas and components from some dominant augmentation and regularization concepts from well-known methods like Consistency Regularization techniques, Entropy Minimization and other traditional regularization techniques [16][46][47][48]. This section summarizes some of the well-known semi-supervised techniques.

2.4.1. Pseudo Labeling

The pseudo-labeling method generally follows some basic steps as demonstrated in Figure 2.2, it starts by training the model on a batch of labeled data following a supervised fashion, then uses it to **predict** labels on a batch of unlabeled data. The model gets **retrained** using the predicted labels (i.e. guessed labels) where it gets used for calculating the loss on unlabeled data, which get combined together with labeled loss (i.e. calculated loss on labeled data) and then **propagate** the combined loss. The combination of both losses differ from just simply adding the unlabeled loss with labeled loss when retraining the model with labeled and pseudo-labeled data, a weight applied to unlabeled loss is more effective [46]. Formally, the overall loss function is given by [46]:

$$L = \frac{1}{n}\sum_{m=1}^{n}\sum_{i=1}^{C} L(y_i^m, f_i^m) + \alpha(t)\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{C} L(y'^m_i, f'^m_i) \tag{2.1}$$

Where the notation is as follows:

-n is the number of batches in labeled data for Stochastic Gradient Descent.

-n’ is the number of batches for unlabeled data.

-C is the number of labels.

-f_i^m is the output units of m’s sample in labeled data.

-$y_i^m$is the label of f_i^m.

-$f_i'^m$is the output units of **m** sample in unlabeled data.

-$y_i'^m$is the pseudo-label of $f_i'^m$.

-$\alpha(t)$ is a coefficient “weight”.

In another simpler way, the loss function can be represented in this manner:

$$Loss = Labeled\ Loss + Weight \times Unlabeled\ Loss$$

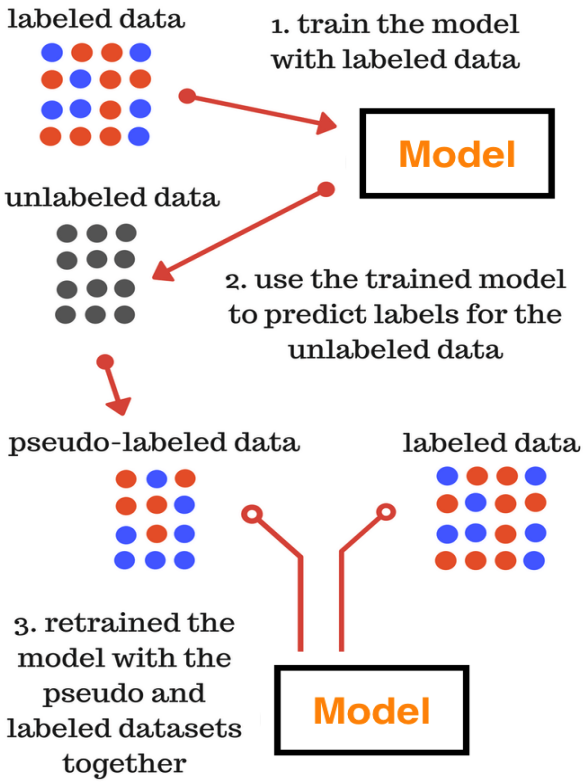


Figure 2.2: Pseudo-labeling steps.

2.4.2. MixMatch

In MixMatch algorithm, given a batch $\boldsymbol{X}$ of labeled examples with one-hot targets (representing one of L possible labels) and an equally-sized batch $\boldsymbol{U}$ of unlabeled examples, MixMatch produces a processed batch of augmented labeled examples $\boldsymbol{X}'$ and a batch of augmented unlabeled examples with "guessed" labels $\boldsymbol{U}'$. $\boldsymbol{X}'$ and $\boldsymbol{U}'$ are then used in computing separate labeled and unlabeled loss terms as illustrated in Figure 2.4. More formally, the combined loss $\mathcal{L}$ for semi-supervised learning is defined as [16]:

$$\mathcal{X}', \mathcal{U}' = MixMatch(\mathcal{X}, \mathcal{U}, T, K, \alpha) \tag{2.2}$$

$$\mathcal{L}_X = \frac{1}{|X'|} \sum_{x,p \epsilon X'} H(p, p_{model}(y \mid x; \theta)) \tag{2.3}$$

$$\mathcal{L}_{\mathcal{U}} = \frac{1}{\boldsymbol{\mathcal{L}}|\mathcal{U}'|} \sum_{u,q \epsilon \mathcal{U}'} \left|\left|q - p_{model}(y \mid u; \theta)\right|\right|_2^2 \tag{2.4}$$

$$\mathcal{L} = \mathcal{L}_X + \lambda_{\mathcal{U}} \boldsymbol{\mathcal{L}_{\mathcal{U}}} \tag{2.5}$$

where:

-$\boldsymbol{\mathcal{X}}$ a batch of labeled examples and their labels.

-$\boldsymbol{\mathcal{X}}'$ a batch of processed labeled examples produced by MixMatch.

-$\boldsymbol{\mathcal{U}}$ a batch of unlabeled examples.

-$\boldsymbol{\mathcal{U}}'$ a batch of processed unlabeled examples with their label guesses produced by MixMatch.

-$H(p, q)$ is the Cross-entropy between "target" distribution p and "predicted" distribution q.

-T is the sharpening temperature.

-K is the number of unlabeled augmentations.

-α is the beta distribution parameter for MixUp.

-p is a (one-hot) label.

-$p_{model}(y \mid u; \theta)$ refers to a generic model (i.e. model that represents all models) that produces a distribution over class labels y for an input x with parameters θ.

-x is a labeled example, used as input to a model.

-u is an unlabeled example, used as input to a model.

-θ model's parameters.

-λ_U is a hyper-parameter weighting the contribution of the unlabeled examples to the training loss.

- **Data Augmentation**

The data augmentation is used on both labeled and unlabeled data. For each x_b in the batch of labeled data $\mathcal{X}$, where a transformed version $\hat{x}_b = Augment(x_b)$ is generated (Figure 2.5, line 3). For each u_b in the batch of unlabeled data $\boldsymbol{\mathcal{U}}$, K augmentations are generated $\hat{u}_{b,k} = Augment(u_b),\ k \in (1, \dots, K)$ (Figure 2.5, line 5).

- **Label Guessing**

For each unlabeled example in $\boldsymbol{\mathcal{U}}$, MixMatch produces a "guess" for the example's label using the predictions produces by model trained fully using supervised learning (i.e. trained on $\boldsymbol{\mathcal{X}}'$). This guess is later used in the unsupervised loss term. To do so, MixMatch compute the average of the model's

predicted class distribution across all the **K** augmentations of u_b which is shown in Figure 2.5, line 7 by:

$$\bar{q}_b = \frac{1}{K}\sum_{k=1}^{K} P_{model}(y \mid \bar{u}_{b,k}; \theta) \tag{2.6}$$

After that comes one additional step, **Sharpening**. This step is inspired by the success of **entropy minimization** in semi-supervised learning. The sharpening function is applied to the average prediction over augmentation $\underline{q_b}$ to reduce the entropy (uncertainty) of the label distribution. In practice, for the sharpening function, a common approach of adjusting the "**temperature**" of this categorical distribution [49] is defined as the operation:

$$Sharpen(p, T)_i := p_i^{\frac{1}{T}} / \sum_{j=1}^{L} p_j^{\frac{1}{T}} \tag{2.7}$$

Where p is the average class prediction over augmentations $\bar{q}_b$. T is the sharpening temperature. Applying the sharpening function on the target distribution for unlabeled data achieves entropy minimization, the effect of the value of sharpening T is illustrated in Figure 2.3.

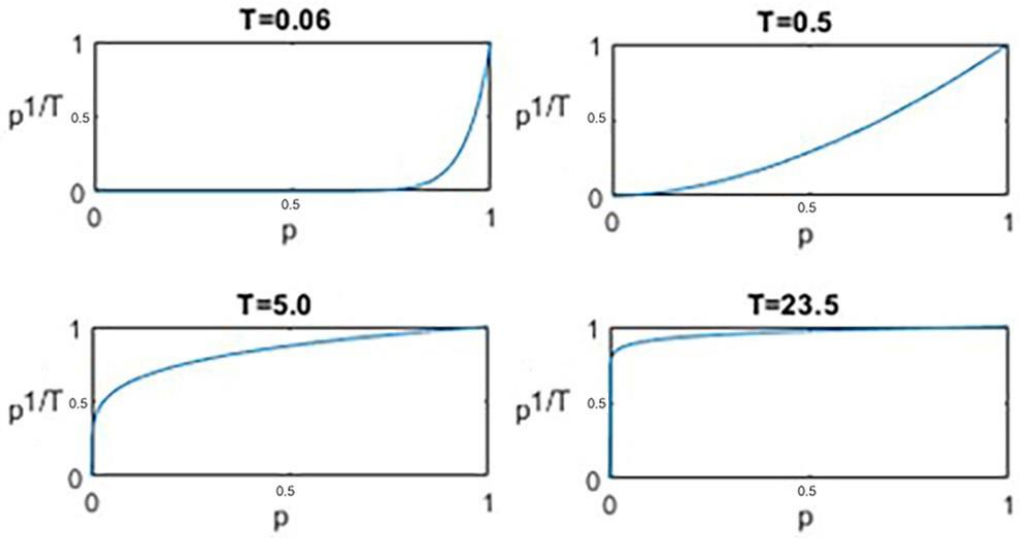


Figure 2.3: Plots of effect of different values of sharpening.

As shown in the figure above, lowering the value of T encourages the model to produce lower-entropy predictions which means the high probabilities are promoted. On one hand, as T get close to 0, the output of the sharpening function will approach one-hot distribution. On the other hand, increasing the value of T will only result in giving more importance to low probabilities.

- **Mix up**

MixUp is a data **augmentation** technique that **mixes** pairs of samples. If x_1 and x_2 are two different input samples and y_1, y_2 their respective labels, then the mixed sample and target are obtained by a simple convex combination:

$$x^{mix} = \lambda x_1 + (1 - \lambda)x_2 \tag{2.8}$$

$$y^{mix} = \lambda y_1 + (1 - \lambda)y_2 \tag{2.9}$$

where $\lambda \sim Beta(\alpha, \alpha) \in [0, 1]$ (**beta distribution** is a family of **continuous** probability distributions defined on the interval $[0, 1]$), the hyper-parameter **α** controls the strength of interpolation between feature-target pairs. **MixMatch uses a slightly modified version of MixUp** to encourage convex behavior "between" examples. MixUp is utilized both as a regularizer (applied to labeled data-points) and a semi-supervised learning method (applied to unlabeled data-points). For a pair of two examples with their corresponding labels probabilities $(x1, p1)$, $(x2, p2)$ it computes (x', p') by:

$$\lambda \sim Beta(\alpha, \alpha) \tag{2.10}$$

$$\lambda' = max(\lambda, 1 - \lambda) \tag{2.11}$$

$$x' = \lambda' x_1 + (1 - \lambda')x_2 \tag{2.12}$$

$$p' = \lambda' p_1 + (1 - \lambda')p_2 \tag{2.13}$$

Vanilla MixUp omits the equation (2.11) (i.e. it sets $\lambda' = \lambda$). In MixMatch both labeled and unlabeled examples are **concatenated** in the same batch, so there is a

need to **preserve** the **order** of the batch to compute individual loss components appropriately, this is achieved by the same omitted equation which ensures that x' is closer to x_1 than to x_2. To Apply MixUp, all augmented labeled examples with their labels and all unlabeled examples with their guessed labels are **collected** into (Figure 2.5 lines 10-11):

$$\hat{\chi} = ((\hat{x}_b, p_b); b \in (1, \ldots, B)) \tag{2.14}$$

$$\hat{u} = ((\hat{u}_{b,k}, q_b); b \in (1, \ldots, B), k \in (1, \ldots, K)) \tag{2.15}$$

Then, these collections are combined and the result gets **shuffled** to form $\mathcal{W}$ which will serve as a **data source** for MixUp (Figure 2.5 line 12). For each i^{th} example-label pair in $\hat{x}$, MixMatch compute $MixUp(\hat{x}_i, \mathcal{W}_i)$ and add the result to the collection $\mathcal{X}'$ (Figure 2.5 line 13), then compute $\mathcal{U}'_i = MixUp(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\chi}|})$ for $i \in (1, \ldots, |\hat{\mathcal{U}}|)$, intentionally using the remainder of $\mathcal{W}$ that was not used in the construction of $\mathcal{X}'$ (Figure 2.5 line 14).

- **Loss Function**

  For the loss function, MixMatch uses the standard semi-supervised loss shown in equations (3) to (5) [16]. Equation (5) combines the typical **cross-entropy** loss between labels and model predictions from $\mathcal{X}'$ with squared L_2 loss on predictions and guessed labels from $\mathcal{U}'$. Then, L_2 loss is used in equation (4) (the multiclass Brier score [50]) because unlike the cross-entropy, it is bounded and less sensitive to incorrect predictions [16]. For this reason, it is often used as the unlabeled data loss in semi-supervised learning [47][51] as well as a measure of predictive uncertainty [52]. MixMatch does not propagate gradients through computing the guessed labels, as is standard [47][51][53][54].

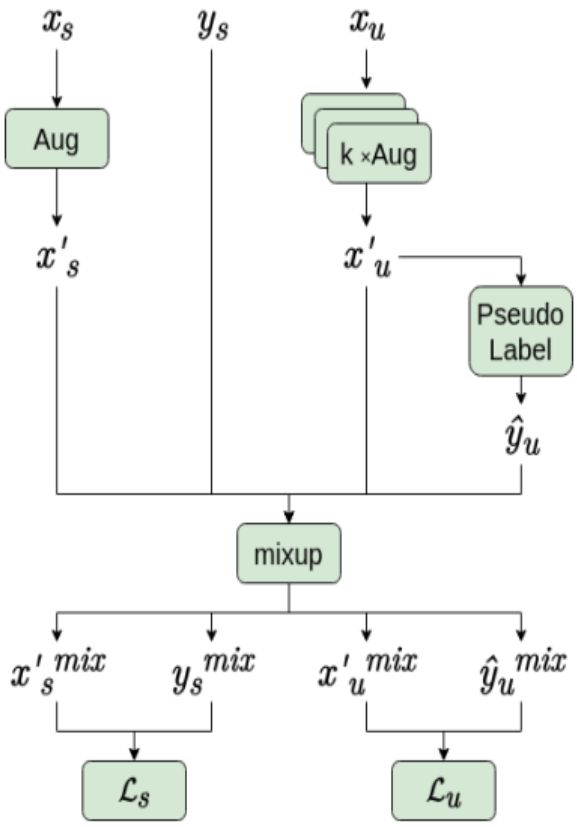


Figure 2.4: MixMatch workflow [55].

Algorithm 1 MixMatch takes a batch of labeled data $\mathcal{X}$ and a batch of unlabeled data $\mathcal{U}$ and produces a collection $\mathcal{X}'$ (resp. $\mathcal{U}'$) of processed labeled examples (esp. unlabeled with guessed labels).

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = ((x_b, p_b); \text{b} \in (1, \dots, \text{B}))$, batch of unlabeled examples $\mathcal{U} = (u_b; \text{b} \in (1, \dots, \text{B}))$, sharpening temperature T, number of augmentations K, Beta distribution parameter α for MixUp.
for b = 1 **to** B **do**
  $\hat{x} = Augment(x_b)$ // Apply data augmentation to x_b
  for k = 1 **to** K **do**
    $\hat{u}_{b,k} = Augment(u_b)$ // Apply k[th] round of data augmentation to u_b
  end for
  $\bar{q}_b = \frac{1}{K}\sum_{k=1}^{K} P_{model}(y \mid \bar{u}_{b,k}; \theta)$ // Compute average predictions across all augmentations of u_b
  $q_b = Sharpen(\bar{q}_b, T)$ // Apply temperature sharpening to the average prediction (see eq. (7))
end for
$\hat{\mathcal{X}} = ((\hat{x}_b, p_b); b \in (1, \dots, B))$ // Augmented labeled examples and their labels
$\hat{\mathcal{U}} = ((\hat{u}_{b,k}, q_b); b \in (1, \dots, B), k \in (1, \dots, K))$ // Augmented unlabeled examples, guessed labels
$\mathcal{W} = \text{Shuffle}(\text{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}}))$ // Combiner and shuffle labeled and unlabeled data
$\mathcal{X}' = (MixUp(\hat{\mathcal{X}}_i, \mathcal{W}_i); i \in (1, \dots, |\hat{\mathcal{X}}|))$ // Apply MixUp to labeled data and entries from $\mathcal{W}$
$\mathcal{U}' = (MixUp\left(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|}\right); i \in (1, \dots, |\hat{\mathcal{U}}|))$ // Apply data augmentation to x_b
return $\mathcal{X}'$, $\mathcal{U}'$

Figure 2.5: MixMatch Algorithm [16].

2.4.3. Mean Teacher

Mean Teacher is a method that propose averaging model **weights** instead of **predictions**, it uses two neural networks: a "**student**" f (a supervised architecture) and a "**teacher**" g, that share the same architecture (i.e. **teacher** is a copy of **student**). Figure 2.6 illustrates the mean teacher method. Both the student and the teacher model evaluate the input by applying random **noise** within their computation (i.e. η for student, η' for teacher). The softmax output of the student model is compared with the one-hot label using **classification cost** and with the teacher output using **consistency cost**, the consistency cost $\boldsymbol{J}$ is defined as the expected distance between the prediction of the student model (with weights θ and noise η) and the prediction of the teacher model (with weights θ' and noise η').

$$J(\theta) = \mathbb{E}_{x,\eta',\eta}\left[\left|\left|f(x,\theta',\eta') - f(x,\theta,\eta)\right|\right|^2\right] \tag{2.16}$$

where:

- $f(x,\theta,\eta)$ is the prediction of the student model
- $f(x,\theta',\eta')$ is the prediction of the teacher model

For the student model, the weights are updated using the standard **gradient descent** algorithm, whereas the weights of the teacher model are the **Exponential Moving Average** (EMA) of the **student** weights, where θ'_t (the weight of the teacher model) is defined at training step $\boldsymbol{t}$ as the EMA of successive θ (weight of student model).

$$\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\theta t \tag{2.17}$$

where α is a smoothing coefficient hyperparameter.

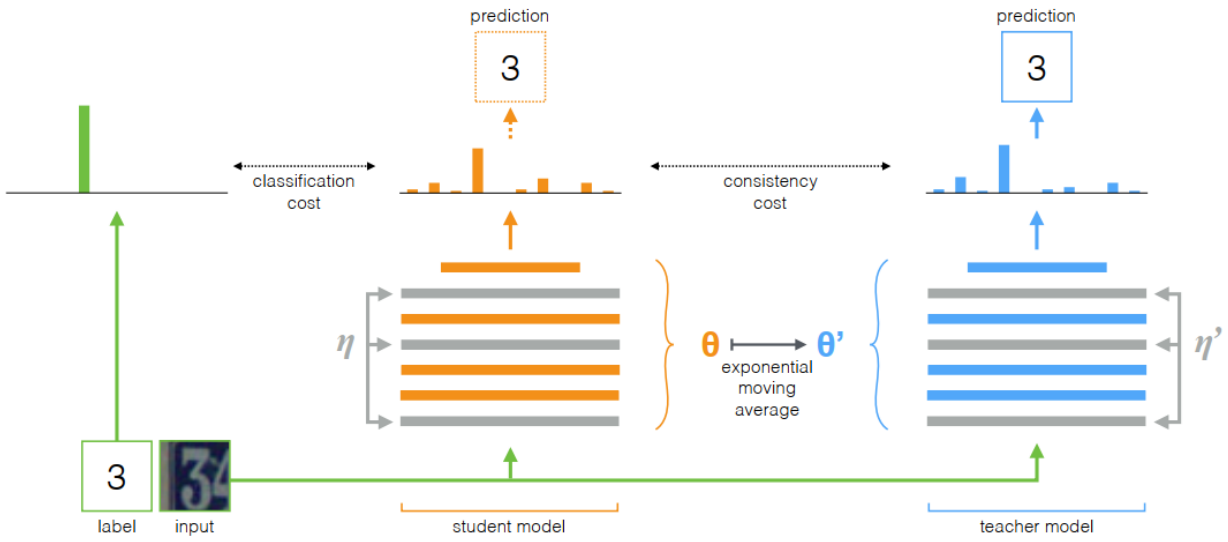


Figure 2.6: The Mean Teacher method [47].

The Moving average methods are used with time series data to **smooth** the random short-term variations and to highlight other components present in the data. The Exponential Moving average method (EMA) is used to **filter** out **noise**, the weight of each element decreases progressively over time, meaning the exponential moving average gives greater weight to **recent data** points (i.e. For EMA recent data is more relevant than old data), because of that EMA reacts faster to changes since it is more sensitive to recent movements.

Finally, both model outputs can be used for prediction, but at the end of the training, the **teacher** prediction is more likely to be correct. A training step with an unlabeled example would be similar, except no classification cost would be applied [47].

2.5. Multitask Learning

Multi-Task Learning (MTL) is the process of **sharing** representation between related tasks, to enable the system to generalize better on the original task [56]. Most multi-task methods focus on how to **combine** the **weights** of several neural networks [56][57][58]. By other means, **optimizing more than one loss** function that share some hidden layers and have different output layers is effectively doing multi-task learning (in contrast to single-task learning). For example, **combining two different losses** $L1$, $L2$

(that must share some hidden layers and have different output layers), is done by adding the loss together (i.e. $Loss = L1 + L2$), then **backpropagate** to fine-tune the weights is effectively doing multi-task learning (i.e. use different loss to fine-tune the shared hidden layers). The Multitask Learning in deep neural networks can be performed in two ways, either by **hard** parameter sharing or **soft** parameter sharing of hidden layers [56].

Hard parameter sharing

Hard parameter sharing is generally applied by **sharing** the hidden layers between all tasks while keeping several **task-specific** output layers as depicted by Figure 2.7.

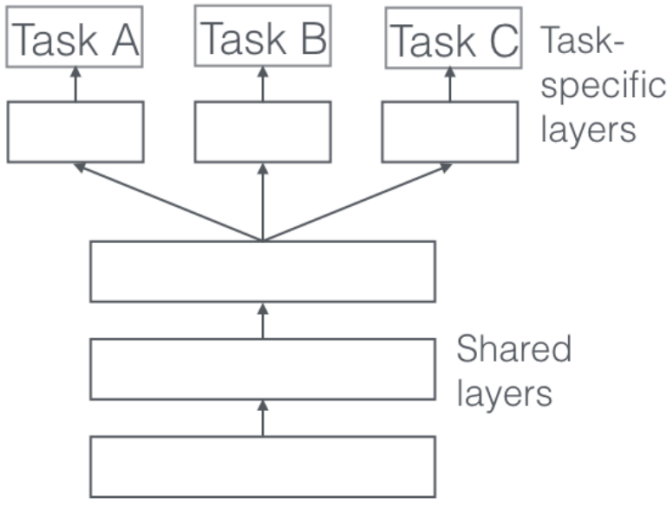


Figure 2.7: Hard parameter sharing for multi-task learning in deep neural networks [56].

Using hard parameter sharing **reduces** the risk of **overfitting** [59], the research showed that the more tasks get learned simultaneously, the more the model has to find a representation that captures all of the tasks and the less is the chance of overfitting the original task.

Soft parameter sharing

In Soft parameter sharing, each task has its own model with its **own parameters**. Also, to encourage the parameters to be similar, the **distance** between the parameters of the model is **constrained**/**regularized** which has been seen in [60][61], and as depicted in Figure 2.8.

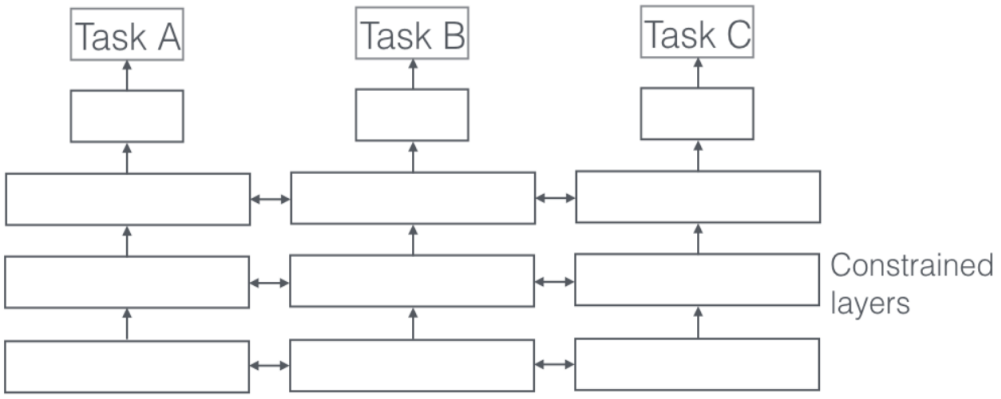


Figure 2.8: Soft parameter sharing for multi-task learning in deep neural networks [56].

The constraints/regularization used for soft parameter sharing in deep neural networks have been greatly inspired by regularization techniques that have been developed for other models.

Finally, in practice, different amounts of sharing tend to work best for different tasks [57], and tend to be used successfully across all applications of machine learning [56]. But, while the gain from multi-task learning is encouraging, getting the most out of it is still tiresome in practice [57].

2.6. Conclusion

In this chapter, we introduced some concepts to handle weakly labeled data or the lack of training data (i.e. labeled data), which mostly manifest in methods from the semi-supervised learning paradigms with a combination of other dominant methods. In the next chapter, we will define the setup that presents the characteristics of our main system alongside the training process.

Chapter 3 : Design of a Semi-Supervised Audio Tagging System

3.1. Introduction

In the previous chapters, we have presented the basic notions required for understanding the materials and components that will be covered in this chapter. We introduce our semi-supervised pipeline by presenting our system characterization, architecture, data preparations and the training steps. In Section 3.2, we explain the nature of inputs used to train the models. Section 3.3, we made an illustration of the pipeline of our system. In the following Section 3.4, we present the data preparation, Preprocessing steps, Feature Extraction parameters and data augmentation techniques. In Section 3.5, we provide an overview of Deep Neural Network Architecture, alongside the ResNet-34 architecture used in the training of our system. Finally, in Section 3.6, we describe the three main stages for training our system.

3.2. Nature of input data

Our audio tagging system requires two types of input: **manually-verified labeled** data denoted as $\Omega_{curated}$ and **non-manually-verified** data as Ω_{noisy} (i.e. weakly labeled). The manually-verified data $\Omega_{curated}$ is referred to as **Curated set** with **curated labels** $\Omega_{\boldsymbol{curated}} = \{(\boldsymbol{w_1}, \boldsymbol{s_1}), (\boldsymbol{w_2}, \boldsymbol{s_2}), \ldots, (\boldsymbol{w_m}, \boldsymbol{s_m})\}$, where $\boldsymbol{w_i}$ denotes the $\boldsymbol{i^{th}}$ audio file and $\boldsymbol{s_i}$ is the set of label classes $s_i \subseteq \{c_1, c_2, \ldots, c_n\}$, i $\subseteq \{1,2, \ldots, m\}$. The curated data is split into two sets: one for training $\Omega_{curated}^{train}$ and the other for validation $\Omega_{curated}^{test}$. Whereas, the weakly labeled data Ω_{noisy} is referred to as **Noisy set** with **noisy labels** $\Omega_{\boldsymbol{noisy}} = \{(\boldsymbol{w_1}, \boldsymbol{s_1}), (\boldsymbol{w_2}, \boldsymbol{s_2}), \ldots, (\boldsymbol{w_q}, \boldsymbol{s_q})\}$, where $\boldsymbol{w_i}$ denotes the $\boldsymbol{i^{th}}$ audio file and $\boldsymbol{s_i}$ is the set of label classes $s_i \subseteq \{c_1, c_2, \ldots, c_n\}$. The Noisy data is used only during the training phase. Note that the noisy labels have not been considered for the training of our system, but they have been used to train some models in the experiments (refer to **Chapter 4 Section 4.4** for more details about the use of noisy labels).

3.3. System Characterization

To better understand our main system characterization, we illustrate the overall pipeline of our system in Figure 3.1.

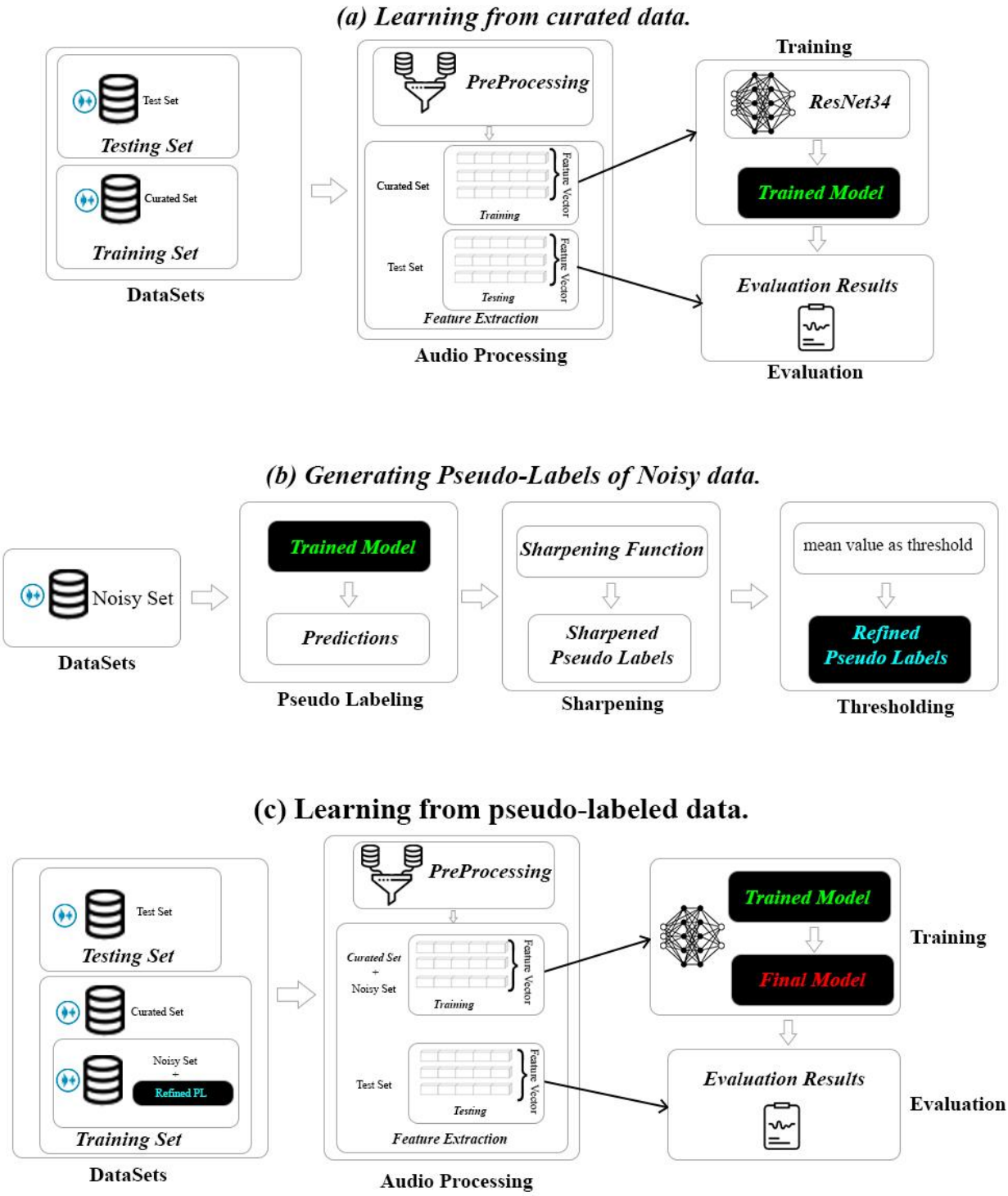


Figure 3.1: Overall pipeline of our system.

Our system operates in three main stages:

(a) Learning from curated data: we have used only the curated data in a supervised fashion to train a ResNet model.

(b) Generating pseudo-labels of noisy data: we have used the trained model from the previous stage to predict labels of the noisy data. Since these predictions have not been verified manually, we refer to them as Pseudo-Labels. Then, we have employed sharpening technique to enhance the quality of the pseudo-labels.

(c) Learning from the pseudo-labeled data: we have replaced the noisy labels with the pseudo-labels before combining the curated and noisy sets to retrain the trained model from (a) to produce the final model. Additional details on data preparation and Feature extraction are provided in the next sections.

3.4. Data Preparation and Feature Extraction

In practice, usually audio passes through Preprocessing steps that involves data preparation. One of the most common problems in data preparation is the imbalanced class distribution. The skewed class distribution may cause poor prediction performance which will require specialized training techniques to achieve good results. But, the dataset we used was already well balanced except in a few classes where there are less audio clips (refer to **Chapter 4 Section 4.2**). The next essential step is the Feature Extraction of Log-Mel spectrograms which will be explained below.

Log-Mel Spectrograms

For training our system, we have used only Log-Mel Spectrograms. We handled the problem of different audio time durations by slicing equal time lengths of each audio. Thereafter, it was normalized by the mean and standard deviation of each data. Log-Mel Spectrograms were extracted from the training and testing datasets using the parameters provided in Table 3.1. These values have shown promising results [8][13][14]. We assume that all audio clips come with a 44.1 kHz sampling rate (i.e. standard sample rate in consumer audio), according to Nyquist theorem, 44.1 kHz allows reproduction of all frequency content below 22.05 kHz which cover all frequencies perceived by human ear, as well as the minimum frequency of 20 Hz. All audio clips are converted to 128 mel

bands Log-Mel spectrogram with 2560 FFT and hop length of 347 samples between successive frames to achieve 128 Hz time resolution (i.e. 1 second represents 128 Hz).

Table 3.1: Log-Mel spectrogram parameters.

Parameter	Value
Sample rate	44.1 kHz
Mel bands	128
FFT	2560
Hop size	347
FrequencyMin	20 Hz
FrequencyMax	22.05 kHz

Data augmentations

To leverage more training data, we have applied numerous augmentation techniques to Log-Mel spectrogram:

- **MixUp**

  MixUp [10] is an augmentation that mixes two pairs of inputs and labels with some ratio (e.g. 70% first input and 30% second input). The mixing ratio is selected from the beta distribution where we used alpha of 1.0 which makes Beta distribution equal to uniform distribution.

- **SpecAugment**

  SpecAugment [9] is an augmentation method for spectrograms that consists of three kinds of deformation. We have used only frequency masking with a random width chosen from 8 to 32 from a uniform distribution. The other two deformations are time warping that deforms time-series in the time direction and time masking.

- **Slicing (cropping)**

  We have used the slicing as a way to adjust and fix the size of the input size and as a data augmentation by selecting every time a random section of each audio clip while preserving the same length across all audio files. The audio clips that have a shorter duration than the input size are extended using zero padding.

- **Gain augmentations**

  We used gain augmentation by changing the decibel ‘dB’ input of each audio with a randomly selected factor from a range of 0.8 to 1.2, that means we lower/increased audio loudness randomly.

3.5. Deep Neural Network Architecture

Deep Neural Network (DNN) architectures have gained a significant interest in the recent years; it has been successfully used to build many Audio tagging systems [8][13][14]. Most importantly, Deep Neural models have helped many researchers achieve better performance in audio tagging tasks compared to shallow acoustic models [62]. Most of today’s neural networks are organized into layers of nodes (i.e. neurons) which are known to be composed of one input layer, one hidden layer and one output layer as shown in Figure 3.2, and they are “feed-forward,” meaning that data moves through them in only one direction.

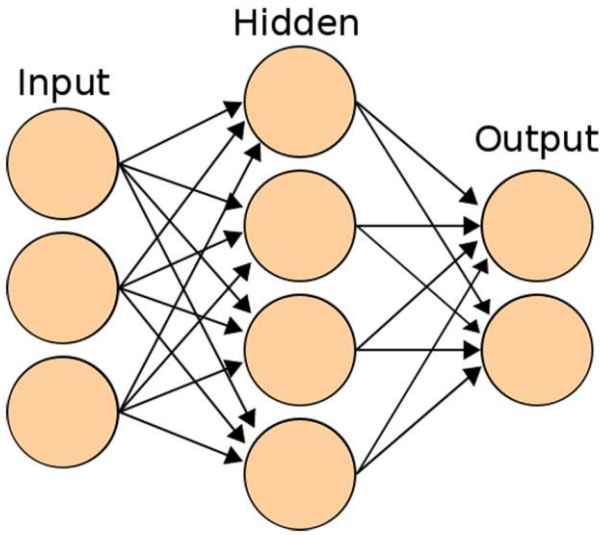


Figure 3.2: One Layer Neural Network.

A neural network consists of thousands or even millions of simple processing nodes that are densely interconnected through connectors called weighted connections. The nodes perform some operations on the inputs to reach the final output. The overall operations done by a node is illustrated in Figure 3.3 [63]. An individual node might be connected to several nodes in the layer before it, from which it receives data, and several nodes in

the layer after it, to which it sends data (i.e. the input layer is the bottom layer). When a neural network is being trained, each node is initialized with a weight and threshold set to random values, which will store and evaluate how significant one of the inputs is to the output. During training, the neural network starts by feeding the training data to the input layer. Then, it passes through the succeeding layers, getting multiplied and added together in complex ways after storing information regarding the input's importance, the information goes through an activation function that decide whether to pass the information to the next neuron; Then, it finally arrives radically transformed at the output layer. After finding patterns that correlate with the output layer (i.e. particular label), the weights and thresholds are repeatedly adjusted until training data with the same labels consistently yield similar outputs through the use of an **optimization** method. This latter estimates the error gradient for the current state of the neural network using examples from the training dataset, then it updates the weights using the backpropagation of losses.

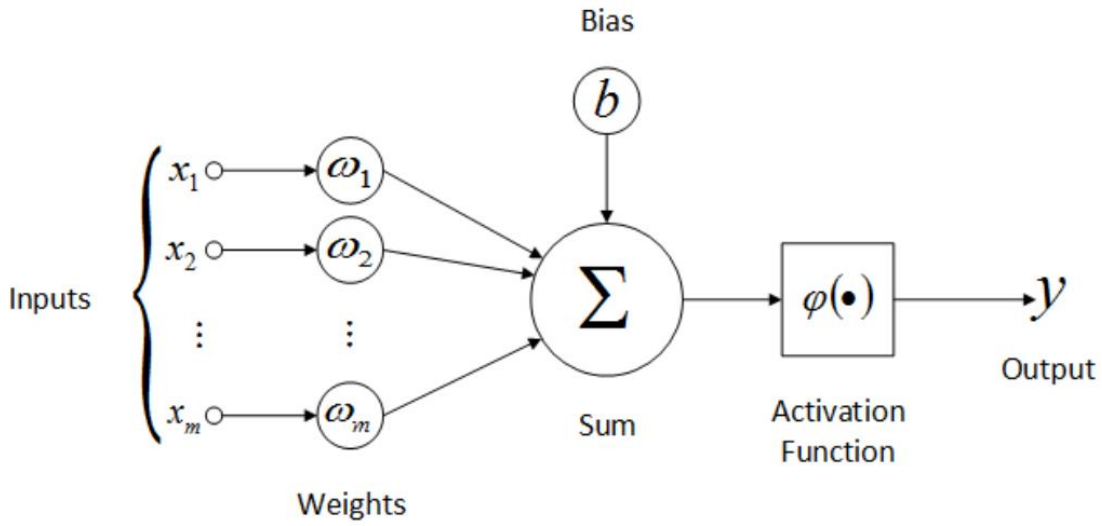


Figure 3.3: Operation done by a neuron [63].

A well-known type of neural networks is Deep Neural Networks, what qualifies a network as a deep learning network is the use of more than one hidden layer. Deep Neural Networks are successful and capable of achieving strong classification performance and can outperform other machine learning methods [64]. Deep Neural Networks have shown success in many research fields and one of the major fields is audio analysis. Many neural networks architectures have been proposed, for instance, AlexNet, VGG, ResNet and many more [65]. Each layer of the Deep Neural Network

learns distributed representations of features presented at the layer before it, where multiple hidden units collaborate together to explain various hidden causes of input data. Such a representation allows the model to focus more and more on aspects that are important for accurate Audio Tagging at deeper layers [66]. Also, the distributed nature of the model's representation helps generalize better to unseen situations (e.g. different audio environments, noise sources), even with limited amounts of training data. Deep Neural network architecture can solve complex problems by stacking additional hidden layers to improve accuracy and performance, but this achievement comes with its drawbacks like vanishing gradient problem which is caused only in deeper networks.

We have chosen Residual Neural Network (**ResNet-34).** The architecture is depicted in Figure 3.4 (i.e. the right architecture) as our architecture for a few reasons, like its success in many audio related research [18][19]. Another reason is the breakthrough that ResNet achieved in image processing with the residual blocks component that helps solve deep network **vanishing gradient** problem. The residual block and vanishing gradient will be explained in the next subsection.

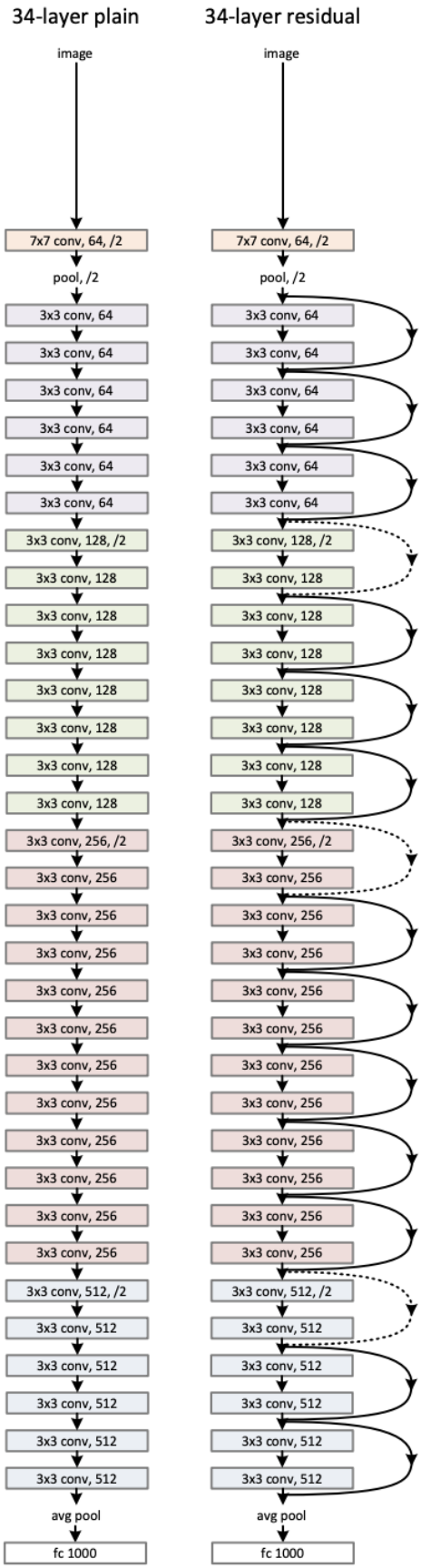


Figure 3.4: Left: a plain network with 34 parameter layers. Right: a residual network with 34 parameter layers. The dotted shortcuts increase dimensions [17].

- **Deep residual neural networks (ResNet)**

The ResNet architecture [17] is one of the most known architectures in deep learning networks. It is possible to train hundreds or even thousands of layers and still achieves great performance. The ResNet does have many variants (i.e. same concept but with different number of layers); few of the most known ResNet Architecture are ResNet-18, ResNet-34, ResNet-50 and ResNet-1202 [67]. The digits in the end of ResNet name simply implies the number of hidden layers. The main intent of ResNet is solving the **vanishing gradient** problem, which can only be observed in **deeper** networks. The vanishing gradient is encountered when training deep neural networks with **gradient-based** learning methods and **backpropagation**. The more layers get added to the network it makes the gradients of the loss function approaches zero (i.e. gradient too small for training to work effectively). The ResNet is a feedforward neural network made of **Residual Blocks** which are **skip-connection** blocks as shown in the following Figure 3.5.

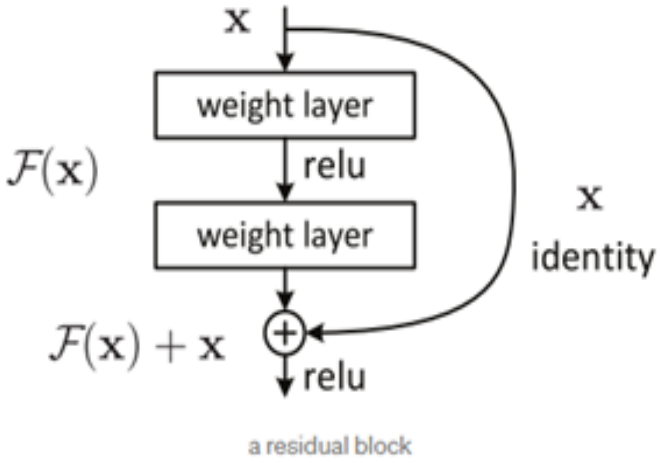


Figure 3.5: Residual learning a building block [17].

The core idea of residual blocks is called "**identity shortcut connection**" that skips one or more layers, is shown in Figure 3.5. The sub-blocks in the architecture represent the complete convolutional layers including the activation functions. A deep residual network can consist of multiple stacked building blocks as depicted previously in Figure 3.4. In one residual building block, the output $H(x)$ of the block is a mapping of the input x. Instead of letting the multiple convolutional layers directly approximate the mapping $H(x)$, the residual mapping $F(x) = H(x) - x$ is to be approximated. A

shortcut connection, also known as a **skip connection**, from the input to the output adds an identity mapping to the output of the stacked layers [68]. Augmenting neural networks with skip connections surprised the community by enabling the training of networks of more than 1,000 layers with significant performance gains. The skip connections in the residual blocks facilitate preserving the norm of the gradient, avoiding by this manner the vanishing gradient problem and leading to stable backpropagation [69].

3.6. Training stages

We used in the training of our main audio tagging system a semi-supervised approach, where the whole process can be broken down into 3 steps:

- **Training Stage 1**

  In the first stage we have built our models following a supervised learning approach. We trained ResNet-34 on the curated data $\Omega_{curated}^{train}$ for several cycles using cyclic cosine annealing [70]. Cosine annealing means that the cosine function is used as the learning rate annealing function. Specifically, we lower the learning rate at a very fast pace, encouraging the model to converge towards its first local minimum after $\boldsymbol{e}$ epochs; this process is repeated several times to obtain multiple convergences. The cosine annealing function has been shown to perform better than alternatives like simple linear annealing in practice [71]. In our case we have stored our models at 64^{th}, 128^{th}, 192^{th} and 256^{th} epochs (i.e. four cycles and the length of a cycle is set to 64 epochs).

- **Pseudo Labeling**

  This step represents our approach in making of the pseudo labels. The core idea of our system is based on this section. The models at 64^{th}, 128^{th}, 192^{th} and 256^{th} epochs from **stage 1** are used to generate pseudo-labels (i.e. guessed labels) of the **Noisy data** Ω_{noisy}. Specifically, we predict the labels of each sample from Ω_{noisy} using the four trained models. Note that the output consists of a vector of probabilities not labels. Then, we compute the average of these predictions to obtain the final results. Next we apply a sharpening function to the obtained pseudo-labels. We have chosen the

sharpening function used in MixMatch [16] due to its simplicity and its huge success (refer to **Chapter 2 Section 4.2**). The value of temperature **T** used in the sharpen function has different impact on the pseudo labels probabilities, the figures below illustrate the effect of T on the pseudo-labels.

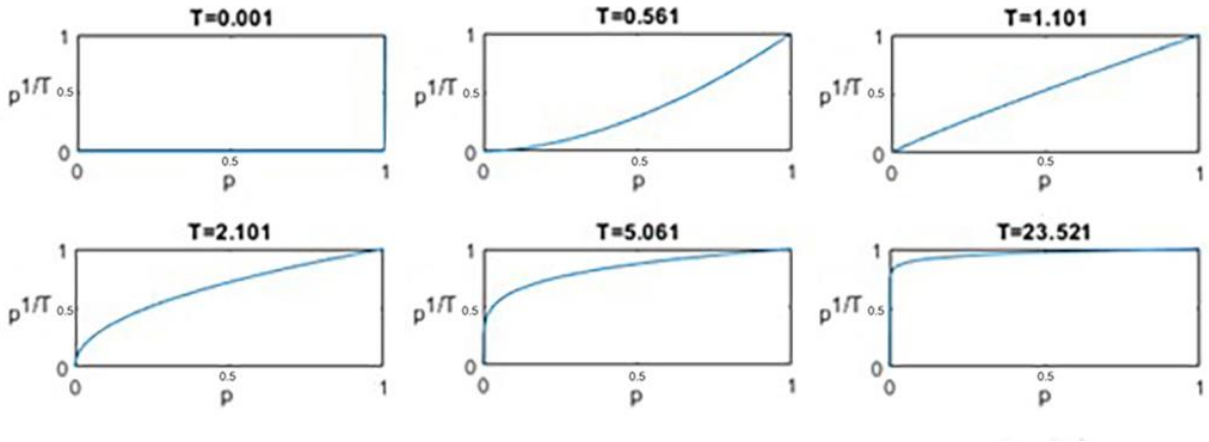


Figure 3.6: Plots of different values of sharpening.

On one hand, when T is small (i.e. less than 1), the pseudo-labels probabilities that are close to 1 get promoted, otherwise the farther values from 1 get degraded. On the other hand, when T gets larger (i.e. larger than 1), the values that are farther than 1 and close to 0 almost get the same importance as values close to 1 (refer to **Chapter 4, experiment 2** for more details on sharpening values).

Note that the next stage requires supervised data, oracle output or crisp labels not probabilities. To this end, a key element for using the pseudo labels for training is to convert the probabilities from float to True or False (i.e. as 1 or 0). We have used a threshold value to achieve the conversion and to decide on what labels are suited as ground-truth labels, so we used the **mean** values of each sample predictions (i.e. pseudo-labels probabilities of each audio sample). The values that are bigger or equal to the mean are treated as True labels and the rest as False labels.

- **Training Stage 2**

  In this stage, we **assign** the processed **pseudo labels** to the **Noisy data** Ω_{noisy} (i.e. replace noisy labels with refined pseudo labels) to obtain $\Omega_{noisy(new)}$ a new data for training, then **combine** both the **Curated data** $\Omega_{\boldsymbol{curated}}^{\boldsymbol{train}}$ with the new **Noisy data** $\Omega_{noisy(new)}$. Specifically, the combined training data $\Omega_{combined}^{train}$ have both curated labels and pseudo-labels. To ensure a proper balance between the curated labels and pseudo-labels and avoid dominance of pseudo-labels over curated labels (i.e. pseudo labels are numerous and not accurate as curated labels), we have assigned a different weight for both to use in the sampling process to helps establish a well **proportional** number of all classes labels in each training **batch**. A higher weight is assigned to the curated labels to give it more importance. The new training set $\Omega_{combined}^{train}$ is used to train the stored model from **stage 1** (e.g. we used 256th, trained for 256 epochs), the model is trained for one or more cycles.

3.7. Conclusion

In this chapter, we have described the process used to train our model, starting from presenting our system characterizing, data preparation, Feature extraction and the steps of the training process. In the following chapter, we will present the results of the experiments and analyze them in order to derive conclusions based on numerous statistical comparisons.

Chapter 4 : Experimental Design and Results Discussion

4.1. Introduction

This chapter presents the results of the experiments done on our audio tagging system. The experiments aim to evaluate the performance of our models while conducting statistical tests of the use of noisy labels in training, impact of different values of sharpening, ensemble learning and the effect of increasing training epochs.

4.2. Dataset Description

In the making of our system, we used FSDKaggle2019 dataset (24 GB) [27], which accommodates more than 102 hours of content. The dataset employs audio clips from the following sources:

- Freesound Dataset (FSD): a dataset under development based on Freesound content organized with the AudioSet Ontology [72]. These data are used to create the curated train set and the test set, which has been manually labeled by humans following a data labeling process using the Freesound Annotator platform [73].

- The soundtracks of a pool of Flickr videos taken from the Yahoo Flickr Creative Commons 100M (YFCC100M) dataset [74]. These data are used to create the noisy train set, which were labeled using automated heuristics applied to the audio content and metadata. Hence, a substantial amount of label noise can be expected.

The dataset is depicted in Figure 4.1, and its main characteristics are listed in Table 4.1.

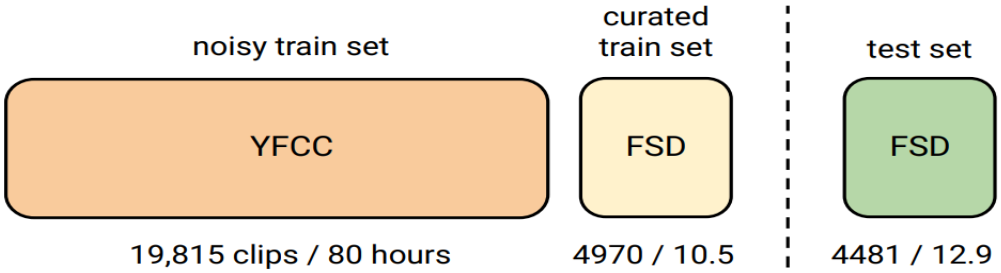


Figure 4.1: Data split in FSDKaggle2019, including number of clips/duration in hours, and data origin. Colors depict quality of labels: orange, yellow and green correspond to noisy labels, correct but potentially incomplete labels, and exhaustive labels, respectively [27].

Table 4.1: Main stats of the sets in FSDKaggle2019. * A few classes have slightly less than 75 clips [27].

Aspect	Curated train	Noisy train	Test
Clips/class	~75*	300	~50-15
Total clips	4970	19815	4481
Labels/clips	1.2	1.2	1.4
Clip length	~0.3-30s	~15s	~0.3-30s
Total duration	~10.5h	~80h	~12.9h
Labeling	Correct(inexhaustive)	Noisy	exhaustive

The audio data is labeled using a vocabulary of 80 labels from Google's AudioSet Ontology [73], which can be split into 8 categories represented in Figure 4.2.

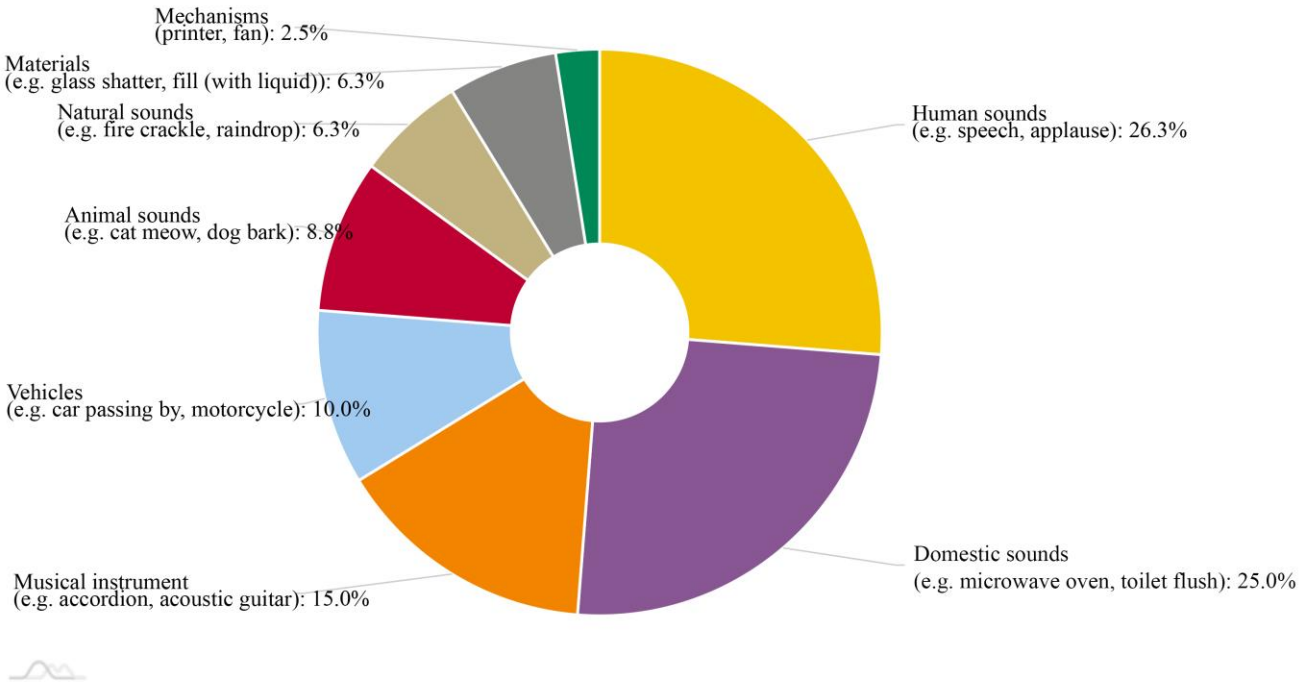


Figure 4.2: Pie Chart of the 8 categories present in FSDKaggle2019 sets [27].

4.3. Development Tools and Environments

The training process of machine learning model requires tons of computational power from CPU, GPU, RAM and storage. To meet the requirement for making the learning of our models faster we relied on Google Colaboratory cloud service. The development tools and environments setup are as follows.

Python

Python is an object-oriented open-source programming language [75]. It is one of the most used languages in the world in machine learning and it is used in Google Colaboratory.

Google Colaboratory

We developed our models and carried out the experiments using the GPU & CPU computational resources provided by Google Colaboratory which is more commonly referred to as “Google Colab” or just simply “Colab”. Google Colab is a free Google-

Hosted cloud service based on the Jupyter environment for machine learning education and research [76].

Google Drive Storage

We chose google drive as our online storage to store the datasets, google drive is an online storage that offers 15GB for free for every Gmail account. Using the online storage make it easier to access the datasets and load all files to Google Colab runtime.

PyTorch

PyTorch is a library for python programs that facilitates building deep learning projects, and it emphasizes flexibility and allows deep learning models to be expressed in idiomatic python [77].

Librosa

Librosa is a python library that has been built to deal with music signal processing and it provides implementations of a variety of common functions used throughout the field of music information retrieval [78].

In addition, to other required libraries for data manipulation and plotting, we used numpy for mathematical operation on arrays and to store spectrograms as npy files for fast reading while training, Pandas for csv file manipulations, matplotlib for plotting graphical representation, pretrainedmodels library to get the base model of Resnet34. Table 4.2 provides versions of the utilities and libraries we used including the additional ones.

Table 4.2: Version of Utilities and libraries used in experiments.

Library	Version
Python	3.6.9
Librosa	0.8.1
PyTorch “torch”	1.9.0
NumPy	1.19.5
Pandas	1.1.5
Matplotlib	3.2.2
Pretrainedmodels	0.7.4

4.4. Experiment 1: Impact of noisy labels and multitasking

This experiment investigates two different approaches for learning from noisy data: Multitasking and Pseudo Labeling. To this end, we have compared our system, which we denote $ResNet_1$+PL (PL stands for pseudo-labeling), with 4 Audio Tagging systems. Recall that the process of building $ResNet_1$+PL system involves three stages: Training stage 1, Pseudo-Labeling and Training stage 2. First, we have trained our Residual Network architecture (presented in Chapter 3, Section 5) on curated data only and discarded the noisy dataset. Second, we have generated the predictions associated with the noisy set, i.e. the pseudo labels, via the aforementioned model $ResNet_1$. Finally, we have carried on the training of our model for 64 epochs on both curated and noisy sets, while considering the inferred pseudo-labels instead of the provided noisy labels.

- **$ResNet_1$:** This system implements a classical supervised learning approach, which considers only the curated set during training and discards the noisy dataset. We have trained ResNet-34 for 256 epochs.
- **$ResNet_2$:** We have built this system using both curated and noisy data. To this end, we consider learning from the curated and the noisy set ***as two distinct tasks***; then, combine the obtained losses within the context of ***multitask learning***. Note that we have trained this system with the provided noisy labels. We have trained this model for 256 epochs.

- **ResNet$_1$+PL+MT:** In order to assess the impact of MultiTask Learning (**MT**) on ResNet$_1$+PL, during the training stage 2, we have formulated the learning process as a multitasking problem, while considering learning from the curated and the noisy datasets as two distinct tasks. Note that we have also trained this model for 64 epochs.
- **ResNet$_2$+PL+MT:** This system adopts a similar training strategy as ResNet$_1$+PL+MT. However, it generates the pseudo-labels using ResNet$_2$ i.e. a model trained on noisy labels.

Note that for all systems we have set the temperature parameter T to 0.5. We have chosen this value based on exploratory experiments; extensive treatment on this matter will be further explored in experiment 2.

Table 4.3 gives the category-wise lwlrap scores of the aforementioned systems. The last row specifies the average score of each technique over all categories.

Table 4.3: Category-wise lwlrap scores of all systems.

Categories	ResNet$_1$	ResNet$_2$	ResNet$_1$+PL	ResNet$_1$+PL+MT	ResNet$_2$+PL+MT
Animals (7 tags)	71.31%	71.04%	**72.01%**	70.31%	71.16%
Domestic (20 tags)	69.20%	69.21%	67.99%	69.46%	**69.78%**
Human (21 tags)	67.71%	66.24%	**72.60%**	69.50%	69.49%
Materials (5 tags)	60.02%	59.76%	59.05%	58.47%	**60.90%**
Mechanisms (2 tags)	49.02%	45.18%	45.77%	**50.22%**	49.07%
Musical Instruments (12 tags)	74.85%	74.30%	77.56%	75.96%	**78.42%**
Natural (5 tags)	**65.19%**	64.79%	62.37%	64.57%	63.89%
Vehicles (8 tags)	62.15%	59.89%	**65.99%**	65.54%	64.11%
System lwlrap	67.54%	66.72%	**68.94%**	68.33%	68.77%

The initial analysis of the above table indicates that $ResNet_1$+PL demonstrates superiority over its counterpart, followed by $ResNet_2$+PL+MT and $ResNet_1$+PL+MT. However, $ResNet_2$ yields the worst scores. This behavior is expected since we have trained this model on noisy data with the provided noisy labels, which can deteriorate the overall performance. In addition, we observe that training with pseudo labels yields overall better predictive performances.

These scores do not reveal considerable differences. In addition, according to numerous papers on Statistical Machine Learning, when the results on different categories of data are not comparable, their averages are meaningless [79]. To cope with this shortcoming, appropriate statistical tests should be conducted thoroughly [43]. To this end, we have statistically compared the performances of these techniques using Friedman test. This test first ranks the techniques for each tag according to the lwlrap scores in ascending order. Specifically, the best performing technique gets the rank 1, the second best gets 2, ..., etc. The average ranks of these methods are specified in the Table 4.4. Then, we have compared the mean ranks of these approaches. We have assumed, under the null hypothesis, that all systems perform similarly and the observed differences are merely due to chance.

Table 4.4: Friedman test Ranking results of all systems.

Algorithm	Ranking
$ResNet_1$	3.42
$ResNet_2$	3.48
$ResNet_1$+PL	2.68
$ResNet_1$+PL+MT	2.8
$ResNet_2$+PL+MT	**2.6**

Friedman test has rejected this hypothesis with $F_F = 6.09 > F(4, 316) = 6.08$ for $\alpha = 0.0001$ (FF is distributed according to the F distribution with $5 - 1 = 4$ and $(5 - 1) \times (80 - 1) = 316$ degrees of freedom). The rejection of the null

hypothesis confirms the existence of at least two systems that have significantly different lwlrap scores.

For further analysis of these results, we have followed up the Friedman test with a Nemenyi post-hoc test with at a 5% significance level with the critical value $q0.05 = 2.72$ and the critical difference $CD = 0.68$. This test aims at identifying pairs of algorithms that are significantly different. The results of the Nemenyi test are depicted in Figure 4.3. On the horizontal axis, we represent the average ranks of each system (given in Table 4.4), and join the groups of systems that are not significantly different using thick lines. On the top left, we display the critical difference CD used in this experiment.

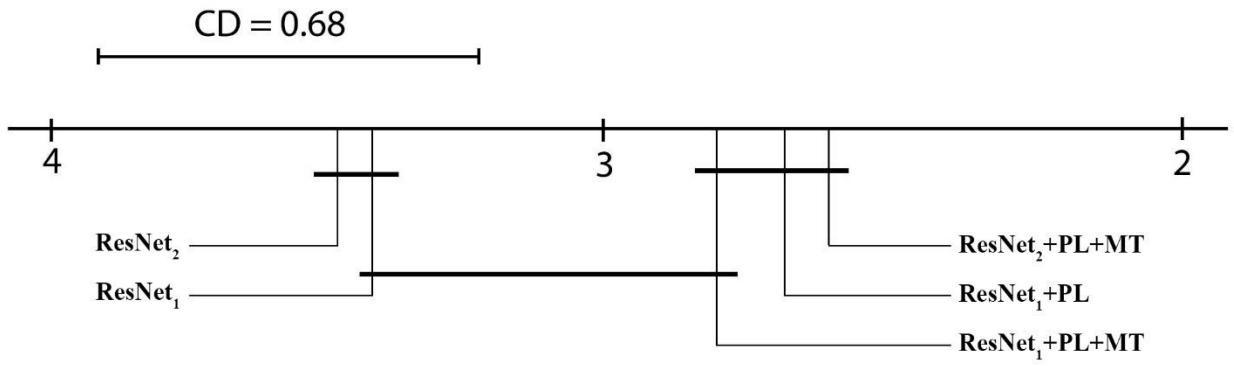


Figure 4.3: Comparison of all systems with Nemenyi test.

Based on the analysis of the above figure, we can identify two groups of systems: semi-supervised models, i.e. PL-based systems (ResNet$_1$+PL,…), and supervised models (ResNet$_1$ and ResNet$_2$). Notice that systems within the same group achieve similar performances, and the observed differences are solely due to chance. Most importantly, the test provides strong evidence that training models on the generated pseudo labels yields significantly better results than its counterpart systems, which confirms our initial assumption. Particularly, ResNet$_1$+PL+MT and ResNet$_1$ are linked together; hence, we are unable to decide which group ResNet$_1$+PL+MT and ResNet$_1$ belong to, at 5% significance level. However, while conducting the Nemenyi test at 10% significance level ($CD = 0.61$), we have found that ResNet$_1$+PL+MT is significantly better than ResNet$_1$.

From the above findings we can derive two important conclusions:

1. Training a ResNet model on pseudo labels has a positive impact on the generalization ability of audio tagging systems.
2. Introducing multitasking does not considerably boost the performance when trained on the provided labels i.e. without pseudo labeling.

4.5. Experiment 2: Impact of sharpening hyperparameter "Temperature T"

In order to further investigate the generation of pseudo labels, we have tested the impact of varying the hyperparameter T on $ResNet_1$+PL. Recall that in the case of $ResNet_1$+PL the value of T regulates the number of inferred tags per audio sample (please refer to **Chapter 2 Section 4.2** for additional details on this parameter). We have performed this experiment with 11 different values of sharpening hyperparameter $T \in \{0.01, 0.06, 0.14, 0.33, 0.4, 0.5, 0.66, 1, 1.33, 2, 100\}$. Figure 4.4 exhibits the change in lwlrap scores per category.

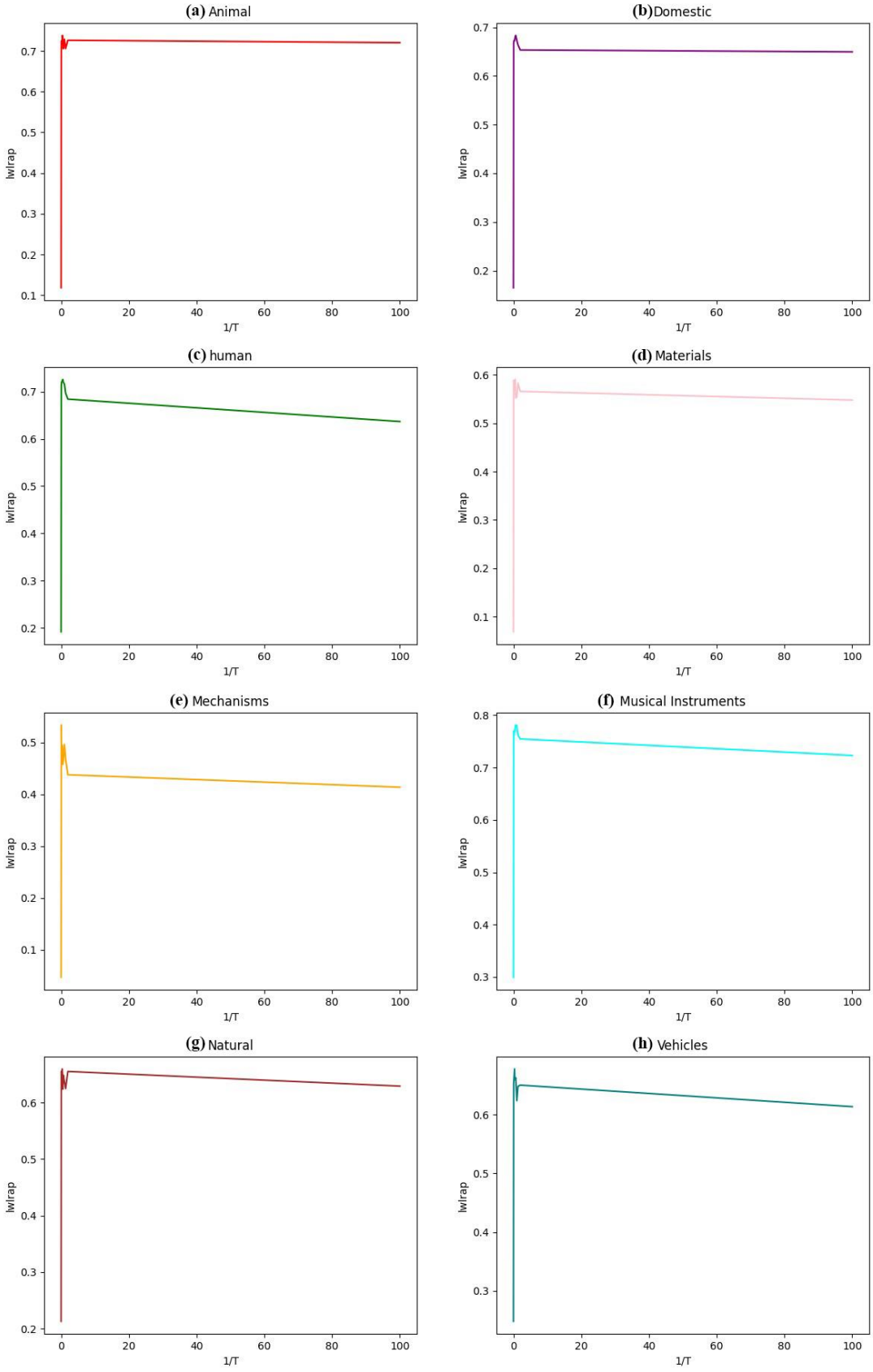


Figure 4.4: Effect of the temperature parameter on the lwlrap score for each tag category.

For all categories we observe that the curves display the same pattern shown in Figure 4.5 that exhibits the average lwlrap over all categories. The performance improves as T increases and reaches its highest value between $T\epsilon[\frac{1}{3},\frac{1}{2}]$. Then, it knows a slight decrease and keeps that score with slight variations.

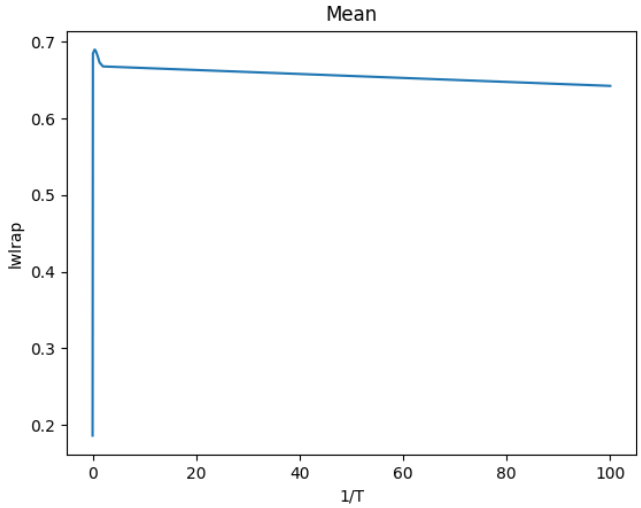


Figure 4.5: Effect of the temperature parameter on the lwlrap score for all categories.

It is worth underscoring that the change in lwlrap scores noticed in range $[\frac{1}{3},\frac{1}{2}]$ may contain significant information; thus, it requires further investigation. In order to statistically examine this and unravel significant differences, we have first conducted the Friedman test. We have assumed that the observed differences are due to random behavior. This test rejects our hypothesis with $FF = 4.29 > F(7,553) = 3.79$ for $\alpha = 0.0005$ (FF is distributed according to the F distribution with $8-1=7$ and $(8-1)\times(80-1) = 553$ degrees of freedom), which indicates an existence of at least one pairwise significant difference.

Next, we have compared these scores in a pairwise manner based on the Wilcoxon test in Table 4.5. The first row of each entry specifies the number of Win/Tie/Loss of the technique in the column over the technique in the row; whereas, the second row shows the p-values for the Wilcoxon test. If the entry is bold, this means that the number of wins/losses over 80 tags is statistically significant using the Wilcoxon test. For example, a $p-value$ = 0.05 indicates that the observed differences are significant at 5% significance level.

Table 4.5: Comparison of the 11 models on a pairwise manner based on Wilcoxon test.

T		2	1.33	1	0.66	0.5	0.4	0.33	0.14	0.06	0.001
100	W/T/L	60/0/20	58/0/22	59/0/21	65/0/15	64/0/16	64/0/16	68/0/12	61/0/19	57/0/23	1/0/79
	P-value	$\mathbf{5.82 \times 10^{-6}}$	$\mathbf{5.57 \times 10^{-6}}$	$\mathbf{2.45 \times 10^{-7}}$	$\mathbf{6.29 \times 10^{-8}}$	$\mathbf{2.94 \times 10^{-8}}$	$\mathbf{1.74 \times 10^{-8}}$	$\mathbf{5.79 \times 10^{-9}}$	$\mathbf{1.04 \times 10^{-7}}$	$\mathbf{2.25 \times 10^{-6}}$	$\mathbf{8.79 \times 10^{-15}}$
2	W/T/L		45/0/35	50/0/30	55/0/25	59/1/20	52/0/28	55/0/25	57/0/23	50/0/30	0/0/80
	P-value		0.1920	**0.0031**	$\mathbf{9.45 \times 10^{-5}}$	$\mathbf{7.36 \times 10^{-5}}$	$\mathbf{1.60 \times 10^{-4}}$	$\mathbf{7.90 \times 10^{-5}}$	$\mathbf{3.04 \times 10^{-4}}$	**0.0037**	$\mathbf{7.84 \times 10^{-15}}$
1.33	W/T/L			44/0/36	48/0/32	55/0/25	50/0/30	51/1/28	48/0/32	46/0/34	0/0/80
	P-value			0.0941	**0.0067**	$\mathbf{7.59 \times 10^{-4}}$	**0.0057**	**0.0013**	**0.0193**	0.0594	$\mathbf{7.84 \times 10^{-15}}$
1	W/T/L				41/0/39	46/1/33	52/0/28	49/0/31	42/0/38	45/0/35	0/0/80
	P-value				0.3932	0.0732	0.0594	0.0557	0.505	0.4808	$\mathbf{7.84 \times 10^{-15}}$
0.66	W/T/L					42/0/38	39/0/41	40/0/40	37/0/43	39/0/41	0/0/80
	P-value					0.4989	0.6625	0.7155	0.7698	0.5780	$\mathbf{7.84 \times 10^{-15}}$
0.5	W/T/L						38/0/42	36/0/44	34/0/46	36/0/44	0/0/80
	P-value						0.8216	0.4868	0.2268	0.4372	$\mathbf{7.84 \times 10^{-15}}$
0.4	W/T/L							41/0/39	37/0/43	38/0/42	**0/0/80**
	P-value							0.8516	0.7012	0.3672	$\mathbf{7.84 \times 10^{-15}}$
0.33	W/T/L								38/0/42	41/0/39	**0/0/80**
	P-value								0.2497	0.4989	$\mathbf{7.84 \times 10^{-15}}$
0.14	W/T/L									36/0/44	**0/0/80**
	P-value									0.5845	$\mathbf{7.84 \times 10^{-15}}$
0.06	W/T/L										**0/0/80**
	P-value										$\mathbf{7.84 \times 10^{-15}}$

Merging the results of the above statistics and those displayed in Figure 4.4, we can elaborate the following figure.

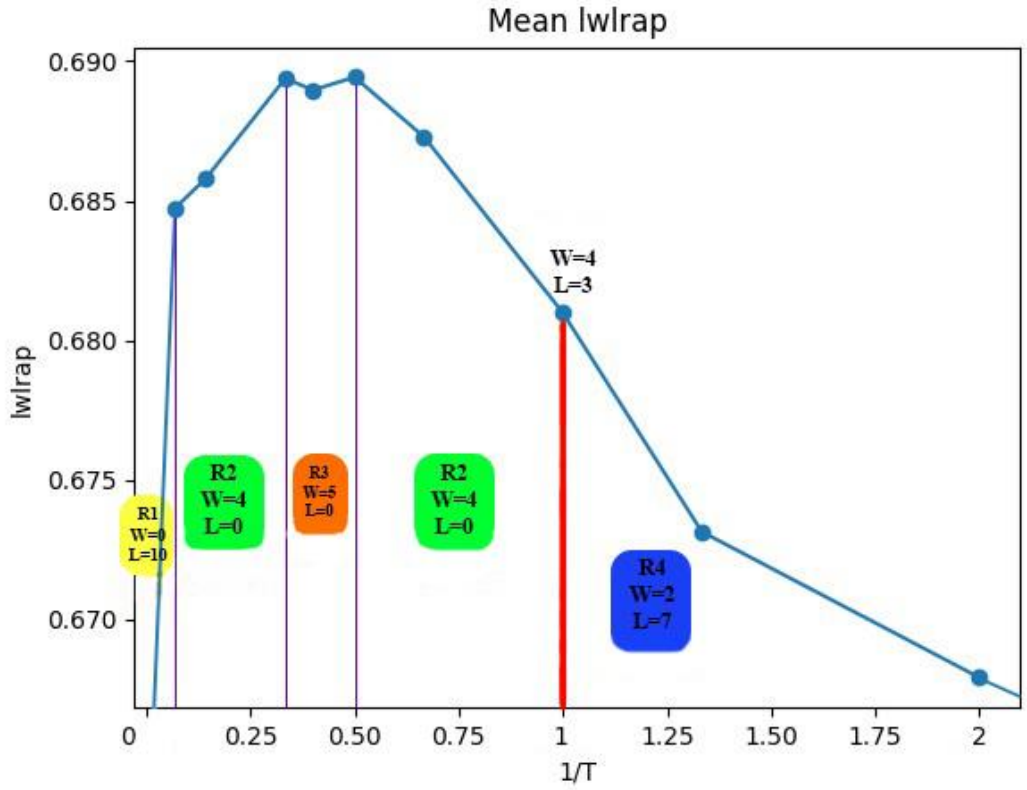


Figure 4.6: Effect of the temperature parameter on the mean lwlrap score. W and L stands for the number of significant wins and losses.

The analysis of the results reported in the above figure indicate that the lwlrap score improves gradually as T increases. Then, it settles when $T\epsilon[\frac{1}{3},\frac{1}{2}]$ with some variations; and it drops dramatically as the temperature value exceeds $\frac{1}{2}$. Most importantly, the lwlrap score reaches its highest value within the range $T\epsilon[\frac{1}{3},\frac{1}{2}]$.

Examining Figure 4.6, we can identify several regions:

- **When $T < 0.06$:** The performance is very poor; and the above statistics indicate that ResNet+PL_T = 0.001 achieves 10 significant losses. This behavior is expected since when the temperature parameter is very low (close to 0), the sharpening curves (Figure 3.6 of Chapter 3) are skewed to the right; hence, very high scores (close to 1) are promoted, whereas, lower scores are flattened and ignored. As a

result, the system outputs/produces/infers fewer tags that can be meaningless and wrong i.e. weak performance.

- **When $0.06 \leq T < \frac{1}{3}$:** We observe a crucial increase in the predictive scores. According to the Wilcoxon tests, ResNet+PL_T=0.06 and ResNet+PL_T = 0.14 score 4 significant wins over ResNet+PL_T=0.001, 1.33, 2, 100.
- **When $\frac{1}{3} \leq T \leq \frac{1}{2}$:** The performance attains its highest value with 5 significant wins. We believe that when T is set to values within this range the sharpening/thresholding step produces the appropriate number of tags per audio file, which boosts the overall scores considerably.
- **When $T > 1$:** The figure shows that the lwlrap scores gradually decrease as the temperature value increases. As indicated by the statistical results, ResNet+PL_T=1.33 and ResNet+PL_T=2 significantly lose 7 times and achieve only 2 significant wins. This behavior is rather anticipated since the sharpening curves are skewed to the left when T is larger than 1 (refer to **Figure 3.6 in Chapter 3**). Therefore, when T gets larger, more importance will be granted to values far from 1, generating more tags per audio recording which can deteriorate the overall performance.

4.6. Experiment 3: Ensemble Learning

We observe from Table 4.3 that the previously developed systems provide diverse predictions of the events present in the dataset, with various confidences. Specifically, the systems characterizations affect the predictive performance differently and achieve better scores on particular sound events. A large body of literature has demonstrated that amalgamating several learners could improve the generalization ability [80][81]. Most importantly, ensemble learning combines the strengths of each system by merging their predictions [82].

Motivated by this, we have tested the impact of fusing our developed systems. To this end, we have created an ensemble made of 19 base learners as summarized in Table 4.7.

Table 4.6: List of all models used in the experiment.

System	System Description
$ResNet_{1\ e=64}$	$ResNet_1$ trained for 64 epochs
$ResNet_{1\ e=128}$	$ResNet_1$ trained for 128 epochs
$ResNet_{1\ e=192}$	$ResNet_1$ trained for 192 epochs
$ResNet_{1\ e=256}$	$ResNet_1$ trained for 256 epochs
$ResNet_{1\ e=320}$	$ResNet_1$ trained for 320 epochs
$ResNet_{2\ e=64}$	$ResNet_2$ trained for 64 epochs
$ResNet_{2\ e=128}$	$ResNet_2$ trained for 128 epochs
$ResNet_{2\ e=192}$	$ResNet_2$ trained for 192 epochs
$ResNet_{2\ e=256}$	$ResNet_2$ trained for 256 epochs
$ResNet_{2\ e=320}$	$ResNet_2$ trained for 320 epochs
$ResNet_1$+PL $_{e=320}$	$ResNet_{1\ e=256}$ +PL trained for 64 epochs
$ResNet_1$+PL+MT $_{e=320}$	$ResNet_{1\ e=256}$ +PL+MT trained for 64 epochs
$ResNet_1$+PL+MT $_{e=384}$	$ResNet_{1\ e=256}$ +PL+MT trained for 128 epochs
$ResNet_1$+PL+MT $_{e=448}$	$ResNet_{1\ e=256}$ +PL+MT trained for 192 epochs
$ResNet_1$+PL+MT $_{e=512}$	$ResNet_{1\ e=256}$ +PL+MT trained for 256 epochs
$ResNet_2$+PL+MT $_{e=320}$	$ResNet_{2\ e=256}$ +PL+MT trained for 64 epochs
$ResNet_2$+PL+MT $_{e=384}$	$ResNet_{2\ e=256}$ +PL+MT trained for 128 epochs
$ResNet_2$+PL+MT $_{e=448}$	$ResNet_{2\ e=256}$ +PL+MT trained for 192 epochs
$ResNet_2$+PL+MT $_{e=512}$	$ResNet_{2\ e=256}$ +PL+MT trained for 256 epochs

We report in Table 4.8 the lwlrap scores of the obtained ensemble named **Ens**. Note that we also include the performance of $ResNet_1$, $ResNet_2$, $ResNet_1$+PL, $ResNet_1$+PL+MT and $ResNet_2$+PL+MT in order to highlight the improvement provided by **Ens**.

Table 4.7: lwlrap results of all systems using post-processing and ensemble learning.

Systems	**ResNet$_1$**	**ResNet$_2$**	**ResNet$_1$+PL**	**ResNet$_1$+PL+MT**	**ResNet$_2$+PL+MT**	**Ens**
lwlrap	67.93%	66.86%	68.94%	68.33%	68.22%	**72.21%**

The results given in Table 4.8 indicate that **Ens** outperforms the other methods in most cases. In order to confirm the significance of the observed differences, we have compared the performances of these techniques using the average ranks over the 80 sound events. Following Demsar's recommendations [40], we have first conducted a Friedman test to statistically compare the performance of these systems, assuming that all systems perform similarly. This test rejects this hypothesis with $FF = 20.05 > F(5,395) = 18.69$ for $\alpha = 1.0 \times 10 - 16$ (FF is distributed according to the F distribution with $6 - 1 = 5$ and $(6 - 1) \times (80 - 1) = 395$ degrees of freedom), and therefore confirms the existence of at least one pair of systems with significantly different performances.

Second, because we are only interested in comparing **Ens** with the other alternatives, we proceed with a Bonferroni-Dunn test while considering **Ens** as the control system. Figure 4.7 shows the results of the Bonferroni-Dunn test at a 0.1% significance level with the critical value $q0.001 = 3.71$ and the critical difference $CD = 1.001$. On the horizontal axis, we represent the average ranks of each system (given in Table 4.7), and we mark using a thick line the interval of one CD to the left and to the right of the average rank of **Ens**. Any system with a rank outside this area is significantly different from **Ens**.

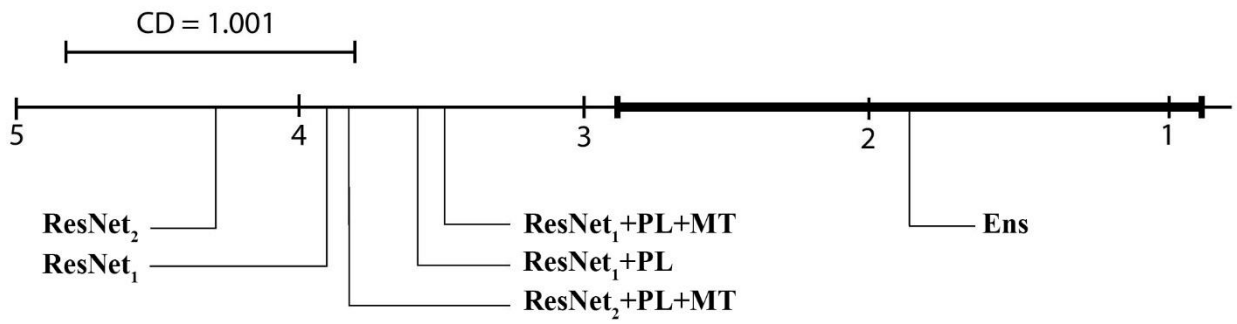


Figure 4.7: Comparison of the ensemble learning system against the other systems with Bonferroni-Dunn test.

The analysis of Bonferroni-Dunn test results illustrated by Figure 4.7 indicates that **Ens** has the lowest rank and all the other systems have higher ranks and fall outside the marked interval. Therefore, we can conclude that **Ens** significantly outperforms the individual models, which coincides with our initial observations.

4.7. Experiment 4: Impact of number of epochs

To investigate the impact of the number of epochs on $ResNet_1$+PL, we have carried out the following experiment. During the **second stage of training**, we have varied the number of epochs and measured the lwlrap scores on the test set. The results of his experiment are illustrated in Figure 4.8.

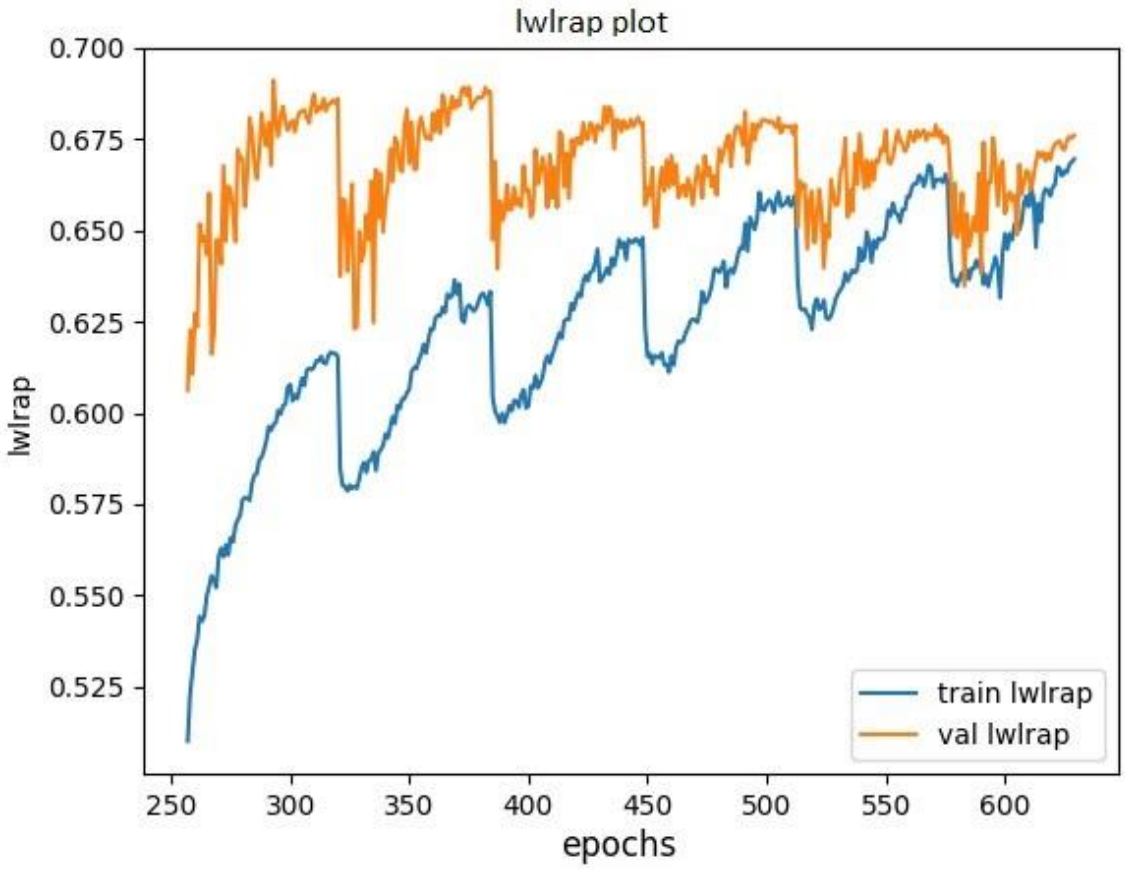


Figure 4.8: Plot of training and validation lwlrap scores of 640 epochs.

The analysis of this curve can be summarized by two main observations:

- The lwlrap performance (testing lwlrap) increases as the number of epochs increases; then, it settles at a certain score and keeps that with some variation. Next, it drops rapidly after 64 epochs. This pattern is repeated throughout all the remaining epochs. This behavior occurs because the learning rate is updated after each cycle of training i.e. after every 64 epochs, which causes the reported sudden drop in lwlrap scores. In addition, we observe fluctuations all along the testing curve. This behavior is an open area of research where it could be caused by ResNet instability [83].
- The overall testing lwlrap performance decreases at a slow pace as the number of cycles increases on one hand; on the other hand, the overall training lwlrap performance is increasing. These results indicate that the model is overfitting the training data which justifies the decrease in the predictive performance. According

to several studies, the problem of overfitting might occur in cases when learning is performed for longer runs or the training data are not representative of the problem at hand [84][85].

Based on the above analysis, training on the pseudo labels ($ResNet_1$+PL) should not exceed 1 cycle i.e. 64 epochs.

4.8. Conclusion and summary of empirical findings

From the experiments that we conduct, we can acquire the following:

- Using the pseudo labels to train ResNet models, improves the generalization ability of the audio tagging system.
- Applying the Multi-task approach to train a system with Noisy labels does not noticeably demonstrate any superiority over pure supervised models. i.e. without using the pseudo labels.
- Setting the values of the temperature T in the right range, does help the model to perform better.
- The simplest type of ensemble learning "averaging", can really boost the generalization ability of audio tagging systems.
- Cyclic cosine annealing can aid the model to reach its first local minimum quickly after a few epochs.
- Training the ResNet model for longer runs, i.e. a large number of epochs, could lead to overfitting and a decrease in performance.

CONCLUSION

The goal of this thesis was to analyze and empirically compare multi-label Audio Tagging systems. To this end, we built several tagging systems and conducted multiple experiments to analyze the differences in behavior between these systems.

1. Contributions and summary of experimental findings

Our contribution to this research area involves deep investigations using a recent large-scale dataset. We have analyzed the impact of fine-tuning the sharpening temperature hyperparameter on the performance of audio tagging systems. Additionally, we have investigated the effect of training the ResNet architecture using cyclic cosine annealing for an extended number of epochs. To avoid deriving conclusions affected by chance, we have used well-known statistical tests to perform comparisons between all of our audio tagging systems. We can derive the following conclusions from the conducted experiments:

- Using the pseudo-labels to train ResNet models effectively improves the generalization ability of the audio tagging system.
- Applying the multi-task approach to train an audio tagging system with noisy labels does not demonstrate noticeable superiority over systems trained in a supervised fashion.
- Proper tuning of the temperature value T i.e. setting its value in the right range, helps the audio tagging model to produce better results.
- Ensemble learning applied via averaging can significantly boost the generalization ability of audio tagging systems.
- Cyclic cosine annealing can aid audio tagging models to reach a first local minimum after training only for a few epochs.
- Training a ResNet-based model for longer runs, i.e. a large number of epochs, could lead to overfitting and a decrease in performance.

2. Limits and Future work

This research work has yet multiple interesting areas that can be further explored. Our developed audio tagging systems have achieved noticeable increases in performance. However, we believe that these performances can be further improved by developing better techniques to use noisy labels i.e. take advantage of the huge amount of web audio data. Throughout our research work, we have used noisy labels following a multi-task approach. Nonetheless, deeper investigations and experimentations are required to take full advantage of multi-task learning within the context of semi-supervised audio tagging.

Another appealing aspect for future experimentations is the choice of the neural network architecture. Experiments using variations of the ResNet architecture (e.g. ResNet-50) or other deep network architectures (e.g. EnvNet, AlexNet) are to be conducted in future works. In addition, future work should consider using raw audio signals (i.e. not spectrograms) as inputs to train the audio tagging systems. The aforementioned data types can also be combined as they are expected to compensate each other [13]. All our experiments have been conducted using the pseudo-labeling approach. However, future work can experiment using other semi-supervised techniques like self-training, Mean Teacher, MixMatch or other recently developed techniques [15]. The impact of employing different ratios of labeled and pseudo-labeled data for training should also be investigated, tinkering with the ratio could potentially lead to improvement in the overall performance.

Machine Learning is a very wide and interesting domain. When it comes to automating human tasks, this domain is full of great potential. Audio Tagging research is vast and resourceful, delving into this research field has helped us acquire valuable knowledge about audio signals, their properties and how to develop audio tagging systems in a semi-supervised fashion. We have nonetheless encountered a few struggles. One of the main struggles is the lack of robust hardware for deep learning. This has caused a substantial increase in training time, which prevented us from conducting further experiments especially in this short period of time. We have learned and gained various useful skills like thesis writing guidelines, implementing machine learning experiments using the python language, as well as developing systems on a cloud-based

platform like Google Collaboratory. In conclusion, we have only explored the tip of the iceberg of machine learning. The rest is yet to be researched as the future of this domain appears very promising.

References

[1] D. Gerhard, “Audio Signal Classification : History and Current Techniques,” *Dep. Comput. Sci. Univ. Regina Regina, Saskatchewan, CANADA*, pp. 0–37, 2003.

[2] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, “Semantic Annotation and Retrieval of Music and Sound Effects,” *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 16, no. 2, pp. 467–476, 2008, doi: 10.1109/TASL.2007.913750.

[3] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, “Large-Scale Weakly Supervised Audio Classification Using Gated Convolutional Neural Network,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2018, vol. 2018-April, pp. 121–125, doi: 10.1109/ICASSP.2018.8461975.

[4] I.-Y. Jeong and H. Lim, “Audio Tagging System Using Densely Connected Convolutional Networks,” 2018.

[5] S. Dimitrov, J. Britz, B. Brandherm, and J. Frey, “Analyzing sounds of home environment for device recognition,” vol. 8850, 2014.

[6] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, “A Survey of Audio-Based Music Classification and Annotation,” *IEEE Trans. Multimed.*, vol. 13, no. 2, pp. 303–319, 2011, doi: 10.1109/TMM.2010.2098858.

[7] V. Morfi, “applied sciences Deep Learning for Audio Event Detection and Tagging on Low-Resource Datasets,” 2018, doi: 10.3390/app8081397.

[8] E. Bouteillon, “Specmix : A Simple Data Augmentation And Warm-Up Pipeline To Leverage Clean And Noisy Set For Efficient Audio Tagging,” *Dcase.Community*, 2019, [Online]. Available: https://www.semanticscholar.org/paper/SPECMIX-%3A-A-SIMPLE-DATA-AUGMENTATION-AND-WARM-UP-TO-Bouteillon/510237b72788e2229778e290cdecc50d7e35841f.

[9] D. Park *et al.*, “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition,” 2019, pp. 2613–2617, doi: 10.21437/Interspeech.2019-2680.

[10] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “MixUp: Beyond

empirical risk minimization," in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018, pp. 1–13.

[11] X. Hong and G. Liu, "MULTI-LABEL AUDIO TAGGING SYSTEM FOR FREESOUND 2019 : FOCUSING ON NETWORK ARCHITECTURES , LABEL NOISY AND LOSS FUNCTIONS Technical Report," 2019.

[12] V. Verma, A. Lamb, J. Kannala, Y. Bengio, and D. Lopez-Paz, "Interpolation Consistency Training for Semi-supervised Learning," 2019, pp. 3635–3641, doi: 10.24963/ijcai.2019/504.

[13] O. Akiyama and J. Sato, "Multitask Learning and Semi-Supervised Learning With Noisy Data for Audio Tagging," 2019, [Online]. Available: http://dcase.community/documents/challenge2019/technical_reports/DCASE2019_ Akiyama_94_t2.pdf.

[14] Y. LIU and Q. WEI, "STACKED CONVOLUTIONAL NEURAL NETWORKS FOR AUDIO TAGGING WITH NOISE LABELS."

[15] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. The MIT Press, 2006.

[16] D. Berthelot, N. Carlini, I. Goodfellow, C. Raffel, A. Oliver, and N. Papernot, "MixMatch : A Holistic Approach to Semi-Supervised Learning," no. NeurIPS, pp. 1–14, 2019.

[17] K. He, "Deep Residual Learning for Image Recognition."

[18] K. Palanisamy, D. Singhania, and A. Yao, "Rethinking CNN Models for Audio Classification."

[19] Y. Kumar, "From Image Classification to Audio Classification."

[20] T. Virtanen, M. D. Plumbley, and D. Ellis, *Computational analysis of sound scenes and events*. 2017.

[21] S. Abu-el-haija, J. Lee, P. Natsev, and G. Toderici, "YouTube-8M : A Large-Scale Video Classification Benchmark."

[22] K. Trohidis and G. Kalliris, "Multi-label classification of music into emotions MULTI-LABEL CLASSIFICATION OF MUSIC INTO EMOTIONS," no. May 2014, 2011, doi: 10.1186/1687-4722-2011-426793.

[23] S. Oramas *et al.*, "Multi-label music genre classification from audio, text, and

images using deep features."

[24] R. Duda, P. Hart, and D. G.Stork, "Pattern Classification," in *Wiley Interscience*, vol. xx, 2001.

[25] A. M. Badshah, J. Ahmad, N. Rahim, and S. W. Baik, "Speech Emotion Recognition from Spectrograms with Deep Convolutional Neural Network," no. July 2019, 2017, doi: 10.1109/PlatCon.2017.7883728.

[26] F. S. Cabral, H. Fukai, and S. Tamura, "Feature Extraction Methods Proposed for Speech Recognition Are E ff ective on Road Condition Monitoring Using Smartphone Inertial Sensors," 2019.

[27] E. Fonseca, M. Plakal, F. Font, D. Ellis, and X. Serra, "Audio Tagging with Noisy Labels and Minimal Supervision," 2019, pp. 69–73, doi: 10.33682/w13e-5v06.

[28] J. Kadis, *The Science of Sound Recording*. 2012.

[29] M. Bosi, R. Goldberg, and J. Mitchell, *Introduction to Digital Audio Coding and Standards*, vol. 13. 2004.

[30] S. Arunachalam, S. Khairnar, and B. S. Desale, "The Fast Fourier Transform Algorithm and Its Application in Digital Image Processing," *Math. theory Model.*, vol. 3, pp. 267–273, 2013.

[31] D. G. Manolakis and V. K. Ingle, *Applied digital signal processing: theory and practice*. New York: Cambridge University Press, 2011.

[32] J. B. Allen and L. R. Rabiner, "Unified Approach to Short-Time Fourier Analysis and Synthesis," vol. 1, no. November, pp. 1558–1564, 1977.

[33] K. Zheng, Z. Xia, Y. Zhang, X. Xu, and Y. Fu, "Speech Emotion Recognition based on Multi-Level Residual Convolutional Neural Networks," no. July, 2020.

[34] B. Zhang, J. Leitner, and S. Thornton, "Audio Recognition using Mel Spectrograms and Convolution Neural Networks," 2020.

[35] H. Meng, T. Yan, F. Yuan, and H. Wei, "Speech Emotion Recognition from 3D Log-Mel Spectrograms with Deep Learning Network," *IEEE Access*, vol. PP, p. 1, 2019, doi: 10.1109/ACCESS.2019.2938007.

[36] L. Rabiner and R. Schafer, "Introduction to Digital Speech Processing," *Found. Trends Signal Process.*, vol. 1, pp. 1–194, 2007.

[37] H. C. Vemula, "Multiple Drone Detection and Acoustic Scene Classification with

Deep Learning," 2018.

[38] Y. Guo and S. Gu, "Multi-Label Classification Using Conditional Dependency Networks," pp. 1300–1305, 1994.

[39] H. Cheng, Z. Qin, C. Feng, Y. Wang, and F. Li, "Conditional Mutual Information-Based Feature Selection Analyzing for Synergy and Redundancy," *ETRI J.*, vol. 33, 2011, doi: 10.4218/etrij.11.0110.0237.

[40] J. Demsar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.

[41] S. i and F. Herrera, "An Extension on 'Statistical Comparisons of Classifiers over Multiple Data Sets' for all Pairwise Comparisons," *J. Mach. Learn. Res. - JMLR*, vol. 9, 2008.

[42] N. Japkowicz and M. Shah, "Evaluating Learning Algorithms: A Classification Perspective," 2011.

[43] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci. (Ny).*, vol. 180, no. 10, pp. 2044–2064, 2010, doi: https://doi.org/10.1016/j.ins.2009.12.010.

[44] X. Zhu and A. Goldberg, "Introduction to Semi-Supervised Learning," 2009.

[45] H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, 2020, doi: 10.1007/s10994-019-05855-6.

[46] D. Lee, "Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks," no. August, 2015.

[47] A. Tarvainen and H. Valpola, "Mean teachers are better role models : Weight-averaged consistency targets improve semi-supervised deep learning results arXiv : 1703 . 01780v6 [cs . NE] 16 Apr 2018."

[48] K. Sohn *et al.*, "FixMatch : Simplifying Semi-Supervised Learning with Consistency and Confidence," no. NeurIPS, 2020.

[49] I. Goodfellow, "Deep Learning."

[50] G. Brier, "VERIFICATION OF FORECASTS EXPRESSED IN TERMS OF PROBABILITY," *Mon. Weather Rev.*, vol. 78, pp. 1–3, 1950.

[51] S. Laine and T. Aila, “Temporal Ensembling for Semi-Supervised Learning,” *ArXiv*, vol. abs/1610.02242, 2017.

[52] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles,” 2017.

[53] T. Miyato, S. Maeda, M. Koyama, and S. Ishii, “Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, pp. 1979–1993, 2019.

[54] A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. J. Goodfellow, “Realistic Evaluation of Deep Semi-Supervised Learning Algorithms,” no. NeurIPS, 2018.

[55] E. Labb and T. Pellegrini, “Improving Deep-learning-based Semi-supervised Audio Tagging with Mixup,” pp. 1–9.

[56] S. Ruder, “An Overview of Multi-Task Learning in Deep Neural Networks ∗ arXiv : 1706 . 05098v1 [cs . LG] 15 Jun 2017,” no. May, 2017.

[57] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, “Cross-stitch Networks for Multi-task Learning.”

[58] C. Rosenbaum, T. Klinger, and M. Riemer, “Routing Networks: Adaptive Selection of Non-linear Functions for Multi-Task Learning,” *ArXiv*, vol. abs/1711.01239, 2018.

[59] S. Thrun, “A Bayesian / Information Theoretic Model of Learning to Learn via Multiple Task Sampling *,” vol. 39, pp. 7–39, 1997.

[60] L. Duong, T. Cohn, S. Bird, and P. Cook, “Low Resource Dependency Parsing : Cross-lingual Parameter Sharing in a Neural Network Parser,” pp. 845–850, 2015.

[61] Y. Yang and T. M. Hospedales, “Trace norm regularised deep multi-task learning,” in *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings*, 2019, no. 2014, pp. 2015–2018.

[62] A. Schindler, T. Lidy, and A. Rauber, “Comparing Shallow versus Deep Neural Network Architectures for Automatic Music Genre Classification.”

[63] C. Simon Haykin (McMaster University, Hamilton, Ontario, *Neural Networks - A Comprehensive Foundation - Simon Haykin.pdf*. 2005.

[64] A. Koutsoukas, K. J. Monaghan, X. Li, and J. Huan, “Deep - learning : investigating deep neural networks hyper - parameters and comparison of

performance to shallow methods for modeling bioactivity data," *J. Cheminform.*, pp. 1–13, 2017, doi: 10.1186/s13321-017-0226-y.

[65] L. Alzubaidi *et al.*, *Review of deep learning : concepts , CNN architectures , challenges , applications , future directions*. Springer International Publishing, 2021.

[66] A. Mohamed, "Deep Neural Network Acoustic Models for ASR," 2014.

[67] M. Z. Alom *et al.*, "A State-of-the-Art Survey on Deep Learning Theory and Architectures," *Electronics*, vol. 8, p. 292, 2019, doi: 10.3390/electronics8030292.

[68] A. Scenes, "Adsc Submission for Dcase 2017 : Acoustic Scene Classification Using Deep," *Dcase 2017*, no. November, pp. 2–6, 2017.

[69] A. Zaeemzadeh, N. Rahnavard, and M. Shah, "Norm-Preservation: Why Residual Networks Can Become Extremely Deep?," *CoRR*, vol. abs/1805.07477, 2018, [Online]. Available: http://arxiv.org/abs/1805.07477.

[70] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017, pp. 1–16.

[71] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get M for free," in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017, pp. 1–14.

[72] J. Gemmeke *et al.*, "Audio Set: An ontology and human-labeled dataset for audio events," *2017 IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 776–780, 2017.

[73] E. Fonseca *et al.*, "Freesound Datasets: A Platform for the Creation of Open Audio Datasets," 2017.

[74] B. Thomee *et al.*, "YFCC100M: the new data in multimedia research," *Commun. ACM*, vol. 59, pp. 64–73, 2016.

[75] M. Lutz, "Learning Python: Powerful Object-Oriented Programming," 2008.

[76] E. Bisong, *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. .

[77] E. Stevens and L. Antiga, "Deep Learning with PyTorch," 2020.

[78] B. Mcfee *et al.*, "librosa : Audio and Music Signal Analysis in Python," no. August

2020, 2015, doi: 10.25080/Majora-7b98e3ed-003.

[79] M. Sugiyama, “Introduction to Statistical Machine Learning,” 2015.

[80] B. Pantic, “Ensemble of Convolutional Neural Networks for General Purpose,” 2018.

[81] J. Read, B. Pfahringer, and G. Holmes, “Multi-label Classification Using Ensembles of Pruned Sets,” in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 995–1000, doi: 10.1109/ICDM.2008.74.

[82] L. Rokach, “Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography,” *Comput. Stat. Data Anal.*, vol. 53, pp. 4046–4072, 2009, doi: 10.1016/j.csda.2009.07.017.

[83] J. Zhang, B. Han, L. Wynter, B. Kian, H. Low, and M. Kankanhalli, “Towards Robust ResNet : A Small Step but a Giant Leap,” no. Section 3, pp. 4285–4291, 2018.

[84] I. Tetko, D. Livingstone, and A. Luik, “Neural Network Studies. 1. Comparison of Overfitting and Overtraining,” *J. Chem. Inf. Comput. Sci.*, vol. 35, pp. 826–833, 1995, doi: 10.1021/ci00027a006.

[85] Z. Li, L. Liu, C. Dong, and J. Shang, “Overfitting or Underfitting? Understand Robustness Drop in Adversarial Training,” pp. 1–10, 2020, [Online]. Available: http://arxiv.org/abs/2010.08034.