Université de BLIDA 1
Faculté des Sciences
Département d'Informatique

Master Thesis

Option: *Ingénerie du Logiciel*

DEEP CO-TRAINING FRAMEWORK FOR SEMI-SUPERVISED AUDIO TAGGING

By :

CHEIFA IKRAM

In front of a jury composed of:

OUKID Lamia	President
BACHA Sihem	Examiner
YKHLEF Hadjer	Supervisor
DIFFALLAH Zhor	Supervisor

2021/ 2022

Abstract

Audio tagging, also known as Sound Event Recognition, is concerned with the development of systems that are able to recognize sound events. A sound event is perceived as a separate individual entity that we can name and recognize, such as helicopter, glass breaking, baby crying, speech, etc. Considerable attention has been geared towards audio tagging for various applications, such as information retrieval, music tagging, and acoustic monitoring. The general framework for audio tagging usually involves two major steps: feature extraction and classification. Clearly, obtaining well-annotated, strongly labeled data is an expensive and time-consuming process. Therefore, a large portion of recent development has been devoted to effectively using weakly labeled data extracted from websites like Youtube, Freesound, or Flickr. Various semi-supervised learning approaches have been proposed in the literature. We can cite Mean Teacher, Pseudo Labeling, Mix Match, and most recently, Deep Co-training. The purpose of this project consists of devising an audio tagging system within the semi-supervised learning paradigm, specifically the Deep Co-training framework. Such systems essentially use both labeled and unlabeled audio data. In addition, our system is trained on two different datasets :Urban8k and Environmental Sound Classification, based on a deep residual neural network (ResNet) and a wide residual neural network (WideResNet). We supported our analysis and discussion with numerous statistical tests to analyze and compare our results. We have investigated the impact of differentiating the supervised ratio on the system's performance and have tested the impact of various variants of DCT systems based on different adversarial attacks. The results demonstrate the efficacy of the Deep Co-training SSL strategy that significantly boosts the overall performance.

Keywords:
Audio Tagging, Semi-supervised learning, Deep Co-training, Feature Extraction, Statistical Tests.

Résumé

L'étiquetage audio, également connu sous le nom de reconnaissance d'événements sonores, est concerné par le développement des systèmes capables de reconnaître les événements sonores. Un événement sonore est perçu comme une entité individuelle distincte que nous pouvons nommer et reconnaître, comme l'hélicoptère, le bris de verre, les pleurs de bébé, la parole, etc. Plusieurs recherches ont montré l'importance et l'utilité de l'étiquetage audio dans diverses applications nommées: la recherche d'informations, l'étiquetage de la musique et la surveillance acoustique. Cette technique est basée sur deux étapes majeures: l'extraction des caractéristiques et la classification. C'est clair que l'acquisition de données bien annotées et fortement étiquetées est un processus coûteux et long. Dès lors, une grande partie du développement récent a été consacrée à l'utilisation efficace des données faiblement étiquetées extraites de sites Web tels que Youtube, Freesound ou Flickr. Pour ce fait, diverses approches d'apprentissage semi-supervisé ont été proposées, nous pouvons citer: Mean Teacher, Pseudo Labeling, Mix Match, et Deep Co-training. L'objectif de ce projet consiste à concevoir un système d'étiquetage audio dans le paradigme d'apprentissage semi-supervisé, en particulier dans le cadre de Deep co-training. Notre système utilise des données audio étiquetées et non étiquetées, et il est entraîné sur deux ensembles de données différents : Urban8k et Environmental Sound Classification, basés sur un réseau de neurones résiduels profonds (ResNet) et un vaste réseau de neurones résiduels (WideResNet). Nous avons soutenu notre analyse et notre discussion par nombreux tests statistiques pour examiner et comparer nos résultats. Nous avons étudié l'impact de la différenciation du ratio surveillé sur les performances du système et nous avons testé l'impact de diverses variantes de systèmes DCT basés sur différentes attaques adversaires. Les résultats obtenus démontrent l'efficacité de la stratégie SSL de Deep Co-Training qui a augmenté la performance globale.

Mots Clés:
Étiquetage Audio, Apprentissage semi-supervisé, Deep Co-training, Extraction des caractéristiques, Testes Statistique.

الملخص

يهتم وضع العلامات الصوتية، المعروف أيضا باسم التعرف على حدث الصوت(SER)، بتطوير الأنظمة القادرة على التعرف على الأحداث الصوتية. يعتبر حدث الصوت كيانا فرديا منفصلا يمكننا تسميته والتعرف عليه، مثل المروحية والكسر الزجاجي وبكاء الطفل والكلام، إلخ. وقد تم توجيه اهتمام كبير نحو وضع علامات صوتية لمختلف التطبيقات، مثل استرجاع المعلومات ووضع علامات على الموسيقى والمراقبة الصوتية . من الواضح أن الحصول على بيانات مشروحة بشكل جيد وموصفة بشكل قوي عملية مكلفة ومستهلكة للوقت. لذلك، تم تخصيص جزء كبير من التطوير الحديث لاستخدام البيانات ذو العينات الضعيفة والتي تم استخراجها من مواقع ويب مثل YouTube أو Freesound أو Flickr بشكل فعال. يتمثل الغرض من هذا المشروع في وضع نظام لوضع العلامات الصوتية في إطار نموذج التعلم شبه الخاضع للإشراف (SSL)، وعلى وجه التحديد إطار التدريب المشترك العميق (DCT). يجب أن يكون النظام قادرا على استخدام كل من بيانات الصوت المصنفة وغير المسماة. بالإضافة إلى ذلك، يتم تدريب نظامنا على مجموعتي بيانات مختلفتين: Urban8k ESC-10، استنادا إلى شبكة عصبية عميقة متبقية (ResNet) وشبكة عصبية كبيرة متبقية (WideResnet). لقد أيدنا مناقشتنا مع العديد من الاختبارات الإحصائية لتحليل نتائجنا ومقارنتها. لقد قمنا بدراسة تأثير التمييز بين النسبة الخاضعة للإشراف على أداء النظام، واختبرنا تأثير مختلف أنواع أنظمة DCT . تظهر النتائج فعالية استراتيجية SSL للتدريب المشترك العميق (DCT) التي تعزز الأداء العام بشكل كبير.

الكلمات المفتاحية

وضع العلامات الصوتية ، التعلم شبه الخاضع للإشراف ، التدريب المشترك العميق ، استخراج الميزات ، الاختبارات الإحصائية.

Acknowledgements

First and foremost, praise and thanks to God, the Almighty, for His showers of blessings throughout our research work. Thank you to my supervisor, Ms. YKHLEF H, and Diffallah Z, for providing guidance and feedback throughout this project. I would also like to thank my family for putting up with me being sat in the office for hours on end, and for providing guidance and a sounding board when required. And a special thanks to my friends Mokhtari A, Abbas N, Feknous L, and Hamel O, who never failed to assist me when needed.

Contents

Introduction **1**

PART I: FUNDAMENTALS OF AUDIO TAGGING **4**

1 OVERVIEW OF SOUND SIGNALS IN AUDIO TAGGING **5**

1.1 Introduction 5

1.2 Single-label Audio Tagging 6

1.3 Audio Signals 7

1.3.1 Audio Signal Components 7

1.3.2 Audio Signal acquisition 8

1.3.2.1 Audio signal sampling 9

1.3.2.2 Audio signal quantization 9

1.3.3 Audio Signal Representation 10

1.3.4 Fourier Transform 10

1.3.5 Short Time Fourier Transform 11

1.3.6 Pre-Processing 13

1.3.7 Feature Extraction 13

1.4 Machine Learning Essentials 16

1.4.1 Classification 16

1.4.2 Deep Learning For Audio Tagging 17

1.5 Statistical Tests 18

1.5.1 Wilcoxon signed-ranks test 19

1.5.2 Kruskal-Wallis Test 19

1.5.3 Bonferroni-Dunn Test 20

1.6 Conclusion 20

2 SEMI-SUPERVISED AUDIO TAGGING **21**

2.1 Introduction 21

2.1.1 Motivation 21
2.2 Main Assumptions in Semi Supervised Learning 22
2.2.1 The Smoothness Assumption 22
2.2.2 Cluster Assumption 22
2.2.3 The Low-Density Assumption 22
2.2.4 The Manifold Assumption 23
2.3 Semi-Supervised Learning Approaches 23
2.3.1 Hybrid methods 24
2.3.2 Consistency regularization 25
2.3.3 Generative Methods 27
2.3.4 Pseudo-Labeling-based method 28
2.3.4.1 Self-Training Models 28
2.3.4.2 Disagreement-Based Models 29
2.4 Conclusion 30

3 DEEP CO-TRAINING 31

3.1 Introduction 31
3.2 Co-training 31
3.3 Multi-view Co-training 32
3.4 Deep Co-Training 33
3.5 Adversarial attacks on neural networks 36
3.5.1 The Attack Surface 36
3.5.2 Adversarial Goals 37
3.5.3 The Adversarial Capabilities 38
3.6 Related works on DCT 48
3.7 Conclusion 50

PART II: DESIGN AND EXPERIMENTATIONS 51

4 DESIGN OF A SEMI-SUPERVISED AUDIO TAGGING SYSTEM 52

4.1 Introduction 52
4.2 Audio Tagging System Description 52
4.2.1 Pre-Processing 53
4.2.2 Feature Extraction 54
4.2.3 Neural Network Architectures 55
4.2.4 Adversarial Attacks 59
4.2.5 Evaluation 61

4.2.5.1 Cross Validation 61
4.2.5.2 Evaluation Measures 62
4.3 Conclusion 63

5 EXPERIMENTAL RESULTS AND DISCUSSION 64

5.1 Introduction 64
5.2 Dataset 64
5.2.1 ESC-10 Dataset 65
5.2.2 UrbanSound8K 65
5.3 Development Tools and Environments 66
5.3.1 Libraires 68
5.4 Experimental Design 69
5.4.1 Cross Validation 69
5.4.2 Feature Extraction 70
5.4.3 Training Stage 71
5.4.4 Evaluation Procedure 76
5.5 EXPERIMENTAL RESULTS 76
5.5.1 FIRST EXPERIMENT : Comparison between supervised, DCT and Mean Teacher 76
5.6 SECOND EXPERIMENT : Impact of Supervised Ratio on Audio Tagging Systems 81
5.7 THIRD EXPERIMENT : Impact of the network architecture 82
5.8 FOURTH EXPERIMENT : Analysis of adversarial attacks on DCT 84
5.8.1 EXPERIMENT 4A : Varying the magnitude of perturbation 84
5.8.2 EXPERIMENT 4B : Varying the adversarial attack strategy 85
5.9 Conclusion and summary of experimental findings 89

Conclusion 90

List of Figures

1.1 Audio Tagging process 6
1.2 Plot of sine wave from a recording of a dog barking. 7
1.3 The sampling process. 9
1.4 Analog to digital conversion. 10
1.5 : (a) Zoom into a 10-ms section of a waveform. (b-d) Comparison of the waveform with sinusoids of various frequencies. 11
1.6 : Time-domain and frequency-domain representations of a dog barking recording applied with the fourier transform function 11
1.7 Windowing a sinusoidal signal with Hann window. 12
1.8 STFT procedure with overlap–add method. 12
1.9 Spectrogram of a dog barking recording. 14
1.10 Mel-Spectrogram of a dog barking recording. 15
1.11 Log Mel-Spectrogram of a dog barking recording. 15
1.12 The overall architecture of the Convolutional Neural Network (CNN). 17

2.1 Illustrations of (a) Smoothness and low-density assumptions (b) Manifold assumption, and Cluster assumption depicted as the colors. 23
2.2 The Mean Teacher technique. 26
2.3 Architecture of the GAN. 28
2.4 Schematic diagram of pseudo-labeling steps. 29

3.1 Representation of Co-training algorithm. 33
3.2 Generic pipeline of an Automated Vehicle System. 36
3.3 A schematic representation of an evasion attack. 37
3.4 A schematic representation of a poisoning attack. 37
3.5 Block-box attack representation. 40

3.6 adversarial crafting framework in awhite-box attack. 41
3.7 Hyperplanes are depicted in solid lines and the boundary of P is shown in green dotted line. 45

4.1 The overall Audio Tagging System pipeline. 52
4.2 The overall process of feature extraction. 54
4.3 Building block of the Residual Neural Network 55
4.4 Building block of the Wide Residual Neural Network 57
4.5 The WRN28_2 architecture. 58
4.6 5-Fold Cross Validation 61
4.7 Confusion Matrix. 62

5.1 The class distribution of each fold of UBS8K 66
5.2 Screenshot of Kaggle notebook. 68
5.3 The pipeline of our baseline for an Audio Tagging system. 72
5.4 The pipeline of DCT and MT approaches for an Audio Tagging system. 75
5.5 Confusion Matrices of the supervised baseline (a) ,DCT (b) and MT (c) on WRN28_2 model. 80
5.6 Effect of the supervised ratio. 81
5.7 Impact of different network architecture. 83
5.8 Impact of different network architecture. 84
5.9 Comparison of the baseline system against the 4 other systems with the Bonferroni-Dunn test on the ESC-10 dataset. 87
5.10 Comparison of the baseline system against the 4 other systems with the Bonferroni-Dunn test on the ESC-10 dataset. 88

List of Tables

3.1 Summary of related work results. 50

4.1 Summary of ResNet variation architectures 56
4.2 Summary of WRN variation architectures. 59

5.1 Technical Specifications for a free kaggle notebook. 67
5.2 Version of libraries used in our study. 69
5.3 Log-Mel Spectrogram setup. 70
5.4 MT and DCT parameters setup. 74
5.5 Average F1-score results of DCT, MT and baseline system on UBS8K and USC-10. 77
5.6 Wilcoxon signed-rank test results. 78
5.7 MT and DCT parameters setup. 85
5.8 Average Ranks and F1-scores of DCT variants. 86
5.9 Time execution for different adversarial attacks. 88

Introduction

Context and problem statement/Background

The sounds in our everyday environment carry a huge amount of information. We are able to recognize and discern many different sound events surrounding us (e.g., streets, factories, cars passing by, car horns,... etc.). Our actions rely on the perception of these sounds (move when hearing a car honk, enter the class when the bell rings, run away when the fire alarm goes off, etc.). This task is not considered crucial for normal human beings since recognizing audio is an inborn ability that doesn't require any effort. On the other hand, state-of-the-art automatic processing of sounds by machines is still far behind. Further research is needed to develop robust systems capable of recognizing a wide range of sound events in realistic audio streams [1]. In recent years, huge amounts of multimedia recordings are generated and uploaded to the web every day. These recordings, such as music, field sounds, broadcast news, and television shows, contain sounds from a wide variety of sources. The demand for analyzing these sounds is increasing, and includes: audio segmentation [2], audio context classification [3], and audio tagging [4, 5]. The goal behind building audio tagging based systems is to develop more general machine listening systems capable of identifying and recognizing a wide range of acoustic events and audio scenes of an audio recording.
This task will provide insight towards the development of broadly applicable sound event classifiers. Potential applications include automatic description of multimedia content, acoustic information, acoustic surveillance, cataloging, searching in audio archives, or recognizing sound events happening in real time. Audio tagging has many applications, such as audio information retrieval [6], audio classification [7], acoustic scene recognition [8], industry sound [9], and music tagging [10].
The process of Audio Tagging consists of two main stages: Feature Extraction and Classification. The feature extraction process involves dividing the audio signal into equal overlapping frames in order to get a feature vector per frame. Each vector of data is associated with its

corresponding event label. Next, the deep neural network model maps the extracted feature vector to a corresponding event (class) label. However, such a task requires a large amount of labeled data to produce efficient predictions and high performance. Still, these datasets have to be hand-labeled by specialists, which is a very costly and time-consuming process since it involves a lot of human interaction [11]. Developing tagging systems is difficult due to this limitation. To address this issue, there is a large amount of audio data on-line, e.g. from YouTube or Freesound, which is either poorly labeled or unlabeled, which makes the ability to use them, predict them, and further add some new tags to the related audio quite a challenge, and it may also negatively impact the performance. Therefore, a wise solution would be to apply a semi-supervised learning approach such as the Deep Co-training technique that uses both labeled and unlabeled data and enhances the system's performance.

Contributions

The increasing availability of audio content through a vast distribution of channels has resulted in many researchers performing comparative studies between sound analysis systems [12]. Audio Tagging systems are actually much more challenging to design and evaluate, and they need to be carried out properly to ensure their performance significance. Therefore, researchers have focused on extracting relevant features, finding suitable classifiers, and applying recent techniques to enhance the overall performance. Moreover, a wide range of semi-supervised learning techniques were applied to audio data. However, a detailed evaluation of the recently presented approaches is required in order to determine the optimal solutions for performing the Audio Tagging task. We have created and examined the behavior of several systems in response to these requirements. In what follows, we summarize our main contributions:

- We have carried out experiments on the Environmental Sound Classification (ESC-10) and the URBAN8K(UBS8K) datasets. The audio clips were extracted from freesound. which are very diverse in terms of acoustic content, recording techniques, clip duration, etc.

- We have trained our labeled and unlabeled data on two semi-supervised (SSL) methods: Deep Co-training (DCT) and Mean Teacher (MT). We discussed their major steps and compared them to a simple baseline that was trained on the same model that the two techniques were trained on.

- We explored two different deep neural architectures consisting of residual neural networks(Resnets) and wide residual neural networks (WideResnets).

- We have analyzed the impact of the hyperparameter epsilon on the adversarial example and the whole model itself.
- We studied the behavior of different adversarial attacks and their impacts on the performance. We conducted our experiment analysis based on different statistical tests.

Thesis structure

This thesis covered four chapters. In Chapter 1, we cover some of the fundamentals behind sound signals. Specifically, we describe the several representations of sound signals and the most frequently used feature extraction methods in literature. In Chapter 2, we introduce some semi-supervised learning methodologies, some of which have been widely used for audio tagging. We provide in Chapter 3 a detailed description of the Deep Co-training methods along with the adversarial attacks used in our experiments. Chapter 4 consists of a detailed description of the experimental setup, including feature extraction and parameter setting; and along with experimental results and discussion, where we present the obtained results through performance tables and plots based on statistical tests. Finally, we conclude by summarizing the contributions of this thesis, the lines of limitation, and future work.

PART I: FUNDAMENTALS OF AUDIO TAGGING

This section explains the concepts needed to understand the ideas developed in this thesis. It consists of three chapters. Chapter 1 is divided into two main parts; the first one gives an overview of audio signals. Specifically, we describe data required for the development of Audio Tagging systems and highlight the importance of feature extraction to convert the signal into a convenient representation. As for the second part of this chapter we review some relevant concepts of classification providing a brief description of the fundamentals of classification, evaluation metrics and statistical tests used in this work. For Chapter 2 we provide a comprehensive overview of the notion of semi supervised learning along with its main assumptions and techniques. In Chapter 3 we have detailed explanations on the Deep-Co training methods along with the different attacks and previous application on it.

Chapter 1

OVERVIEW OF SOUND SIGNALS IN AUDIO TAGGING

1.1 Introduction

Sound carries a lot of information about our everyday environment and the actions that occur within it. Acoustic sensors capture sounds that allow computers to perceive the environment as humans do. Furthermore, the popularity of sound event recognition has increased and gained a lot of attention due to its great potential, and the widespread use of machine learning and deep learning in many fields. This has given computers the power to learn numerous solutions to change people's lives for the better. Sound recognition includes tasks such as acoustic scene classification [13], sound event detection [14], and audio tagging [4]. This latter task has become one of the most functional tasks nowadays since it has a diverse and wide range of applications such as surveillance [15], monitoring [16] and health care [17]. Audio tagging systems aim to predict acoustically relevant tags from the audio signal. Examples of tags can be acoustic events (e.g., air conditioner, car horn, drilling, gunshot, siren), animal sounds (e.g., dog bark, birds chirping, pig oink), human sounds (e.g., male or female speech), acoustic scenes (e.g., lakeside beach, forest path, metro-station), or even music sounds (e.g., guitar, trumpet). Audio tagging consists of two major steps: audio signal processing and; in our case, single-label audio classification [18]. Audio processing involves transforming the signal into a suitable representation that is relevant for a single-label audio classification task. As a result, a trained classifier is used to assign a label to every audio instance.

In this chapter, we provide the fundamental concepts of audio tagging in order to allow further understanding of our thesis. We initiate with a profound definition of audio signals and their components. Then, we briefly discuss time and frequency representations, and we explain the different feature extraction techniques that are widely used in literature.

Next, we present a few fundamental concepts of deep learning in the context of audio tagging,

including : neural network architectures, evaluation measures and statistical testing.
And finally, we summarize the main concepts that we have learned.

1.2 Single-label Audio Tagging

Audio tagging aims to identify sound events that occur in a given audio recording and enables a variety of artificial intelligence-based systems to disambiguate sounds and understand the acoustic environment [19]. The audio tagging task has been adopted in various fields, including music tagging [20], domestic audio tagging [21], and acoustic scene classification [22]. Audio tagging has a wide range of applications, including surveillance, monitoring, health care, and safety in the home, office, industry, and transportation. It has become one of the most researched topics in the field of acoustic signal processing [19, 8]. There are two main steps in the audio tagging process: audio signal processing and single-label classification. First, audio preprocessing is the main focus of signal processing where this process is used for removing silence, reducing noise in the audio, and making each sample of audio files the same duration [23]. After this operation, a feature extraction step is generally needed to transform the audio signal into a representation that highlights its physical properties. This provides the necessary information that is used in the single-label classification process, which takes the extracted feature as an input to train the classifier and test its performance on unseen data by predicting accurate event labels known as class labels. given a set of n labels $L = \{l_1, l_2, ...l_n\}$ and a set of m Items $I = \{i_1, i_2, ...l_m\}$, in single-label classification, the goal is to associate one label l to every item i [24, 8]. The overall Audio Tagging process is illustrated in Figure 1.1.

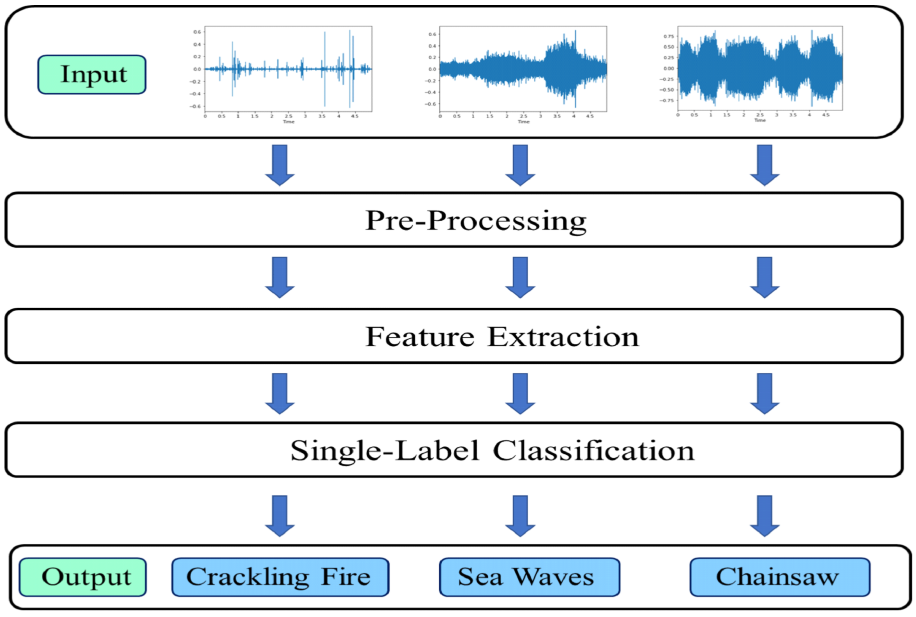


Figure 1.1: Audio Tagging process

1.3 Audio Signals

An audio signal is the representation of sound, which is a vibration that propagates through air by the movement of air molecules over time. These sounds can be broadly categorized into artificial sounds, speech, and music. For example, music can be viewed as the evolving pattern of notes [25]. Sound is a signal since it holds information about each vibration. These vibrations cause molecules to rattle against each other, producing what is called a sound wave. Air itself does not travel with the wave; each air molecule moves away from a rest point and eventually returns to it. When we hear a sound, we are sensing the vibrations in the air. The number of vibrations per second is known as the frequency [26]. The human auditory system is responsive to sounds in the frequency range of 20 Hz to 20 kHz as long as the intensity lies above the frequency-dependent "threshold of hearing" [25].

1.3.1 Audio Signal Components

In order to understand the different elements that represent an audio signal, we illustrate the audio wave of a sound as an example. Figure 1.2 shows the sine wave representation of our example that illustrates a periodic signal that is repeated after a fixed length of time, known as the period [27].

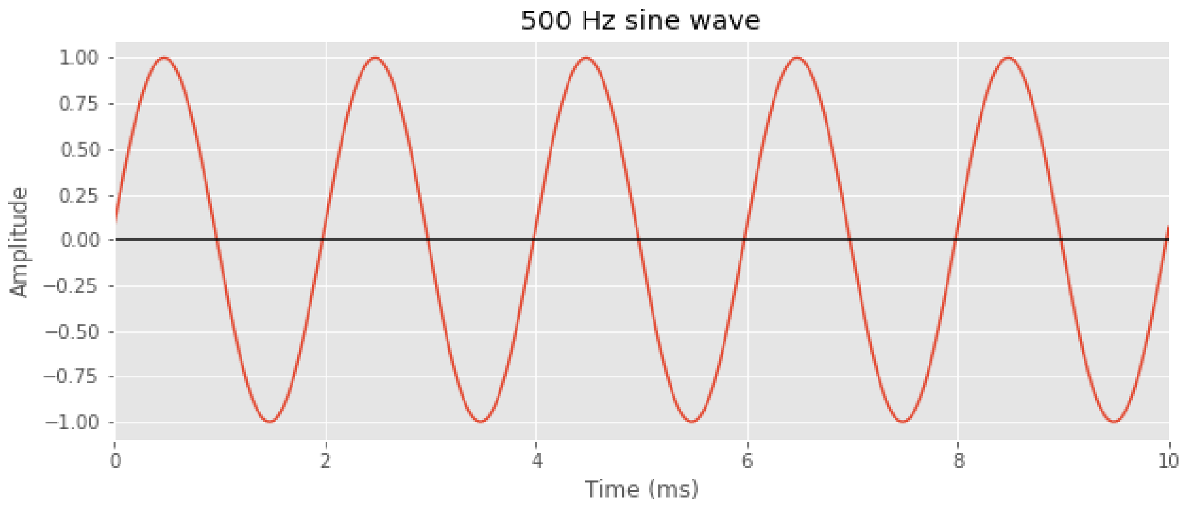


Figure 1.2: Plot of sine wave from a recording of a dog barking.

Periodic signals are characterized by the following components:

Amplitude: the amplitude is defined as the maximum displacement of vibrating particles in a medium from their mean position when the sound is created; in other words, it represents the magnitude of an air pressure disturbance [25].

Cycle: cycle is one repetition of the pattern. The instantaneous displacement waveform in Figure 1.2 shows five cycles, or five repetitions of the pattern. Since the wave has a frequency of 500 Hz, one cycle of the wave takes 0.002 seconds. This means that the first 0.01 seconds of the waveform contain 5 full cycles of the wave [25].

Period: period is the time required to complete one cycle of vibration. For our previous example, 5 cycles are completed in 0.01 second. The period is 0.01/5th of a second (s), or 0.002s (ie 2ms), since the most commonly used unit of measurement for period is the millisecond (ms) [25]:

$$1ms = \frac{1}{1,000s} = 0.001s = 10 - 3s \tag{1.1}$$

A somewhat less commonly used unit is the microsecond (μs):

$$1\mu s = \frac{1}{1,000,000s} = 0.000001s = 10 - 6s \tag{1.2}$$

Frequency: frequency is the number of cycles completed in one second. Its unit of measurement is the hertz (Hz). Fundamentally, frequency refers to the rate of vibration. The most crucial function of the auditory system is to serve as a frequency analyzer—a system that determines how much energy is present at different signal frequencies [25]. The formula for frequency is:

$$f = \frac{1}{t} \tag{1.3}$$

where f is the frequency in hertz and t is the period in seconds. So, for a period of 0.002 s:

$$f = \frac{1}{t} = \frac{1}{0.002} = 500Hz \tag{1.4}$$

1.3.2 Audio Signal acquisition

Audio signals are the representation of sound, which is generated and processed by a transducer. A transducer is an electronic device that takes physical energy as input and converts it into a signal. For example, a microphone takes physical sound waves and converts them into an electrical signal, which has a continuous range of values for both time and amplitude. Since a computer can only store and process a finite number of values, one has to convert the waveform into a discrete representation, i.e the acquisition is the process of converting the physical phenomenon we call sound into a representation suitable for digital processing. This process is commonly known as digitization, which involves sampling and quantization processes [26].

1.3.2.1 Audio signal sampling

Sampling a continuous time signal implies taking snapshots of the signal at specific instances of time, as shown in Figure 1.3 [5]. One important parameter is the sampling rate, which is the number of samples taken per second, measured in Hertz (Hz). For example, if we sample at a rate similar or higher than that shown in Figure 1.2, it is possible to reproduce a waveform almost identical to the original waveform. Therefore, we need to determine a convenient sampling rate, which is generally required to be higher than twice the frequency of the sampled signal [5].

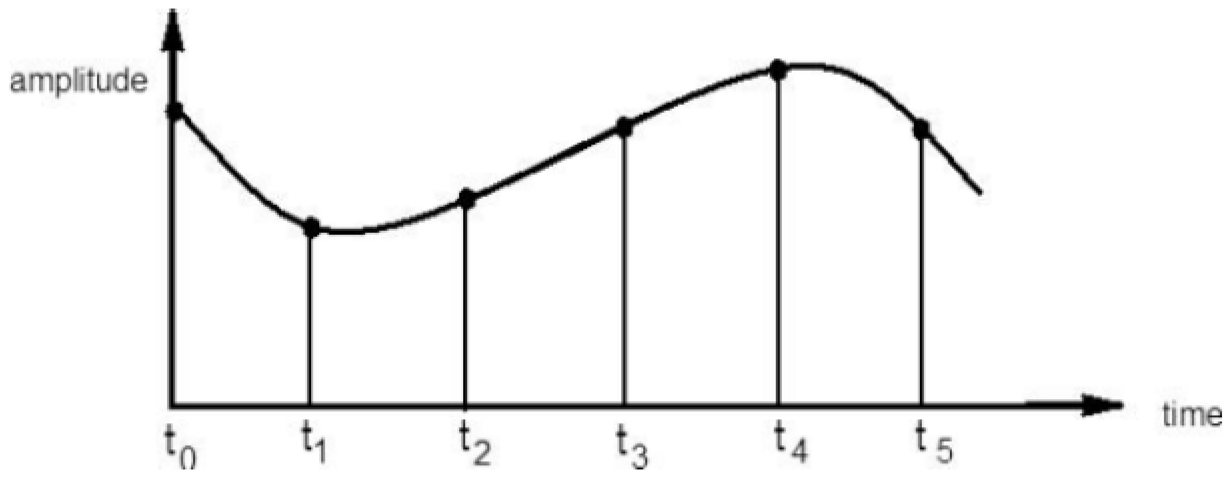


Figure 1.3: The sampling process.

1.3.2.2 Audio signal quantization

Quantization refers to the process of converting a continuous amplitude to a discrete one [5] by replacing the continuous values with a limited set of values separated by discrete steps. Usually, the number of steps is chosen to be a power of two, as this yields the most economical representation in binary digital electronics [28]. It is possible for quantization to be done by rounding to the nearest quantization level. Each level is coded using 2 bits in a four-level quantization system. The output of this process is therefore binary, as shown in Figure 1.4 [5].

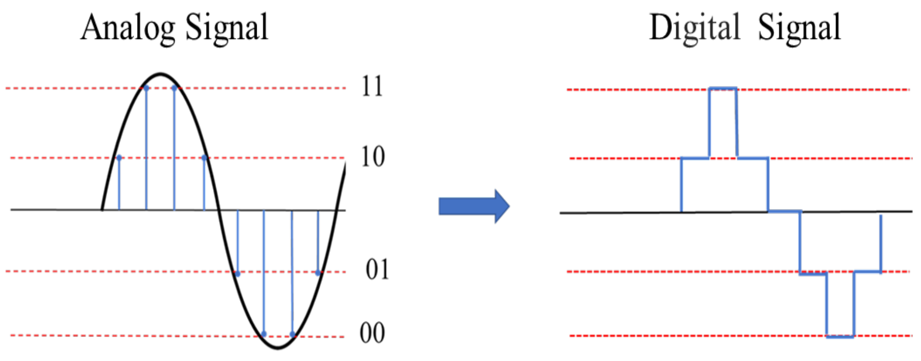


Figure 1.4: Analog to digital conversion.

1.3.3 Audio Signal Representation

Objects move or evolve; electrical signals change all along a continuous time-line and get converted into a discrete time-line as mentioned earlier. The observation of processes along this time-line is called the time-domain observation of the process [29]. However, it can not be interpreted in a straightforward way. It is nearly impossible to identify or localize sound events from a waveform and to distinguish between sound scenes [24]. Although time domain representation has been useful for years since it lines up with the human perception of sounds, unfortunately, it does not allow us to observe several characteristics of the signal. Therefore, it needs to be converted into a frequency with a pair of operators called transforms. Moreover ,a perfect example of a transform is the Fourier transform [29].

1.3.4 Fourier Transform

A signal is defined as any physical quantity that varies as a function of time; it conveys information in its patterns of variation. The manipulation of this information involves the acquisition, storage, transmission, and transformation of it. In order to find the different frequencies that are present in a signal, we apply the fourier transform that allows the passage from the temporal representation that shows the way the overall sound amplitude changes over time to the frequency representation that shows how much of the signal lies within each given frequency band over a range of frequencies [30]. We take, for example, this signal (Figure 1.5a) with a 10-ms section, behaving in a nearly periodic way. The main idea of Fourier analysis is to compare the signal with sinusoids of various frequencies (measured in Hz). Each such sinusoid may be thought of as a prototype oscillation. As a result, we obtain for each frequency a magnitude along with a phase. In the case where the magnitude is large, there is a high similarity between the signal and the sinusoid of frequency, and the signal contains a periodic oscillation at that frequency (see Figure 1.5b). In the case where the magnitude is small, the signal does not contain a periodic component at that frequency (see Figure 1.5c).

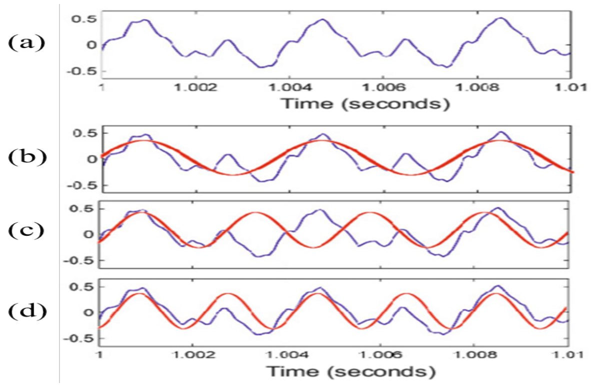
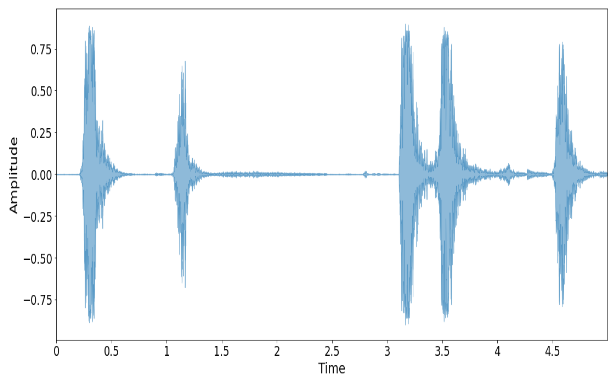
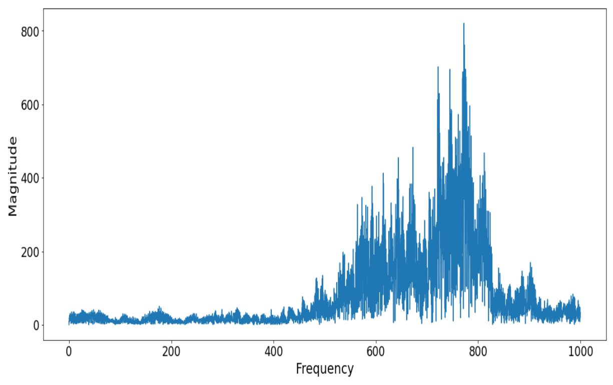


Figure 1.5: : (a) Zoom into a 10-ms section of a waveform. (b-d) Comparison of the waveform with sinusoids of various frequencies.

Furthermore, as illustrated by Figure 1.5d , we can observe a high similarity between the signal and the sinusoid with a frequency equal to 523 Hz. With this example, we can conclude that the fourier transform breaks up a signal into frequency components. For each frequency, it yields a magnitude and a phase that tells us to which extent the given signal matches a sinusoidal prototype oscillation of that frequency [31], Figure 1.6 displays a representation of the time-domain and frequency-domain applied with the fourier transform function.



(a) Time-domain representation.



(b) Frequency-domain representation.

Figure 1.6: : Time-domain and frequency-domain representations of a dog barking recording applied with the fourier transform function .

1.3.5 Short Time Fourier Transform

The Fourier transform assumes that the signal is analyzed over all time, i.e., an infinite duration, which implies that there can be no concept of time in the frequency domain and, likewise, no concept of frequency changing over time. The two domains can not be mixed together; they are orthogonal to each other. The Fourier transform does convert the audio from the time-domain to the frequency-domain, but it does not cover the change of the frequency over time, which is a must for an audio tagging system. That is why the concept of Short-Time Fourier transform (i.e., STFT) solves this issue by evaluating the frequency changes of a signal over time. To achieve this, we need to perform framing and windowing [32].

Framing allows the decomposition of a sound signal into a series of frames. In other words, the signal is cut into blocks of finite length, and then the Fourier transform of each block is computed, to end up with a time-dependent representation, which depicts how the spectrum changes over time, but the issue with applying framing is that there is often a discontinuity at the frame boundaries and that would corrupt the frequency spectrum estimation, and that would cause what is called spectral leakage. The solution to this problem is by multiplying a hamming window function by each short time frame signal, which would smooth the discontinuities of each beginning and end of the signal frame boundaries [32, 33]. Hann window function is shown in Figure 1.7 [34].

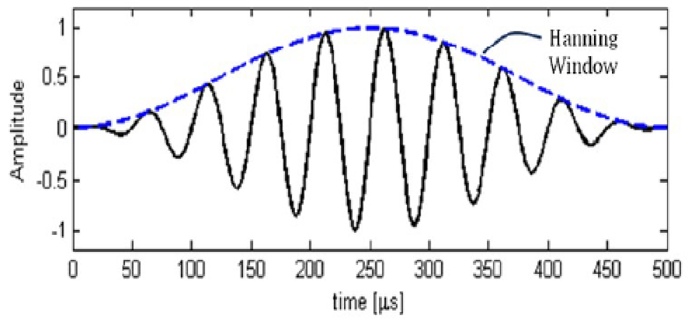


Figure 1.7: Windowing a sinusoidal signal with Hann window.

It is clear that some information is lost along the frame boundaries when windowing functions are applied to the signal. For this reason, overlapping the frames would be an additional improvement to the STFT by analyzing each part of the signal in more than one frame; therefore, lost information would be recovered within the next frame boundary as illustrated in Figure 1.8 [35, 33].

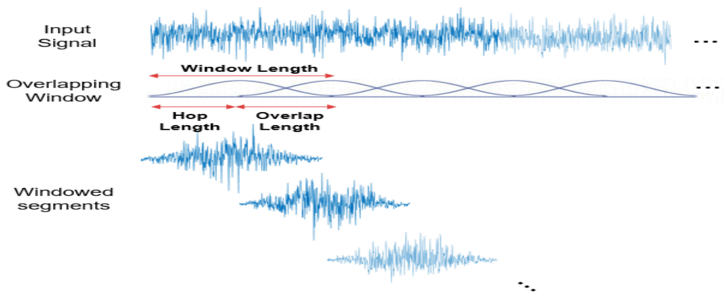


Figure 1.8: STFT procedure with overlap–add method.

1.3.6 Pre-Processing

Many real-world applications rely on machine learning. One of the important applications of machine learning is audio processing, which aims to extract meaningful information from audio [36, 37]. However, the signal would undergo some preprocessing before the machine learning process begins, with the goal of preparing it for the primary processing or improving certain features and characteristics of the signal for further analysis in order to maximize audio analysis performance in the later phases of the analysis system. In some cases, this is achieved by reducing the effects of noise or by emphasizing the target sounds of the signal [38].

1.3.7 Feature Extraction

Any machine learning system requires performing an in-depth analysis of the incoming audio signal, aiming at making the most of its particular characteristics. This analysis starts with the extraction of appropriate parameters of the audio signal that contain information about its most significant traits, a process that usually goes by the name of audio feature extraction. Logically, extracting the right features from an audio signal is a key issue in guaranteeing the success of machine learning applications. Indeed, the extracted features should provide a compact yet descriptive view of the signal, highlighting those signal characteristics that are most useful to accomplish the task at hand [39, 40]. These extracted features reflect the characteristics of the signal from a Temporal or physical point of view [41].

- **Temporal features**
  Temporal features, or time amplitude, are represented as amplitude fluctuations with time (waveform signal). Temporal audio features are extracted directly from raw audio signals with no preceding data. Representative instances of temporal features are zero-crossing rate, amplitude-based features, and power-based features. Such features normally suggest a simple tactic to investigate audio signals [42].

- **Physical features**
  Physical features are low-level signal parameters that can be calculated directly from the relative amplitudes of audio waveforms or from their short-time spectral values [43, 44].

  1. **SPECTROGRAMS**
     A spectrogram is a visual way of representing the signal strength as each individual frequency changes over time. In essence, spectrograms are two-dimensional graphs with a third dimension represented by colors. Time is displayed along the x-axis; frequency is displayed along the y-axis, which can also be thought of as pitch or tone,

with the lowest frequencies at the bottom and the highest frequencies at the top; and the amount of the amplitude or energy (loudness) of a particular frequency at a particular time is represented by the third dimension, level of gray. During regions of silence and at frequency regions where there is little energy, the spectrogram appears white, whereas, on the other hand, dark regions indicate areas of high energy [31, 45] . The Figure below shows the spectrogram of a dog barking recording.

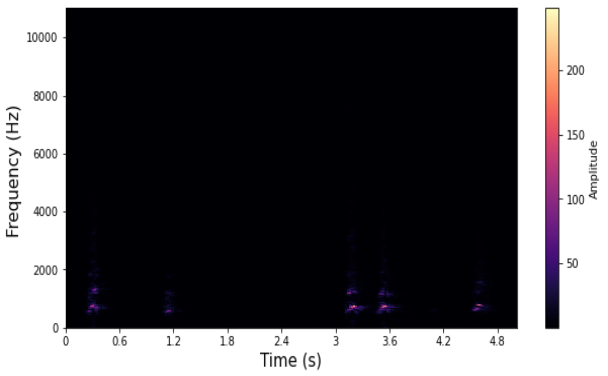


Figure 1.9: Spectrogram of a dog barking recording.

2. **MEL SPECTROGRAMS**

Audio data usually has complex features, so it is necessary to extract useful features to recognize the audio. The Mel-spectrogram is one of the efficient methods for audio processing. Therefore, each audio wave would be transformed into its own Mel spectrogram representation [46]. 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 for the human auditory system below 1000 Hz [47], and is related to Hertz by the following formula, where m represents Mels and f represents Hertz:

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

The inverse transform can be readily derived as:

$$f(m) = 700(10^{m/2595} - 1) \tag{1.6}$$

The Mel spectrogram is used to provide classifiers with sound information similar to what a human would perceive. The raw audio waveforms are passed through filter banks to obtain the Mel spectrogram [48, 49]. Figure 1.10 displays a mel frequency representation of a dog barking recording used earlier.

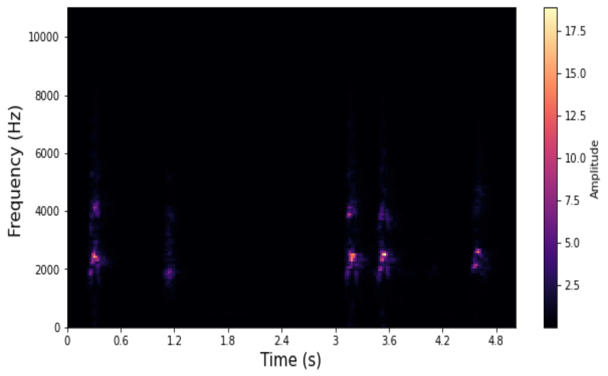


Figure 1.10: Mel-Spectrogram of a dog barking recording.

3. **LOG MEL SPECTROGRAMS**

The logarithmic spectrum is a much more intuitive representation compared to the mel spectrogram, but it is still considered another visual representation of mel Spectrograms where it is converted from power to decibels. The log-scaled spectrogram is not only more visual but also motivated by the human perception of loudness, which has a logarithmic relationship with the physical energy of sound [49, 50]. Despite the fact that learning a logarithmic function is a trivial task for neural networks, it can be difficult to implicitly learn an optimal nonlinear compression when it is embedded in a complicated task [50]. But still, Log-Mel Spectrograms are considered one of the best variants of the visual features that could be used as an input feature to convolutional neural networks nowadays [51]. Figure 1.11 illustrates a Log Mel-Spectrogram of a dog barking recording.

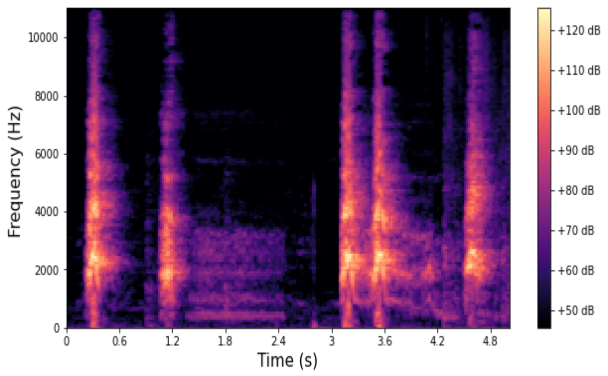


Figure 1.11: Log Mel-Spectrogram of a dog barking recording.

1.4 Machine Learning Essentials

Machine learning is the capability of a machine to imitate intelligent human behavior. Machine learning is based on designing models that are typically trained to recognize certain types of patterns in data and providing them with an algorithm that can be used to learn and understand in a way to automatically find patterns and structure in data by optimizing the parameters of the model. The purpose of it is to find good models that generalize well to yet unseen data, which we may care about in the future [52].

The field of machine learning has branched into several subfields dealing with different types of learning tasks [53]. Learning methods fall into three major categories: supervised learning, which uses labeled data to train models to classify new data or predict outcomes accurately [54]. Whereas in unsupervised learning, no labels are given to the learning algorithm, leaving it on its own to identify patterns and find structure in the input set [55]. A half-way between supervised and unsupervised learning that overcomes their drawbacks is Semi-Supervised Learning (SSL), where the data is partially labeled and the model is built with a few labeled patterns as a training set and treating the rest of the patterns as test data [56].

1.4.1 Classification

Classification is one of the most widely used tasks in machine learning and, in our case, audio analysis. It is used to identify the class of unseen instances on the basis of labeled training data. The sample (instance) is characterized by a feature vector x and its class label y, where classes can be called as labels or categories. For example, in an audio tagging task, the process would involve a model to classify sounds and predict the category of the output . In which we can distngwich two main types [57]:

- crisp label : where the label is unique and symbolic y appartien Y.
- probability distribution : the model would return a probability vector that represents the likelihood of the predicted class label and the expected class label over the k class labels, $\mu = [\mu_1, \mu_2, ..., \mu_k]^T \in [0, 1]^k$

In our Audio Tagging task, we will mainly focus on the probability distribution since we work with k classes. The classification process is correspondingly divided into two phases: training, when a classification model is built from the training set, and testing, when the model is evaluated on the test set. One of the major goals of a classification algorithm is to maximize and give a high predictive performance obtained by the classification model when classifying unseen test sets. One important measurement of performance is accuracy which describes how the model performs across all classes. We will discuss the most well-known evaluation metrics in one of the following sections.

Evaluating a machine learning model is as important as building it. We are creating models to perform on new unseen data. Hence, it is necessary to create a robust model. When it comes to classification models, deep learning models take the lead, and that is due to the efficiency of analysis that this latter has that we will discuss in the next section [58].

1.4.2 Deep Learning For Audio Tagging

Audio Tagging mainly aims at determining the presence of events in the acoustic scene [59]. It has been addressed with different deep learning approaches, such as deep convolutional neural networks (CNNs), that achieve state-of-the-art performance for the Audio Tagging task [60]. CNNs are a particular type of neural networks, which use the convolution operation in one or more layers for the learning process. These networks are inspired by the primal visual system, and are therefore extensively used with image and video inputs. CNNs are comprised of three types of layers. These are convolutional layers, pooling layers and fully-connected layers. When these layers are stacked, a CNN architecture has been formed. A simplified CNN architecture is illustrated in Figure 1.12 [61].

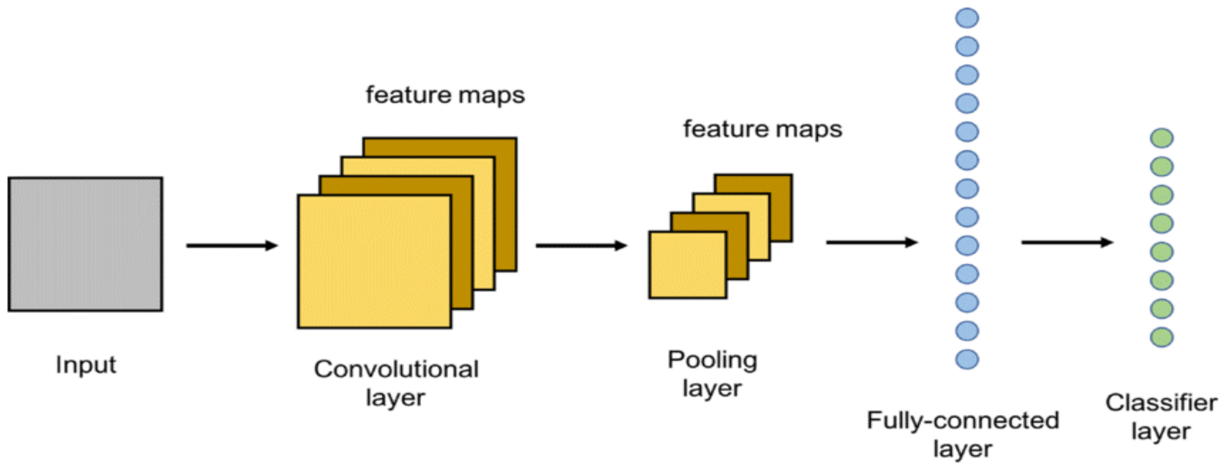


Figure 1.12: The overall architecture of the Convolutional Neural Network (CNN).

The basic functionality of the example CNN above can be broken down into four key areas [62].

- As found in other forms of artificial neural networks (ANN) the input layer will hold information of the data.
- The convolutional layer will determine the output of neurons of which are connected

to local regions of the input through the calculation of the scalar product between their weights and the region connected to the input volume. The rectified linear unit (commonly shortened to ReLu[1]activation) aims to apply an 'elementwise' activation function to the output of the activation produced by the previous layer.

- The pooling layer will then simply perform downsampling along the spatial dimensionality of the given input, further reducing the number of parameters within that activation.
- The fully-connected layers will then perform the same duties found in standard ANNs and attempt to produce class scores from the activations,to be used for classification in which which it outputs the probability of the input belonging to each of the classes. It is also suggested that ReLu may be used between these layers, as to improve performance.

Through this simple method of transformation, CNNs are able to transform the original input layer by layer using convolutional and downsampling techniques to produce class scores for classification purposes [62].

The reason why we used CNNs in our approach is due to the intrinsic nature of audio signals. CNNs are extensively used with images and, since the spectrum of the audio is an actual image of the signal, it is straightforward to see why CNNs are the best fit for such kinds of input, being able to exploit the adjacency properties of audio signals and recognize patterns in the spectrum images that can properly represent each of the classes taken into consideration [63]. But one drawback of traditional CNNs states that training the neural networks becomes more difficult with the increase in the number of added layers, and in some cases, the accuracy decreases as well. The solution to this drawback is using the residual neural networks that will be explained in one of the next chapters.

1.5 Statistical Tests

Given multiple learning algorithms, model evaluation aims at identifying which algorithm produces the most accurate classifiers. In fact, this is one of the most fundamental concerns in machine learning, and that's when statistical tests were introduced by many domain experts, who tried different statistical and logical techniques to decide whether the differences between the algorithms were real or random [64]. That's when Demsar [65], Garca et al [66], introduced several statistical tests. which are used in hypothesis testing where they can determine whether a predictor variable has a statistically significant relationship with an outcome variable or estimate the difference between two or more groups [65].

[1]ReLU stands for rectified linear unit which is a piecewise linear function that will output the input directly if it is positive, otherwise it will output zero.

1.5.1 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 [67]. This test is expressed as follows. We designate by d_i the difference between the performance scores of two techniques on N datasets. $i \in \{1, ..., N\}$. 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: (1.9) (1.10)

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2} \sum_{d_i=0} rank(d_i) \tag{1.7}$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2} \sum_{d_i=0} rank(d_i) \tag{1.8}$$

Notice that the ranks of $d_i = 0$ 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 values for T_w can be found in any textbook on general statistics [68], whereas for larger N, the statistics:

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{1.9}$$

follows the normal distribution with 1 mean and 0 variance. For instance, the hypothesis which states that two approaches perform equally is rejected if $z \preceq -1.96$ at a 5% significance level[65].

1.5.2 Kruskal-Wallis Test

The Kruskal-Wallis is a nonparametric statistical test that assesses the differences among three or more independently sampled groups. All the data are pooled and ranked from smallest (1) to largest (N), then the sums of ranks in each subgroup are added up, and the probability is calculated. The statistic H is:

$$H = \frac{12}{N(N+1)} \sum \frac{R_i^2}{n_i^2} - 3(N+1) \tag{1.10}$$

where N is the total number, n_i is the number in the i-th group, and R_i is the total sum of ranks in the i-th group. The value of H is tested against the chi-square distribution for $k-1$ degrees of freedom, where k is the number of groups. If there are tied ranks a correction is used but makes very little difference [69].

1.5.3 Bonferroni-Dunn Test

The Bonferroni-Dunn test shows that the power of the post-hoc test is much greater when all techniques are compared only to a control algorithm and not between themselves. The Bonferroni-Dunn test controls the family-wise error rate by dividing α by the number of comparisons made $(k-1)$. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference (CD).

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

where critical values q_α are based on the Studentized range statistic divided by $\sqrt{2}$.

Therefore an alternative way to compute the same test is to calculate CD but using the critical values for $\alpha/(k-1)$ [65].

1.6 Conclusion

In this chapter, we covered the fundamental basics of sound and audio signal presentation that play an important role in our thesis. We have presented numerous types of feature extraction techniques since they are needed as input for the learning stage. Although one particular problem consists of the shortage of audio data. The goal of the next chapter is to present a solution that involves creating an audio tagging system based on semi-supervised learning approaches.

Chapter 2

SEMI-SUPERVISED AUDIO TAGGING

2.1 Introduction

Data collection is one of the most difficult processes in machine learning, especially for producing an Audio Tagging system where the challenge lies in using only a small amount of supervised data. Therefore, we introduce the semi-supervised learning approach, which is concerned with using labeled as well as unlabeled data. SSL aims to understand how pairing labeled and unlabeled data impacts the learning behavior and design algorithms that take advantage of such a combination. In this chapter, we present the semi-supervised paradigm, explaining some popular semi-supervised learning methods and highlighting the underlying assumptions that SSL methods should rely on.

2.1.1 Motivation

Countless tasks have been solved using machine learning algorithms in a variety of fields. However, the different tasks require the use of various machine learning techniques. On one hand, there is supervised learning that helps to solve many types of real-world computation problems because its methods are more accurate and reliable since they use labeled data, whereas the process of data annotation (labeling the dataset) is a time-consuming and laborious task when done manually as it requires humans to review each training example before assigning a label to it. Meanwhile, in the case of unsupervised learning, the machine takes unlabeled training data as an input, where one of its advantages is that it extracts the essential information from the data and barely needs human interaction. However, it is a very time-consuming process where the learning phase might take a lot of time as it analyzes all possibilities. Also, the model would be learning from raw data without any prior knowledge, which wouldn't serve our audio tagging task. Therefore, a middle ground exists that consists of semi-supervised learning that uses both labeled and unlabeled data to create an Audio Tagging system. The process would start with training a model that would be used to assign labels to the unlabeled data. The unlabeled data would be added to the labeled set when its confidence is sufficiently large. The main research question in using SSL is how much unlabeled data is required to have an efficient model with accurate results for an audio tagging system [24] .

2.2 Main Assumptions in Semi Supervised Learning

SSL algorithms only work under some assumptions about the structure of the data they need to hold. The ability to generalize from a finite training set to an infinite number of potentially untested cases is not possible without such assumptions. The main assumptions in SSL are:

2.2.1 The Smoothness Assumption

If two points $x_1,\ x_2$ in a high-density region are close, then so should be their corresponding outputs $y_1,\ y_2$ [70]. This means that if two inputs are of the same class and belong to the same cluster, which is a high-density region of the input space, then their corresponding outputs need to be close. The inverse also holds true; if the two points are separated by a low-density region, the outputs must be far from each other. This assumption covers a classification task, which is the case for our Audio Tagging system [71].

2.2.2 Cluster Assumption

Consider the possibility that the points of each class tend to form a cluster. Then the unlabeled data could aid in finding the boundary of each cluster and using the labeled points to assign a class to each cluster. If points are in the same cluster, they are likely to be of the same class. The cluster assumption can easily be seen as a special case of the smoothness assumption, given that clusters are usually described as groups of points connected by short curves that only pass through high-density areas [70].

2.2.3 The Low-Density Assumption

The low-density assumption states that a classifier's decision boundary should preferably pass through low-density input space areas. The assumption is defined over $p\,(x)$, the input data's true distribution. When only a few samples from this distribution are considered, this essentially suggests that the decision boundary should lie in a region where few data points are observed. Therefore, the low-density assumption is closely related to the smoothness assumption; in fact, it can be considered the counterpart of the smoothness assumption for the underlying data distribution [72].

2.2.4 The Manifold Assumption

The (high-dimensional) data lies (roughly) on a low-dimensional manifold. In high-dimensional spaces, where the volume grows exponentially with the number of dimensions, it can be quite hard to estimate the true data distribution considering it is represented in Euclidean space. The observed data points in the high-dimensional input space are usually concentrated along lower-dimensional substructures. These substructures are known as manifolds. In particular, considering a 3-dimensional input space where many points lie along the surface of a sphere, the data is considered to lie on a 2-dimensional manifold. For semi-supervised learning, the manifold assumption holds that (a) the input space is composed of multiple lower-dimensional manifolds on which all data points lie and (b) data points lying on the same manifold have the same label. Therefore, by determining the data points that lie on a certain manifold, the class of the unlabeled data can be inferred from the labeled data that lies on the same manifold.

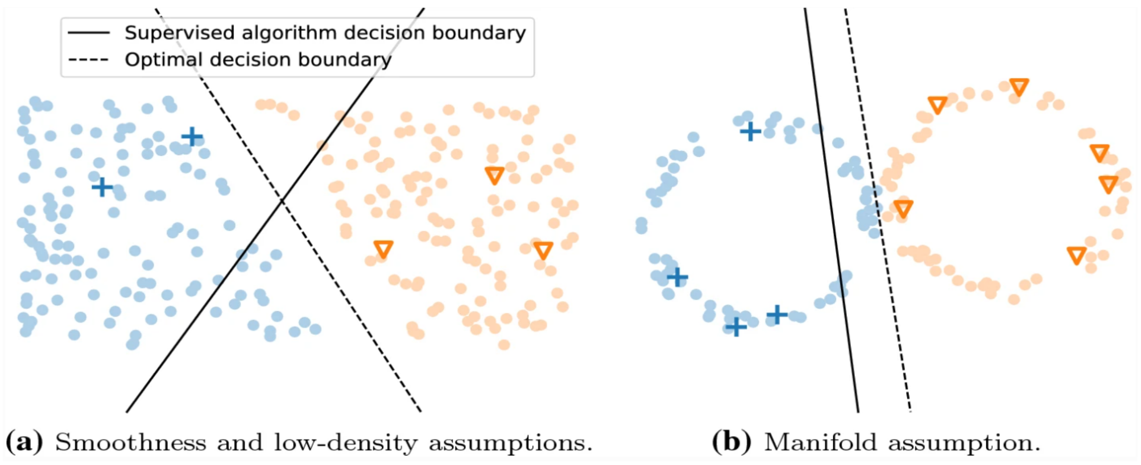


Figure 2.1: Illustrations of (a) Smoothness and low-density assumptions (b) Manifold assumption, and Cluster assumption depicted as the colors.

The Figure above [72] illustrates all the explained assumptions. The cluster assumption is represented by the different colors; each cluster has a different color where the dots of the same color, e.g. orange, 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 assumptions [71, 72].

2.3 Semi-Supervised Learning Approaches

Xiangli Yang et al. [73] classified the semi-supervised learning approaches into 5 categories, i.e., generative methods, consistency regularization, graph-based methods, pseudo-labeling, and hybrid methods. This section summarizes some of the well-known semi-supervised techniques that have been shown to work efficiently under audio tagging and in audio analysis in general. Hybrid methods consist of integrating two or more machine learning and/or soft computing technologies for better performance and optimum results[74]. While consistency regularization aims to encourage the prediction of the network to be similar in the vicinity of the observed training samples [75]. Generative methods are a powerful way of automatically discovering and learning the regularities or patterns in an input data in such a way that the model can be used to generate or output new instances [76]. Pseudo-labeling-based methods generate pseudo-labels for unlabeled samples with a model trained on labeled data [72, 77]. We will introduce some of the algorithms that have effectively achieved remarkable results in the audio analysis field.

2.3.1 Hybrid methods

Hybrid approaches combine distinct single techniques to generate a method with greater flexibility and capacity than single methods [74] . As is the case when generating a semi-supervised hybrid model that would be composed of an unsupervised learner (cluster) to preprocess the training data and a supervised learner (classifier) to learn the clustering result or vice versa [78]. In addition, a learning principle known as "mixup" is introduced in such hybrid methods. It is a convex combination of paired samples and their associated labels that may be thought of as a basic data augmentation strategy. Formally, Mixup constructs virtual training examples that would be used in the mixmatch algorithm [73]. The most well-known algorithms are Interpolation Consistency Training (ICT), MixMatch, FixMatch, ReMixMatch and DivideMix. We will give a brief explanation of MixMatch since it is the most commonly used algorithm for audio tagging tasks.

- **MixMatch**
  MixMatch [79] (MM) is an SSL approach that uses entropy minimization and standard regularization, namely pseudo-labeling, mixup, and weak data augmentation, to leverage the unlabeled data and provide better generalization capabilities. The different steps are detailed in the following paragraphs. During the learning phase, each minibatch is composed of labeled x_s and unlabeled x_u samples in equivalent proportions. The first step consists of applying an augmentation to the labeled part of the mini-batch and k augmentations to the non-labeled part. In the second step, pseudo-labels y_u are generated for the non-labeled files using the model's prediction averaged on these k variants as

shown in Eq 2.1 , where $x'_{u,i}$ denotes the i-th variant of an unlabeled augmented file.

$$\hat{Y}_u = \frac{1}{k}\sum_{i=1}^{k} f\left(x'_{u,i}\right) \tag{2.1}$$

For encouraging the model to produce confident predictions, a post-processing step is necessary to decrease the output's entropy. This process is called "sharpening" by the method authors, and it is defined as:

$$sharpen\,(p,T)_i := p_i^{1/T} \Big/ \sum_{j=1}^{|p|} p_i^{1/T} \tag{2.2}$$

The sharpen function is applied on to the pseudo-labels $p = \hat{Y}_u$. The parameter T, called Temperature, controls the strength of the sharpen function. When T tends towards zero, the entropy of the distribution produced is lowered. Finally, the labeled and unlabeled augmented samples are concatenated and shuffled into a W set then used as a pool of training samples used by the asymmetric mixup function [80].

2.3.2 Consistency regularization

One of the most widely used semi-supervised learning approaches in audio analysis is consistency regularization which is based on the manifold assumption or the smoothness assumption, and describes a category of methods that the realistic perturbations of the data points should not change the output of the model. Consequently, consistency regularization can be regarded to find a smooth manifold on which the dataset lies by leveraging the unlabeled data [73]. Consistency regularization employs various algorithms such as Ladder Network, Temporal Ensembling, Mean Teacher, VAT, Dual Student, SWA, UDA.

- **Mean Teacher**

  The MT algorithm is designed for semi-supervised consistency regularization tasks. The asset of this method is that it uses the Exponential Moving Average (EMA) of the model parameters instead of predictions. The key idea is to have two models called "Student" f and "Teacher" g. The student model is a regular model, and the teacher model has the same architecture as the student model, but its weights are set using an exponential moving average of the student model's weights.

  Figure 2.2 [80] illustrates that 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), in which two cost functions play an important role while backpropagating, in which they are classification cost and consistency cost.

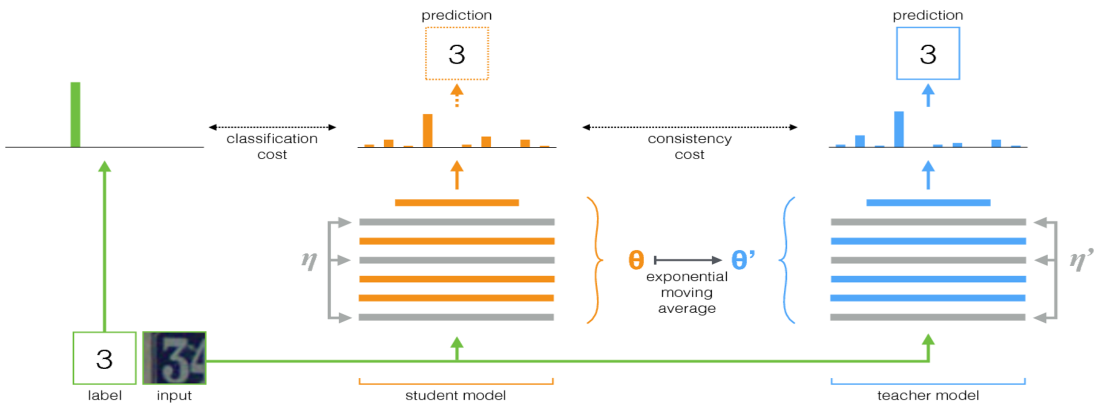


Figure 2.2: The Mean Teacher technique.

The classification $cost\,(C\,(\theta))$ is calculated as cross-entropy between label predicted by student model and the original label, where the consistency cost $cost\,(J\,(\theta))$ is the mean squared difference between the predicted outputs of the student (weights θ and noise η) and teacher model (weights θ' and noise η'). Noise is one of the important factors that play a crucial role in adding robustness to the model in which it would fool models by not being biased towards a particular target and also can perform well while predicting unseen data. The mathematical declaration of consistency $cost$ is as follows:

$$J\left(\theta\right)=E_{x},\eta^{'},\eta\left[||f\left(x,\theta^{'},\eta^{'}\right)-f\left(x,\theta,\eta\right)||^{2}\right] \tag{2.3}$$

- (x,θ,η) is the prediction of the student model
- $\left(x,\theta^{'},\eta^{'}\right)$ is the prediction of the teacher model

The final loss function is the sum of the supervised loss function and the consistency cost weighted by a factor λ which controls its influence.

$$O\left(\theta\right)=C\left(\theta\right)+\lambda J\left(\theta\right) \tag{2.4}$$

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 that are assigned at every step, and the proportion of weights assigned is controlled by smoothing coefficient hyperparameter (α). While assigning weights, the teacher model holds its previous weights in the (α) proportion and $(1-\alpha)$ portion of student weights.

$$\theta_{t'}=\alpha\theta_{t'-1}+\left(1-\alpha\right)\theta_{t'} \tag{2.5}$$

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. It 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 model performs better than the student model. However, the convergence of the teacher model depends on epoch, batch size, train data size, and smoothing coefficient hyperparameter α. Therefore parameter adjustments are required to get better results [81].

2.3.3 Generative Methods

Generative methods, also known as generative paradigms, have achieved tremendous success in just a few years. All types of generative models aim at learning the true data distribution of the training set to generate new data points with some variations. Two of the most commonly used and efficient approaches are Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN) [70, 82].

- **Generative Adversarial Networks (GAN)**
  Generative adversarial networks (GANs) have become a research focus in artificial intelligence. They have been widely studied due to their enormous prospects for applications, including image and vision computing, speech and language processing, etc. GANs comprise a generator and a discriminator, both trained under the adversarial learning idea. The goal of GANs is to estimate the potential distribution of real data samples and generate new samples from that distribution. The generator tries to capture the potential distribution of real samples and generates new data samples. The discriminator is often a binary classifier, discriminating real samples from generated samples as accurately as possible [76].

  Figure 2.3 [83] illustrates the architecture of a generative adversarial network. Suppose the samples from the training data have their own distribution of features. The task of the generative model is to try to simulate the features of the real data and generate fake samples with random input noise as close to the real thing as possible. The task of the discriminative model is to estimate the probability of the input, which contains samples both from real data and generated fake data from the generator, being real. In other words, the generator attempts to deceive the discriminator [84].

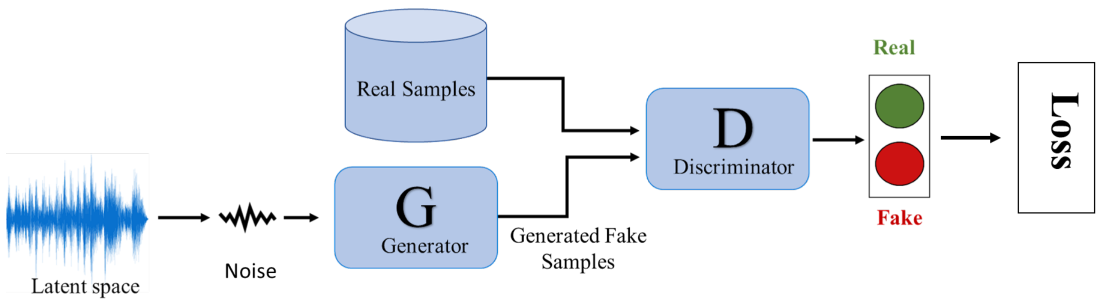


Figure 2.3: Architecture of the GAN.

The main idea of GAN comes from the Nash equilibrium in game theory. It assumes two game participants: one generator and one discriminator. The generator aims to learn the distribution of real data, while the discriminator aims to correctly determine whether the input data is from the real data or from the generator. In order to win the game, the two participants need to continuously optimize themselves to improve their generation ability and discrimination ability, respectively [76].

2.3.4 Pseudo-Labeling-based method

Pseudo-Labeling-based methods select unlabeled samples with high confidence as training targets; this can be considered as a form of entropy minimization, which minimizes the density of data points at the decision boundaries. One of the benefits of pseudo-labeling over consistency regularization is that it does not require augmentations and can be used in a wide range of domains. The goal of pseudo-labeling methods is to generate pseudo-labels for unlabeled samples with a model trained on labeled data. (i.e., created from the predictions of a trained neural network). Pseudo-labeling based methods have two main parts: disagreement-based models and self-training models [72, 77].

2.3.4.1 Self-Training Models

Self-training algorithm is one of the simplest general algorithms in SSl based methods. It leverages the model's own confident predictions to produce the pseudo labels for unlabeled data. In other words, it can add more training data by using existing labeled data to predict the labels of unlabeled data. It chooses the most confident named entity recognition predictions from the unlabeled data as the additional targets to boost the performance [72].

- **Pseudo Labeling**

  Pseudo-labeling is one of the most simple and efficient self-training models. It aims at using the labeled data model to predict labels for unlabelled data. This method follows some basic steps, as shown in Figure 2.4 [85].

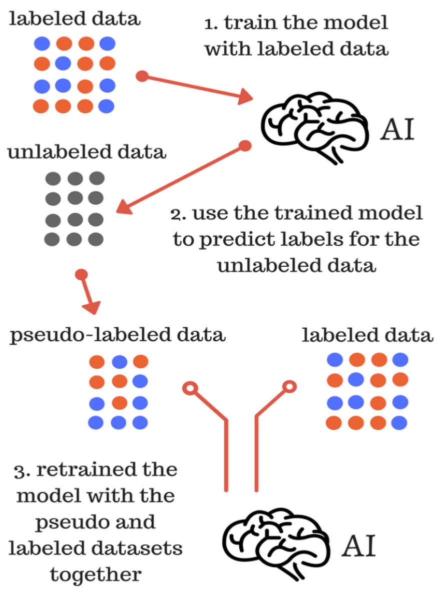

labeled data
1. train the model with labeled data
AI
unlabeled data
2. use the trained model to predict labels for the unlabeled data
pseudo-labeled data
labeled data
3. retrained the model with the pseudo and labeled datasets together
AI

Figure 2.4: Schematic diagram of pseudo-labeling steps.

  The process begins by training a supervised model on labeled data with the use of cross entropy loss. Then the trained model would predict labels for the unlabeled data. After that, the model would be retrained on both the labeled data and the pseudo labeled data (i.e., guessed labels), and then the supervised classifier would be retrained on both the original labeled data and the newly obtained pseudo labeled data (most confident predictions) [72, 85].

2.3.4.2 Disagreement-Based Models

The idea of disagreement-based SSL is to train multiple learners for the task and exploit the disagreement during the learning process. In such model architectures , two or three different networks are trained simultaneously and classify unlabeled data for each other. Disagreement-based methods differ in whether the data has a single view, i.e., Tri-Net or Co-training for multiview data [86, 87].

- **Deep Co-Training**

  The fundamental principle of co-training is based on the assumption that two independent views on a training dataset are allowed to train two models separately. The two models are then used to make predictions on the non-labeled data subset. The most confident predictions are selected and added to the labeled subset, and this process will iterate for a certain amount of time. Deep Co-training (DCT) is an adaptation of Co-training (CT) in the context of deep learning. Instead of relying on views of the data that are different, DCT makes use of adversarial examples to ensure the independence of the "view" presented to the models. We will cover this section in detail in the next chapter [72].

2.4 Conclusion

In this chapter, we have presented an overview of the field of semi-supervised learning, addressing the issues of supervised and unsupervised learning approaches and covering the most important methods from the semi-supervised learning paradigms with a combination of other dominant methods. In the next chapter, we will explore the Deep Co-Training Technique and adversarial examples in further detail.

Chapter 3

DEEP CO-TRAINING

3.1 Introduction

In the previous chapter, we have addressed the importance of semi-supervised learning and a few of its techniques and briefly introduced the deep co training (DCT) algorithm. Since our main interest revolves around the DCT algorithm, this chapter will cover its fundamentals as follows: in section 3.2 we will explain the SSL algorithm co-training where the DCT was adapted from. In the following section, we will explore DCT in greater detail, covering the different aspects and types of adversarial examples, and finally in section 3.6 we will analyze the empirical and theoretical findings related to the DCT algorithm.

3.2 Co-training

Co-training is a self-training extension that includes many supervised classifiers. In Co-training, two or more supervised classifiers are iteratively trained on the labeled data, adding their most confident predictions to the labeled data set of the other supervised classifiers in each iteration. It's crucial that the base learners' predictions are not too strongly correlated for Co-training to be successful. If they are, their ability to share useful information with one another is limited. This requirement is frequently referred to as the "diversity criterion". Zhou et al. [88] provided a survey of semi-supervised learning methods relying on multiple base learners. They jointly refer to these methods as disagreement-based methods, referring to the observation that co-training approaches exploit disagreements between multiple learners; they exchange information through unlabeled data, for which different learners predict different labels.

The Co-training algorithm relies on two main assumptions to succeed: (1) each individual subset of features should be sufficient to obtain good predictions on the given data set, and

(2) Given the class label, features should be conditionally independent. The first assumption states that if one of the two feature subsets is insufficient to form good predictions, a classifier based on that set can never contribute positively to the overall performance of the combined approach. The second assumption is related to the diversity criterion: if the feature subsets are conditionally independent given the class label, the predictions of the individual classifiers are unlikely to be strongly correlated.

In practice, the second assumption is generally not satisfied even if a natural split of features exists, it is unlikely that information contained in one view provides no information about the other view when conditioned on the class label [72].

3.3 Multi-view Co-training

In an audio tagging task, we are provided with an audio dataset $D = S \cup U$ where audio files in S are labeled however in U are not. The goal is to build classifiers on the categories C in S using the data in D. The test data contains only the categories that appear in S. There has been extensive research done on learning models from supervised datasets, and the state-of-the-art methods are deep convolutional networks [89]. The core problem is how to use the unlabeled U to help learning on S.

Co-Training assumes that each data x in D has two views, i.e. x is given as $x = (v_1, v_2)$, and each view v_i is sufficient for learning an effective model. Let X be the distribution that D is drawn from. Co-Training assumes that f_1 and f_2 trained on view v_1 and v_2 respectively have consistent predictions on X, i.e., $f(x) = f_1(v_1) = f_2(v_2)$ $\forall x = (v_1, v_2) \sim X$ (Co-Training Assumption) (1) .

Based on this assumption, Co-Training proposes a dual-view self-training algorithm where two independent views on a training dataset are available to train two models separately. Ideally, the two views are conditionally independent given the class. The two models are then used to make predictions on the non-labeled data subset. The most confident predictions are selected and added to the labeled subset for a specific number of iterations.

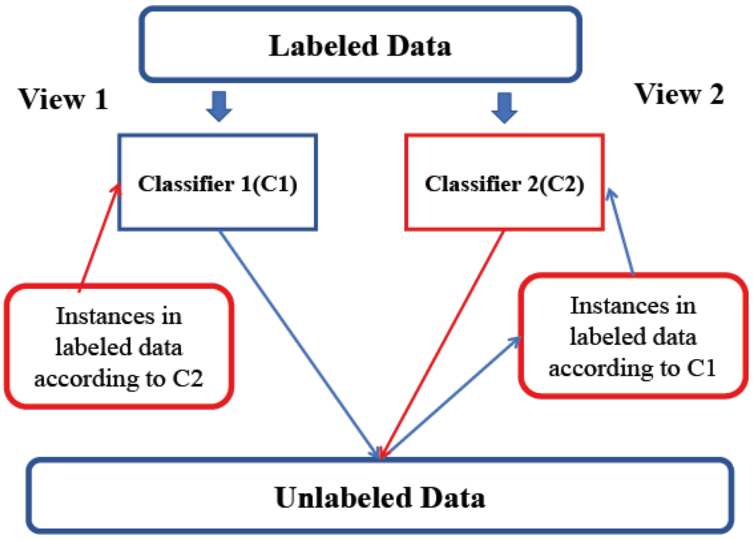

Labeled Data
View 1
View 2
Classifier 1(C1)
Classifier 2(C2)
Instances in labeled data according to C2
Instances in labeled data according to C1
Unlabeled Data

Figure 3.1: Representation of Co-training algorithm.

Given the superior performance of deep neural networks on audio analysis, it is interesting to extend the Co-training framework to apply deep learning to semi-supervised audio recognition and audio tagging. A simple implementation is to train two neural networks simultaneously on D . But this method suffers from a critical drawback: there is no guarantee that the views provided by the two networks will give different and complementary information about each data point. Although, Co-training is only beneficial if the two views are different. After all, training two identical networks serves no purpose. Moreover, the Co-training assumption pushes the two models to make similar predictions on both S and U, which can lead to collapsed neural networks. Therefore, For the Co-training framework to take advantage of deep learning, there must be a force that pushes networks apart to balance the co-training assumption [87].

3.4 Deep Co-Training

Deep co-training (DCT) has been recently proposed by Qiao and colleagues [87]. DCT is an adaptation of Co-Training (CT) in the context of deep learning without the drawbacks discussed above. Specifically, we model the co-training assumption by minimizing the expected loss between the predictions of the two networks on U. To avoid the neural networks from collapsing into each other and encouraging them to be different, DCT makes use of adversarial examples to ensure the independence of the "view" presented to the models. Therefore, we impose the view difference constraint formulated by Eq. 3.1 by training each network to be resistant to the adversarial examples of the other.

$$\exists X^{'} : f_1(v_1) \neq f_2(v_2), \forall x = (v_1, v_2) \sim X^{'} \tag{3.1}$$

The result of the training is that each network can maintain its predictions unaffected on the examples that the other network fails on. In other words, the two networks provide different and complementary information about the data because they are trained not to make errors at the same time on the adversarial examples for them.

The goal for building an Audio Tagging system based on DCT, is by starting with the dual-view case where we are interested in Co-training two deep neural networks. Therefore we use S and U to denote the labeled and the unlabeled dataset. Let $D = S \cup U$ denote all the provided data. Let $v_1(x)$ and $v_2(x)$ denote the two views of data x which are convolutional representations of x before the final fully-connected layer $f_i(.)$ that classifies $v_i(x)$ to one of the categories in S.
On the supervised dataset S, we use the standard cross entropy loss which measures the performance of a model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label [90].

$$L_{sup}(x, y) = H(y, f_1(v_1(x))) + H(y, f_2(v_2(x))) \tag{3.2}$$

The standard supervised classification loss function.
as eq 3.2 shows for any data (x, y) in S where y is the label for x and $H(p, q)$ is the cross entropy between distribution p (target) and q (predicted).

The two classifiers are expected to provide consistent and similar predictions on both the labeled and unlabeled data. Co-Training assumes that on the distribution X where x is drawn from, $f_1(v_1(x))$ and $f_2(v_2(x))$ agree on their predictions. In other words, the two classifiers $p_1(x) = f_1(v_1(x))$ and $p_2(x) = f_2(v_2(x))$ are expected to provide consistent and similar predictions on unlabeled data U. To encourage this behavior, we use a natural measure of similarity, the Jensen-Shannon divergence between $p_1(x)$ and $p_2(x)$, that quantifies how distinguishable $p_1(x)$ and $p_2(x)$,are from each other. Eq.3.3 gives the JS analytical expression.

$$L_{cot}(x) = H(\frac{1}{2}(p_1(x) + p_2(x))) - \frac{1}{2}(H(p_1(x)) + H(p_2(x))) \tag{3.3}$$

The Jensen-Shannon divergence between $p_1(x)$ and $p_2(x)$.

Where $x \in U$ and $H(p)$ is the entropy of p. Training neural networks based on the Co-training assumption minimizes the expected loss E $[L_{cot}]$ on the unlabeled set U. As for the labeled set S, minimizing loss L_{sup} already encourages them to have close predictions on S since they are trained with labels; therefore, minimizing L_{cot} on S is unnecessary, and we only implement it on U (i.e., not on S).

For Co-training to be successful, the two views must differ and provide complementary information about each data x. But minimizing Eq. 3.2 and 3.3 only encourages the neural networks to output the same predictions on $D = S \cup U$. Therefore, it is necessary to encourage the networks to be different and complementary. To achieve this, we create another set of data D' where $p_1(x) \neq p_2(x)$, $\forall x \in D'$, which we will generate by adversarial examples. Since Co-Training assumes that $p_1(x) = p_2(x)$, $\forall x \in D$, we know that $D \cap D' = \varnothing$. But D is all the data we have; therefore, D' must be built up by a generative method. We consider a simple form of generative method $g(x)$ which takes data x from D to build D', i.e. $D' = \{g(x) | x \in D\}$. For any $x \in D$, we want $g(x) - x$ to be small so that $g(x)$ also looks like the original data. But when $g(x) - x$ is small, it is very possible that $p_1(g(x)) = p_1(x)$ and $p_2(g(x)) = p_2(x)$. Since Co-Training assumes $p_1(x) = p_2(x)$, $\forall x \in D$ and we want $p_1(g(x)) \neq p_2(g(x))$, when $p_1(g(x)) = p_1(x)$, it follows that $p_2(g(x)) \neq p_2(x)$. These considerations imply that $g(x)$ is an adversarial example of p_2 that fools the network p_2 but not the network p_1. Therefore, in order to prevent the deep networks from collapsing into each other, we propose to train the network p_1 (or p_2) to be resistant to the adversarial examples $g_2(x)$ of p_2 (or $g_1(x)$ of p_1) by minimizing the cross entropy between $p_2(x)$ and $p_1(g_2(x))$ (or between $p_1(x)$ and $p_2(g_1(x))$, i.e.,

$$L_{dif}(x) = H(p_1(x), p_2(g_1(x))) + H(p_2(x), p_1(g_2(x))) \tag{3.4}$$

To summarize the Co-Training with the view difference constraint in a sentence, we want the models to have the same predictions on D but make different errors when they are exposed to adversarial attacks. By minimizing Eq. dif on D, we encourage the models to generate complementary representations, each is resistant to the adversarial examples of the other.

In Deep Co-Training, the objective function is of the form :

$$L = E_{(x,y) \in S} L_{sup}(x, y) + \lambda_{cot} E_{x \in \mu} L_{cot}(x) + \lambda_{dif} E_{x \in D} L_{dif}(x) \tag{3.5}$$

which linearly combines Eq. 3.3, Eq. 3.4 and Eq. 3.5 with hyperparameters λ_{cot} and λ_{dif} where it calculated as follow

$$\lambda(epoch) = \lambda_{max}(1 - e^{-5 \times (1-(epoch/w_1))}) \tag{3.6}$$

- w_1 represents the warm up length

Finally Compute the gradients with respect to L by backpropagation and update the parameters of p_1 and p_2 using gradient descent.

3.5 Adversarial attacks on neural networks

As neural networks have found their way from labs to the real world, the security and integrity of the applications pose great concern. Recent work has demonstrated that deep neural networks are vulnerable to adversarial examples. Adversarial attacks can craftily manipulate inputs by adding perturbations that are imperceptible to the human eye and are almost indistinguishable from natural data, leading trained models to produce incorrect results [91, 92]. These perturbations are far from considered noises since noise is a random or uncontrolled interference. On the other hand, perturbations are controllable and measurable and aren't detectable by standard noise filters. The security of any machine learning model is measured concerning the adversarial goals and capabilities. We begin with the identification of the attack surface of systems built on deep learning models in order to identify potential vulnerabilities caused by the adversary.

3.5.1 The Attack Surface

A system built based on a deep Learning task can be viewed as a generalized data processing pipeline. For illustration, consider a generic pipeline of an automated vehicle system as shown Figure 3.2 [92].

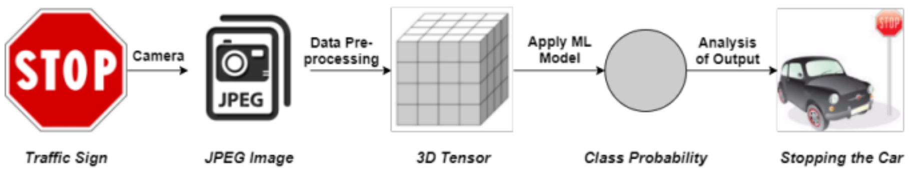


Figure 3.2: Generic pipeline of an Automated Vehicle System.

The system collects sensor inputs (images using camera) from which model features are extracted and used by the models. Following that a decision is then made based on the output (probability of stop sign), and takes a proper action (stopping the car). In this case an adversary can attempt to manipulate either the collection or the processing of data to corrupt the target model, hence tampering the original output. The fundamental attack scenarios set by the attack surface are:

- Evasion Attack: which is the most popular type of attack in the adversarial setting. The adversary tries to evade the system by adjusting and setting malicious samples through the testing phase. This setting does not imply any impact over the training data. The Figure below [92] illustrates the process of an evasion attack.

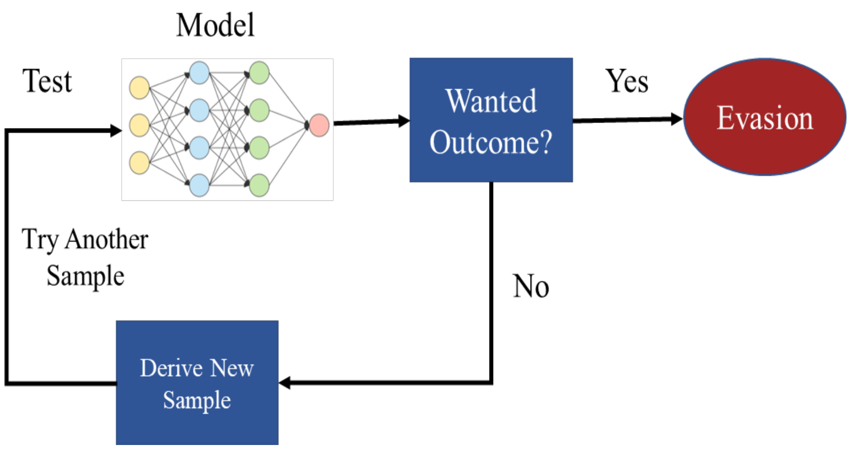


Figure 3.3: A schematic representation of an evasion attack.

- Poisoning Attack: this type of attack takes place on the training time of the model. An adversary tries to corrupt the training data by injecting malicious designed samples to compromise the whole learning process eventually. The Figure below [92] illustrates the process of a poisoning attack.

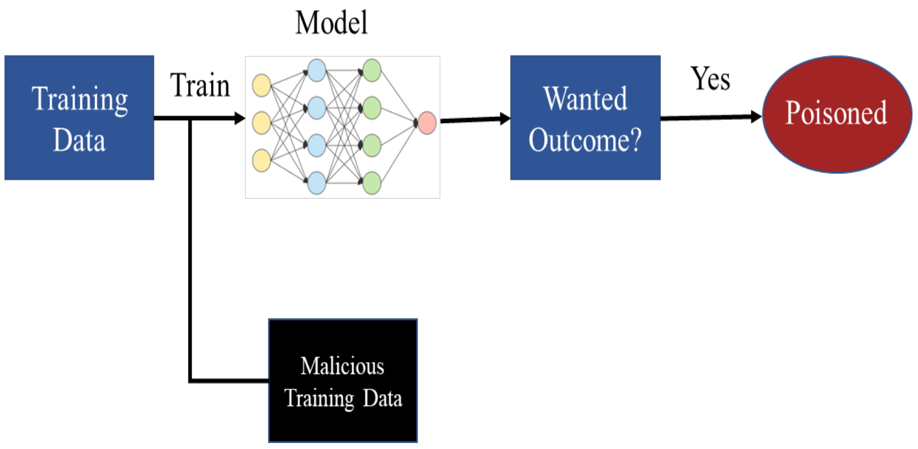


Figure 3.4: A schematic representation of a poisoning attack.

3.5.2 Adversarial Goals

An adversary tries to provide an input x to a classification system that results in an incorrect output classification. The objective of the adversary is concluded from the incorrectness of the model. Based on the influence on the classifier output integrity, the adversarial goals can be classified as follows:

1. Confidence Reduction: The adversary tries to minimize the confidence of prediction for the target model. For example, an audio of a "siren" can be predicted with a lower confidence having a lesser probability of class belongingness.

2. Misclassification: The adversary tries to change the output classification of an input example to any class distinct from the original class. For example, the audio of a "siren" will be predicted as any other class different from a "siren".

3. Targeted Misclassification: The adversary tries to make inputs that force the output of the classification model to be a specific target class. For example, any input audio to the classification model will be predicted as a class of a "siren".

4. Source/Target Misclassification: The adversary tries to force the output of classification for a specific input to be a particular target class. For example, an audio of a 'siren' will be predicted as a 'helicopter' by the classification model.

3.5.3 The Adversarial Capabilities

The term adversarial capabilities refer to the amount of information available to an adversary about the system.

- **Training Phase Capabilities**

  Attacks during training time attempt to impact or corrupt the model directly by changing the dataset used for training. The most straightforward and weakest attack on the training phase is by directly accessing partial or full training data. There are three attack strategies for altering the model based on the adversarial capabilities.

  1. Data Injection: The adversary does not have access to the training data nor the learning algorithm but has the ability to insert adversarial samples into the training dataset, therefore corrupting the target model.
  2. Data Modification: The adversary does not have access to the learning algorithm but has total access to the training data. By corrupting the training data directly by affecting and manipulating the data before it is used for training the target model.
  3. Logic Corruption: The adversary is capable of intervening with the learning algorithm. These attacks are called "logic corruption". Apparently, it becomes very difficult to purposefully implement counter strategies against these adversaries who can change the learning logic, thereby controlling the model itself.

- **Testing Phase Capabilities**

  Adversarial attacks at the testing time do not manipulate the targeted model but rather force it to make incorrect outputs. The efficiency of such attacks is determined fundamentally by the amount of information available to the adversary about the model.

Attacks are classified by knowledge where they are classified by black box and white box attacks.

1. **Black box attack**

   Black-box adversarial attacks describe scenarios in which the attacker does not have access to the network. In other words, the attacker does not have any information about the parameters of the model nor does he have access to the training stage and mainly uses information about the settings or past inputs to study and analyze the vulnerability of the model. Black Box attacks can be classified into the following categories:

   (a) Non-Adaptive Black-Box Attack: For a target model (f), a non-adaptive black-box adversary can only get access to the target model's training data distribution (µ). The adversary then selects a procedure train for a model architecture f and trains a local model over samples from the data distribution (µ) to approximate the model learned by the target classifier. The adversary crafts adversarial examples on the local model f using white-box attack strategies and then applies these crafted inputs to the target model to force mis-classifications. The Figure below shows this kind of attacks[93].

   (b) Adaptive Black-Box Attack: For a target model (f), an adaptive black-box adversary does not have any information concerning the training procedure but can access the target model as an oracle. The adversary issues adaptive oracle queries to the target model and labels a carefully selected dataset, i.e., for any arbitrarily chosen x the adversary gets its label y by querying the target model f . The adversary then selects a procedure train and model architecture f to train a surrogate model over tuples (x,y) obtained from querying the target model. The surrogate model then produces adversarial samples by pursuing white-box attack techniques for forcing the target model to misclassify malicious data.

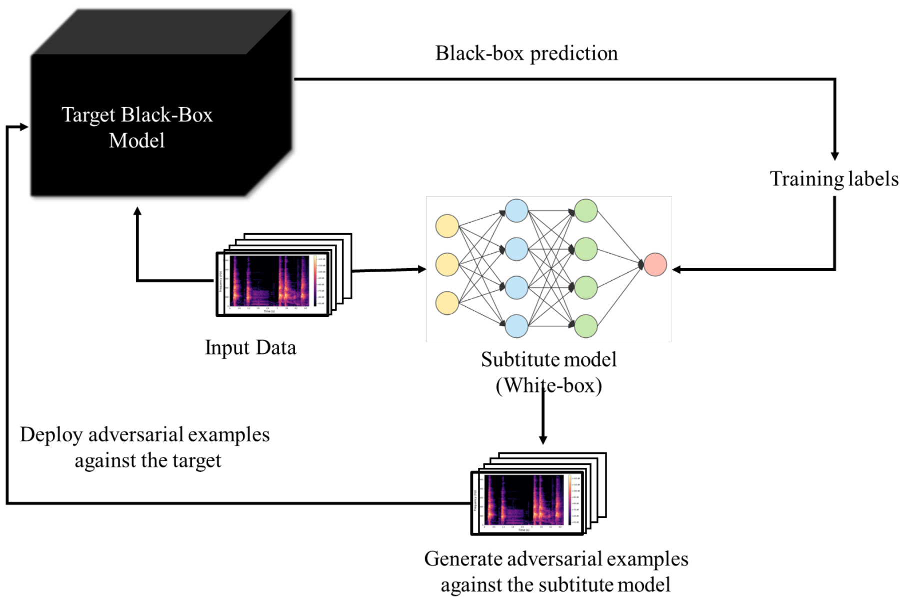

Target Black-Box Model
Black-box prediction
Training labels
Input Data
Subtitute model (White-box)
Deploy adversarial examples against the target
Generate adversarial examples against the subtitute model

Figure 3.5: Block-box attack representation.

The primary objective of a black-box adversary is to train a local model with the data distribution (µ) in the case of a non-adaptive attack and with a carefully selected dataset by querying the target model in the case of an adaptive attack.

2. **White box attack**
In a white-box attack on a deep learning model, an adversary has the entire knowledge about the model (f) used for classification. The attacker has information about the algorithm used in training (e.g., gradient-descent optimization), can access the training data distribution (µ) and is also familiar with the parameters (θ) of the fully trained model architecture. The adversary utilizes available information to recognize the feature space where the model may be vulnerable. Then the model is employed by altering an input using an adversarial example crafting method. To explore how adversaries craft adversarial samples in a white-box setup. Papernot et al [93],introduced a general framework which builds on the attack approaches discussed in recent literature. The framework is divided into two phases: a) direction sensitivity estimation and b) perturbation selection, as shown in Figure 3.6 [93].

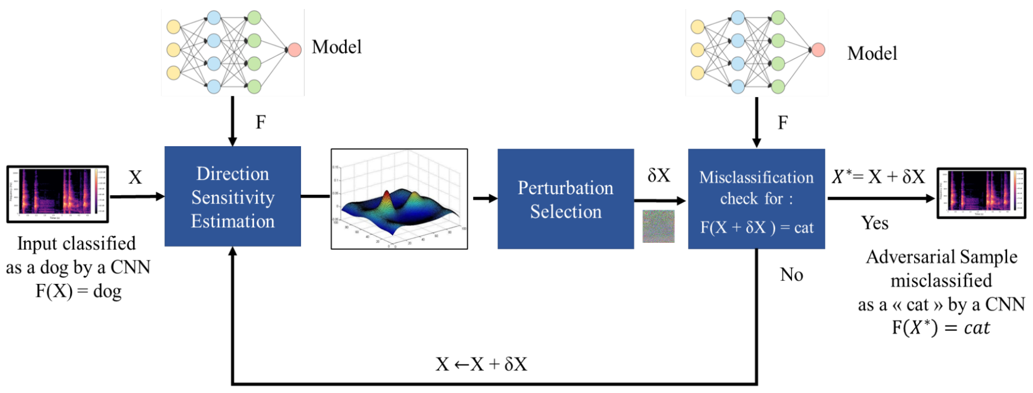

Model
F
X
Direction Sensitivity Estimation
Input classified as a dog by a CNN F(X) = dog
Perturbation Selection
δX
Model
F
Misclassification check for : F(X + δX) = cat
X^*= X + δX
Yes
No
Adversarial Sample misclassified as a « cat » by a CNN F(X^*) = cat
X ←X + δX

Figure 3.6: adversarial crafting framework in awhite-box attack.

The Figure proposes an adversarial example crafting process for an audio classification using a CNN, which can be generalized for any supervised learning algorithm. Suppose X is as Log Mel spectrogram of a dog recording, and F is a trained CNN classification model. The objective of an adversary is to craft a malicious example $X* = X + \delta X$ by adding a perturbation δX with the sample X, so that $F(X*) = Y*$ where $Y* \neq F(X)$ is the target output, which depends on the objective of the adversary.

The adversary use a two-step process for the adversarial sample crafting, which is discussed below:

(a) Direction Sensitivity Estimation: The adversary estimates the sensitivity of a class change to each input feature by identifying directions in the data manifold around sample X in which the model F , learned by the CNN, is most sensitive to result in a class change.

(b) Perturbation Selection: The adversary then uses the knowledge of sensitive information to choose a perturbation δX between the input dimensions in order to get an adversarial perturbation that is most effective. Both the steps are repeated by substituting X with $X + \delta X$ before the start of each new iteration, until the adversarial purpose is satisfied by the perturbed sample. The point to be remembered in this situation is that the total perturbation used for crafting the adversarial sample from a valid example needs to be as small as possible. As mentioned before, this is necessary for the adversarial samples to remain undetected in human eyes.

In the direction sensitivity estimation step, the adversary considers a sample X, an n-dimensional input vector. The goal here is to find those dimensions of X which will produce an expected adversarial performance with the smallest chosen perturbation. This can be accomplished by changing the input components of X and evaluating the sensitivity of the trained CNN model F to these alterations. Developing the knowledge of the model sensitivity can be achieved in several ways [92]. Some of the well-known techniques mentioned in the recent literature are discussed below:

- **L-BFGS :**

  Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) is a non-linear gradient-based numerical optimization algorithm. However, since Szegedy et al [94] defined the problem as an optimization problem which can be solved using L-BFGS; the attack is now often referred to as the L-BFGS attack. where this latter was one of the very first adversarial attack approaches. It is designed to fool models for different tasks. Its end goal is to find a perceptually minimal input perturbation arg $min_r \parallel r \parallel_2$ i.e., $r = x^{'} - x$, within bounds of the input space, that is adversarial, i.e., $\hat{y}(x^{'}) \neq y$. They used the L-BFGS method to find an adversarial sample $x^{'}$ that satisfies the following box-constrained optimization problem [92, 95]:

  $$min\, c \parallel r \parallel_2 + L(x^{'}, t), x^{'} \in [0, 1] \tag{3.7}$$

  Where elements of $x^{'}$are normalized to $[0, 1]$, $L(x^{'}, t)$ is the true loss function of the targeted model, and t is the target misclassification label. Considering an image related task m is presented as $m = hw$ (height*width) if the image is gray scaled and $m = 3hw$ if the image is colored. Since this objective does not guarantee that $x^{'}$ will be adversarial for any specific value of $c \succ 0$, the above optimization process is iterated for increasingly large values of c via line search until an adversary is found. Because of this computational intensive linear searching method for optimal c, the L-BFGS attack is time consuming and impractical [95, 96].

  L-BFGS produces adversaries that are perceptually alike to the original input x. Moreover, the key advantage of crafting the adversarial example generation process as a general optimization issue is that it allows for flexibility in folding additional standards into the objective function. For instance, one may select to employ perceptual similarity metrics where instead of computing distances in the image space, we compute distances between image features extracted by deep neural networks depending on the requirements of a given application domain[95].

- **Fast gradient sign method:**

  The Fast Gradient Sign Method (FGSM) combines a white-box approach with a misclassification goal where it tricks a neural network model into making wrong predictions. This method works by using the gradients of the neural network to create an adversarial example. Given a simple linear classifier $\omega^T x$ where ω is the weight vector , Let's denote the perturbation as η. This latter causes a change that is non-perceivable and ostensibly innocuous to the human eye, yet destructive and adverse enough for the classifier to the extent that its predictions are no longer accurate.

$$\hat{x} = x + \eta \tag{3.8}$$

where we apply a constraint such that

$$\parallel \eta \parallel_\infty \leq \epsilon \tag{3.9}$$

The infinity norm is defined as

$$\parallel A \parallel_\infty = max_{1 \leq i \leq m} \sum_{j=1}^{n} \mid a_{ij} \mid \tag{3.10}$$

which means the largest absolute value of the element in the vector. In this context, it means that the largest magnitude of the element in η does not exceed the precision constraint ϵ . Consider the dot product between a weight vector ω and an adversarial example $\hat{x}$:

$$\omega^\top \hat{x} = \omega^\top x + \omega^\top \eta \tag{3.11}$$

Hence the addition of $\omega^\top \eta$ should not cause the model to behave any different in the absence of any perturbation.

The adversarial perturbation causes the activation to grow by $\omega^\top \eta$. We can maximize this increase subject to the max norm constraint on η by assigning $\eta = sign(w)$. If w has n dimensions and the average magnitude of an element of the weight vector is m, then the activation will grow by mn. Since $\parallel \eta \parallel_\infty$ does not grow with the dimensionality of the problem but the change in activation caused by perturbation by η can grow linearly with n, then for high dimensional problems, we can make many infinitesimal changes to the input that add up to one large change to the output.

The idea behind FGSM specifically is that instead of doing a typical gradient descent, we would do the opposite in order to maximize the loss, since confusing the model is the goal of an adversarial attack.

Let θ be the parameters of a model, x the input to the model, y the targets associated with x and $J(\theta, x, y)$ be the cost used to train the neural network. Add the gradient to its original input variable to create a perturbation. Where the required gradient can be computed efficiently using backpropagation. Mathematically, this can be expressed as follows:

$$\eta = \epsilon \cdot sign(\nabla_x J(w, x, y)) \tag{3.12}$$

Then, we can create an adversarial example via

$$\hat{x} = x + \epsilon \cdot sign(\nabla_x J(w, x, y)) \tag{3.13}$$

FGSM aims to encourage perceptual similarity between x and $\hat{x}$. Under the infinity norm constraint, the sign of the gradient vector maximizes the magnitude of the input perturbation, which consequently also amplifies the adversarial change in the model's output [95, 82].

- **Deep Fool**

  The DeepFool algorithm estimates the distance of an input instance x_0 to the closest decision boundary of a multi-class classifier. This result can be used both as a measure of the robustness of the model to attacks, and as a minimal adversarial perturbation direction. As motivation, the authors note that in order to have a binary linear classifier misclassify an input x_0, the label of x_0 needs to be projected orthogonally onto the decision boundary (which is simply a line). This can be analytically computed using the point-to-line distance formula. Iteratively, this process would be performed until the shortest distance was found from the input point to the decision boundary to give the total perturbation $\hat{r}$. This readily generalizes to a multi-class linear classifier, where the desired measure can be computed as the distance to the nearest of the decision boundary lines that are presented as a polyhedron formed by classifiers as shown in the green region in Figure 3.7 [97], and then projected x_0 onto that decision boundary and extends it further, thus misclassifying it with the minimal perturbation possible.

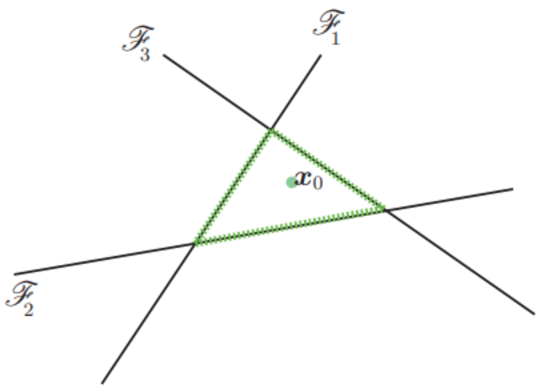
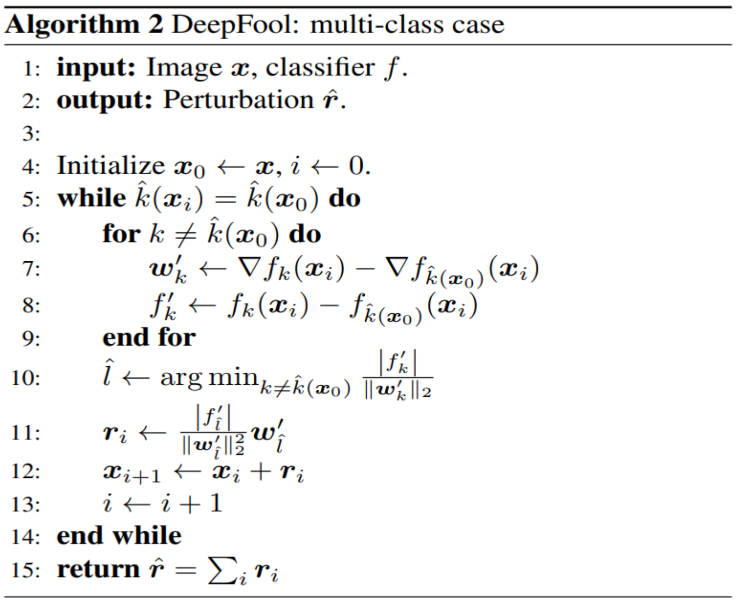


Figure 3.7: Hyperplanes are depicted in solid lines and the boundary of P is shown in green dotted line.

To illustrate furthermore this procedure we explore the associated algorithm illustrated below:



Algorithm 2 DeepFool: multi-class case

```
1: input: Image $\boldsymbol{x}$, classifier $f$.
2: output: Perturbation $\hat{\boldsymbol{r}}$.
3:
4: Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
5: while $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ do
6:     for $k \neq \hat{k}(\boldsymbol{x}_0)$ do
7:         $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
8:         $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
9:     end for
10:    $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2}$
11:    $\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{\|\boldsymbol{w}'_{\hat{l}}\|_2^2} \boldsymbol{w}'_{\hat{l}}$
12:    $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
13:    $i \leftarrow i + 1$
14: end while
15: return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$
```

Where x represents the input image and f the associated classifier, resulting in a minimum perturbation. We initialize the perturbed image with the original image and iterate the algorithme uptil the outputs are different. Then consider going through each of the k^{th} classes besides $\hat{k}(x_0)$ where we store the minimum difference between the original gradients and the gradients of each of these classes ω_k and the difference in the labels f_k. Afterward the inner loop stores the minimum $\omega_k^{'}$ and $f_k^{'}$ to calculate the closest hyperplane for the input x_0, Formally, $\hat{l}(x_0)$ can be computed as follows:

$$\hat{l}(x_0) = argmin_{k \neq \hat{k}(x_0)} \frac{\mid f_k(x_0) - f_{\hat{k}(x_0)}(x_0) \mid}{\parallel w_k - w_{\hat{k}(x_0)} \parallel_2} \tag{3.14}$$

We calculate the minimal perturbation vector with the use of $\hat{l}(x_0)$ where it can be computed as:

$$r_*(x_0) = \frac{\mid f_{\hat{l}(x_0)}(x_0) - f_{\hat{k}(x_0)}(x_0) \mid}{\parallel w_{\hat{l}(x_0)} - w_{\hat{k}(x_0)} \parallel_2^2} (w_{\hat{l}(x_0)} - w_{\hat{k}(x_0)}) \tag{3.15}$$

Adding the minimal perturbation to the image and determining if it has been misclassified, if not an iteration would occur until the output class is changed. Finally return the total perturbation, which is a sum over all the calculated perturbations. The size of the resulting perturbation can be interpreted as a measure of the model's robustness to adversarial attacks. DeepFool can compute this measure using a variety of different l_p distance metrics including $p = 2$ Euclidean norm and $p = \infty$ supremum norm. In practice, once an adversarial perturbation $\hat{r}$ is found it would be multiplied by a constant $1 + \eta$, with $\eta \gg 1$, where the adversarial example is nudged further beyond the decision boundary to guarantee misclassification [95, 97].

- **CW**

  Carlini and Wagner introduced a family of attacks for finding adversarial perturbations that minimize diverse similarity metrics: l_0, l_2, and l_∞ They relied on the initial formulation of adversarial examples and formally defined the problem of finding an adversarial instance for an image x as follows:

$$\begin{gathered} minimize\ \ D(x, x, +\delta) \\ such\,that\,C(x + \delta) = t\,.....\,constraint\,1 \\ x + \delta \in [0, 1]^n\,....\,constraint\,2 \end{gathered} \tag{3.16}$$

  Where x = input image, δ= perturbations, D = distance metric between the adversarial and the original image, C = Classifier function, n= dimensions, t = target class.

  The distance metric is usually specified in terms of Lp norms (l_0, l_2, l_∞). Constraint 1 makes sure that the image is indeed misclassified and constraint 2 makes sure that the adversarial image is valid i.e. it lies within the normalized dimensions of x. But since $C(x + \delta) = t$ is highly non-linear (the classifier is not a straightforward linear function). Carlini and wagner expresses constraint 1 in a different form as an objective function f such that when $C(x+\delta)=t$ is satisfied $f(x+\delta)\le 0$ is also satisfied.

The authors evaluated 7 different objective functions f (all of them are loss functions) and selected the best one among them that is given by:

$$f(x^{'}) = max(max_{i\neq t}\left\{Z(x^{'})_{(i)}\right\} - Z(x^{'})_{(t)}, -k) \tag{3.17}$$

Where $Z(x^{'})_{(i)}$ denotes the i^{th} component of the classifier's logits, t denotes the target label, and represents a tuning parameter that allows to steer the confidence of the function. Conceptually, this loss function minimizes the distance in logit values between class t and the second most-likely class. If t currently has the highest logit value, then the difference of the logits will be negative, and so the optimization will stop when this logit difference between t and the runner-up class exceeds . On the other hand, if t does not have the highest logit value, then minimizing $f(x^{'})$ brings the gap between the logits of the winning class and the target class closer together, i.e., either reducing the highest class prediction confidence and/or increasing the target class confidence. Finally, Equation 2 can be alternatively represented as:

$$minimize \parallel \delta \parallel_p + cf(x+\delta)\ subject\,to\ x+\delta \in [0,1]^n \tag{3.18}$$

Where c is a constant that controls both the effectiveness of the attack and the success rate of the attack. The attack is more effective when the adversarial instance is similar to the original image. The attack is accurate if it successfully misguides the model to classify the adversarial instance to the target class t. The tradeoff is measured by the magnitude of c. The authors experimentally found that the best way to choose value of c is use the smallest value of c using the binary search algorithm, for which the misclassification occurs ($f(x+\delta)\le$ occurs).

Now, expressing the formulation in terms of L_p norm instead as distance D, it becomes:

$$\begin{gathered} minimize \quad \parallel \delta \parallel_p + c\cdot f(x+\delta) \\ such\,that\ \ x+\delta\in[0,1]^n \end{gathered} \tag{3.19}$$

To ensure the modification yields a valid image, constraint 2 must have an upper and lower bound $0 \le x_i + \delta_i \le 1$ for all i. In the optimization literature, this is known as a “ box constraint ” Carlini evaluated 3 methods and select one method known as “ change of variables ” in which instead of optimizing over variable δ in constraint 2, they optimized over , which is given by:

$$\delta_i = \frac{1}{2}(\tanh(w_i + 1) - x_i \tag{3.20}$$

since $-1 \leq tanh(w_i) \leq 1$, it follows that $0 \leq x_i + \delta_i \leq 1$, so the solution will automatically be valid.
The final form of the optimization problem using L_2 distance metric [95, 98] is:

$$minimize \parallel \frac{1}{2}(\tanh(w) + 1) - x \parallel_2^2 + c \cdot f(\frac{1}{2}(\tanh(w) + 1) \tag{3.21}$$

with f defined as

$$f(x^{'}) = max(max\left\{Z(x^{'})_i : i \neq t\right\} - Z(x^{'})_t, -k) \tag{3.22}$$

3.6 Related works on DCT

Table 3.1 presents some literature works that have employed the Deep Co-training approach in both computer vision tasks; such as Image segmentation, image recognition and in audio-related tasks, such as audio tagging. These studies analyze the behavior and give a general overview of both the visual and audio fields. Since Deep Co-training is a recent adaptation of Co-training only a few studies have been conducted based on it.

For instance, In 2018 S Qiao et al [87], studied the problem of semi-supervised image recognition, which involves training a convolutional neural network (CNN) based on $\prod$ model. and a resent18 model using both labeled and unlabeled images .They applied the Deep Co-training algorithm by using the adversarial attack FGSM to generate adversarial examples. Their experiments have shown that this additional force has pushed their models away, was very helpful for training and improved accuracy significantly. The researchers have tested their method on SVHN, CIFAR-10/100 and ImageNet datasets, where they extend the dual-view DCT to a scalable multi-view DCT method where the hyperparameters for two views are also suitable for increased numbers of views. Their method outperforms the previous state-of-the-art methods by a large margin.

In 2019, the purpose of Peng, J et al. [99] was to improve the performance of semantic image segmentation in a semi-supervised setting where they trained a U-net model on both annotated data and non-annotated images to exchange information with each other. They conducted their work on three clinically-relevant benchmark datasets for medical image segmentation: Automated Cardiac Diagnosis Challenge (ACDC), Spinal Cord Gray Matter Challenge (SCGM) and Spleen sub-task dataset of the Medical Segmentation Decathlon Challenge. They used the Deep Co-training algorithm based on the FGSM attack to generate adversarial samples to enforce the diversity across models. In which their results outperformed recent approaches.

Recently, Cances, L. et al [81], adapted the Deep Co-training algorithm (DCT) to perform an audio tagging task while using an FGSM attack to improve the performance of their classifier. They conducted their experiments on three standard audio datasets: Environmental Sound Classification (ESC-10), UrbanSound8K (UBS8K), and Google Speech Commands (GSC). While using a fraction of the labeled data available, and the remaining data as unlabeled data, they would be transformed into a log mel spectrogram to be fed into their wideResnet28 model. In almost all configurations, DCT consistently outperformed another SSL approach called Mean Teacher (MT).

S Qiao et al. [87] have conducted their work on Imagenet, SVHN, CIFAR 10 and CIFAR 100 which are natural real world images, and have used the Deep-Co training algorithm to generate adversarial examples using FGSM with 0.02 of magnitude of perturbation to fool the models and since this latter have shown promising results, Peng, J et al. [99] tested the DCT algorithm on an image segmentation task in which it used ACDC, SCGM, and spleen datasets, consisting of different MRI and CT scans, and also used the FGSM adversarial attack with a 0.01 magnitude of perturbation to deceive the U-net model and achieved remarkable results. Since this SSL approach has been shown to achieve state-of-the-art results on image datasets while using a small limited amount of labeled data and SSL methods in general applied to audio data are still sparse, Cances, L. et al. [81], have tested audio datasets such as Urban8k, Environmental Sound Classification (ESC-10) and Google Speech Commands Datasets to determine whether the DCT method would perform as well as it did on an image dataset. They fooled the wide residual network using FGSM since it gave fast and promising results with a magnitude of perturbation of 0.02. This latter has shown the effectiveness of Deep Co-training applied to audio tagging, in which promising results were obtained.

Table 3.1: Summary of related work results.

Ref	Type Dataset	Dataset	Architecture	Performance
[87]	Image	Imagenet	Resnet18	Error rate = 22.73
	Image	SVHN	CNN	Error rate = 3.61
	Image	CIFAR 100	CNN	Error rate = 38.77
	Image	CIFAR 10	CNN	Error rate = 9.03
[99]	Image	ACDC	U-net	DSC[1]= 86%
	Image	SCGM	U-net	DSC =72.76%
	Image	Spleen	U-net	DSC = 91.81%
[81]	Audio	ESC-10	WideResnet28	Accuracy = 91.72%
	Audio	UBS8K	WideResnet28	Accuracy = 76.85%
	Audio	GSC	WideResnet28	Accuracy = 94.87%

3.7 Conclusion

Throughout this chapter, we have reviewed the importance of semi-supervised learning by explaining the classic Co-Training and highlighting the main feature of the Deep-Co training algorithm by using adversarial attacks on neural networks in order to obtain reliable and robust audio tagging systems. Furthermore, we summarized some empirical and theoretical findings based on the DCT approach. In the next part, we will first introduce the experimental setup and describe the overall pipeline that we followed in conducting the thesis. Then, we will present and discuss the obtained results based on robust statistical tests.

[1]The Dice similarity coefficient (DSC) is a validation metric to evaluate the performance of automated probabilistic fractional segmentation of MR images, its range is between 0 and 1.

PART II: DESIGN AND EXPERIMENTATIONS

In this part we describe the methodology that we have followed for evaluating and comparing different Audio Tagging systems. It consists of two chapter. Chapter 4 covers the design to build the Audio Tagging system, whereas in Chapter 5 we analyze and discuss the results of our experiments.

Chapter 4

DESIGN OF A SEMI-SUPERVISED AUDIO TAGGING SYSTEM

4.1 Introduction

In this chapter, we present the full design used to carry out the objective of our thesis. In Section 4.2, we describe our implemented Audio Tagging system, providing the pre-processing step along with feature extraction techniques, an overview of both Resnet and WideResnet, along with their variation architectures that were used in training our systems,also the advrasarial attacks used and finally the evaluation measures used to measure the performance of the Audio Tagging system.

4.2 Audio Tagging System Description

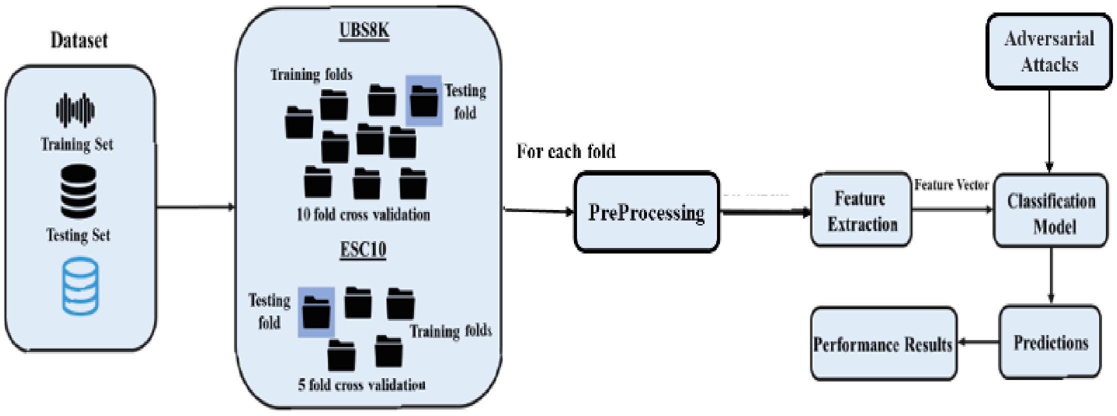


Figure 4.1: The overall Audio Tagging System pipeline.

The Figure above represents the overall pipeline of our Audio Tagging system that is able to recognize and identify a wide variety of sound events that occur in audio recordings (such as human sounds, domestic sounds, animals, tools, etc.). The process of Audio Tagging consists of two main stages: Feature Extraction and Classification. The feature extraction process includes transforming the audio signal into a relevant time-frequency representation. Next, a deep neural network model classifies the extracted features into a corresponding event label. Following that, the system is then evaluated through an evaluation procedure.

4.2.1 Pre-Processing

Audio is highly dimensional and contains redundant and often unnecessary information. Therefore, a preprocessing stage is highly needed in which audio is prepared and processed for machine learning algorithms in the audio processing phase of the overall system design. Pre-processing is applied to the audio signal before the machine learning process starts. The main goal of this step is to enhance certain characteristics and properties of the incoming signal in order to maximize audio analysis performance in the later phases of the analysis system. Prepossessing audio data includes tasks like resampling audio files to a consistent sample rate, removing regions of silence, and trimming audio to a consistent duration to make the audio files fairly equal.

During our thesis, we have converted the audio data into two channels when needed, for example, when some of the sound files are mono (i.e., 1 audio channel) while most of them are stereo (i.e., 2 audio channels). It is one necessary step since our model expects all items to have the same dimensions, and therefore we will duplicate the first channel of the mono files to create a stereo file. Another important point is standardizing the sampling rate, since sound files are sampled at different rates. Once again, we must standardize and convert all audio to the same sampling rate in order to have the same dimensions. Also, we need to resize all the audio samples to have the same length by either extending their duration by padding them with silence or by truncating them.

4.2.2 Feature Extraction

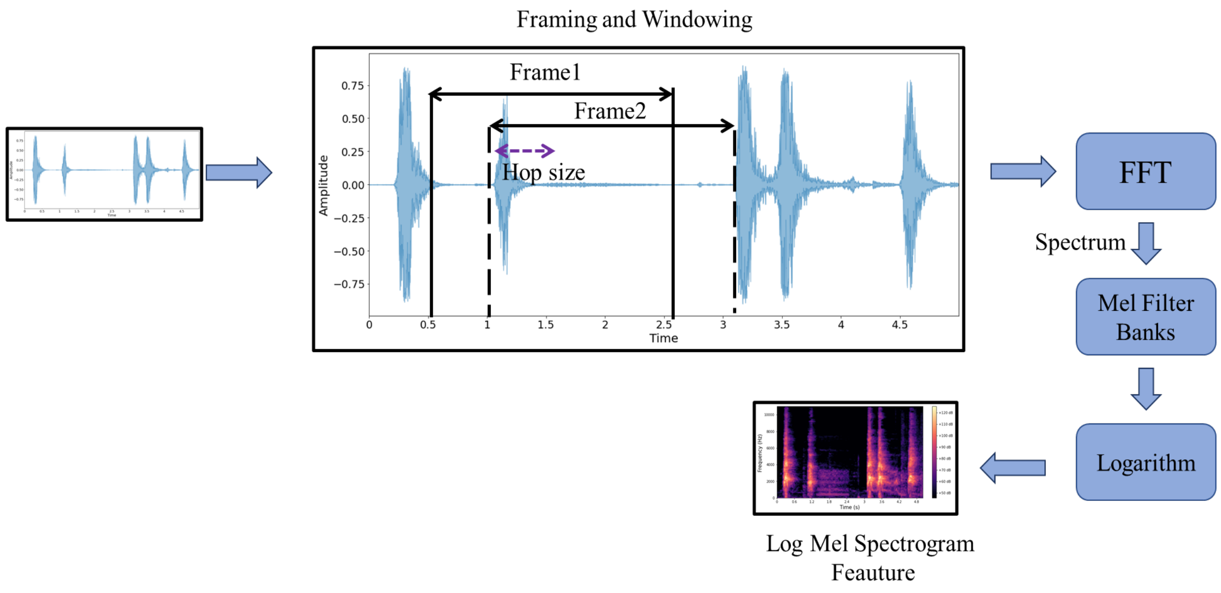


Figure 4.2: The overall process of feature extraction.

Feature extraction is a crucial process in which the audio signal is transformed from a waveform into representations that maximize the sound recognition performance [24] in our case a Log-mel spectrogram. The first step of feature extraction as Figure 4.3 shows, consists of converting an audio recording from an analog to a digital time signal. However, the time-domain representation of a sound signal is not easy to interpret. For that reason, using a frequency-domain representation is a must, where the digital signal is converted from the time domain to the frequency domain by the use of the Fourier transform, more precisely, the short time Fourier transform (STFT). This later has two techniques: framing, which divides the signal into several chunks in which the frames will overlap each other as they slide across the audio signal. After slicing the signal into frames, a Hamming windowing function is applied to smooth the edges of each chunk to avoid spectral leakage. Afterwards, applying triangular filters is a crucial step in which filters are applied on a Mel-scale to the power spectrum to extract frequency bands. The Mel-scale aims to imitate the non-linear human ear's perception of sound, specifically, the ability to discriminate lower frequencies from higher frequencies. After applying the filter bank to the power spectrum of the signal, we finally compute the logarithmic energy of each filter bank output to obtain the Log-mel spectrogram features that would be fed to the model.

4.2.3 Neural Network Architectures

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 the vanishing gradient problem that arises during backpropagation and is caused only in deeper networks. In this regard, residual networks (ResNet) can be helpful since they overcome this latter difficulty. The gradient norm of earlier layers decreased to zero and almost vanished as the training proceeded [100]. In ResNet, the yield of each residual layer is convolved with its input to be the input of the next layer. Let H(x), represents the residual mapping to build a residual learning block as shown in Figure 4.4 residual learning block [101].

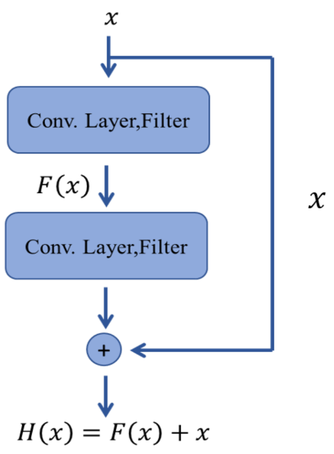


Figure 4.3: Building block of the Residual Neural Network

This ResNet block approximately calculates H(x) := F(x) + x . The formulation of F(x) + x is recognized by feedforward neural systems with “skip connections” also known as the shortcut connection which is the core of residual blocks. This latter combines the input and the output of the stacked layer through an identity mapping operation without any additional parameters. Therefore, the gradients can easily flow back, resulting in faster training and many more layers [100]. However, a slight problem appears within this approach. When the dimensions of F(x) are different from x, a projection method needs to be used to match the dimension which is done by adding 1×1 convolutional layers to the input. In such a case, the output is : H(x)=f(x)+w1.x given w1 is an additional parameter. Two main types of blocks are used in a ResNet, an identity block, which refers to the standard block used in ResNets and corresponds to the case where the input activation has the same dimension as the output activation, whereas a convolutional block is used when the input and output dimensions don't match up [102].

Three of the most well-known variations of ResNets based on the number of layers are namely: Resnet18, Resnet34, and Resnet50. Table 4.1 illustrates a summary of the output size at every layer and the dimension of the convolutional kernels at every point in the structure.

Table 4.1: Summary of ResNet variation architectures

Layer name	**Output Size**	**Resnet18**	**Resnet34**	**Resnet50**
Conv1	216 × 216	7×7, 64, stride 2		
Conv2 x	108 × 108	3×3 MaxPool, stride 2		
		$\begin{bmatrix} 3\times3 & 64 \\ 3\times3 & 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3 & 64 \\ 3\times3 & 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1 & 64 \\ 3\times3 & 64 \\ 1\times1 & 256 \end{bmatrix} \times 3$
Conv3 x	54 × 54	$\begin{bmatrix} 3\times3 & 128 \\ 3\times3 & 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3 & 128 \\ 3\times3 & 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1 & 128 \\ 3\times3 & 128 \\ 1\times1 & 512 \end{bmatrix} \times 4$
Conv4 x	27 × 27	$\begin{bmatrix} 3\times3 & 256 \\ 3\times3 & 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3 & 256 \\ 3\times3 & 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 & 256 \\ 3\times3 & 256 \\ 1\times1 & 1024 \end{bmatrix} \times 6$
Conv5 x	14 × 14	$\begin{bmatrix} 3\times3 & 512 \\ 3\times3 & 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3 & 512 \\ 3\times3 & 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1 & 512 \\ 3\times3 & 512 \\ 1\times1 & 2048 \end{bmatrix} \times 3$
1 × 1		average pool, 10 fc		
Params		11,181,642	21,289,802	23,528,522

Deep residual networks were shown to be able to scale up to thousands of layers and still improve performance. However, each fraction of a percent of improved accuracy costs nearly doubled the number of layers, which made it very slow to train. To tackle this problem, another architecture was proposed, in which we decrease the depth and increase the width of residual networks. We call the resulting network structures wide residual networks (WideResnets) [103]. This network architecture has proven that shallow networks with increased width are able to provide similar or better results than those obtained with very deep neural networks [104]. Figure 4.5 shows the building block of a wide residual network (WideResnets) [103].

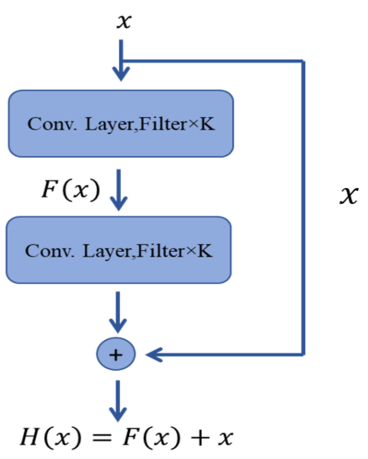


Figure 4.4: Building block of the Wide Residual Neural Network

Essentially, to increase the representational power of residual blocks, it is needed to add more convolutional layers per block, to widen the convolutional layers by adding more feature planes (maps) and to increase filter sizes in convolutional layers by applying two factors, deepening factor l and widening factor k, where l is the number of convolutions in a block and k multiplies the number of features in convolutional layers, thus the baseline "basic" block corresponds to l= 2, k = 1. We refer to original residual networks with k = 1 as "thin" and to networks with k > 1 as "wide". WideResnet architectures consists of an initial convolutional layer conv1 that is followed by 3 groups (each of size N) of residual blocks conv2, conv3 and conv4, followed by average pooling and a final classification layer [103].

To further explain the inside of the wide residual network, we take WRN28_2 model architecture as an example in which is illustrated in Figure 4.6, in which it has a depth of 28 deep layers and a width of 2. It consists of Conv1, which is based on a convolution, batch normalization (BN) that normalizes the input of the activation functions, and a max pooling operation with a stride of 2. Afterward, we can observe in the Figure below that WRN28_2 has three groups and each group is composed of four blocks (basic block), which includes two layers. A layer here refers to a convolution, a batch normalization, and a ReLU except the last operation of a block, which lacks the ReLU. The next step is to escalate from a block to a block of the next group. This involves increasing the stride by the down sampling of the volume through the network. In addition to that, we see another pattern repeating over the groups of WRN28_2 , where the first layer of each group is reducing the dimension, so we also need to resize the volume that goes through the skip connection, by applying 1×1 convolution(stride2) to ensure the volumes at this addition operation are the same size. The behavior is exactly the same for the following layer, changing only the dimensions of the incoming volumes.

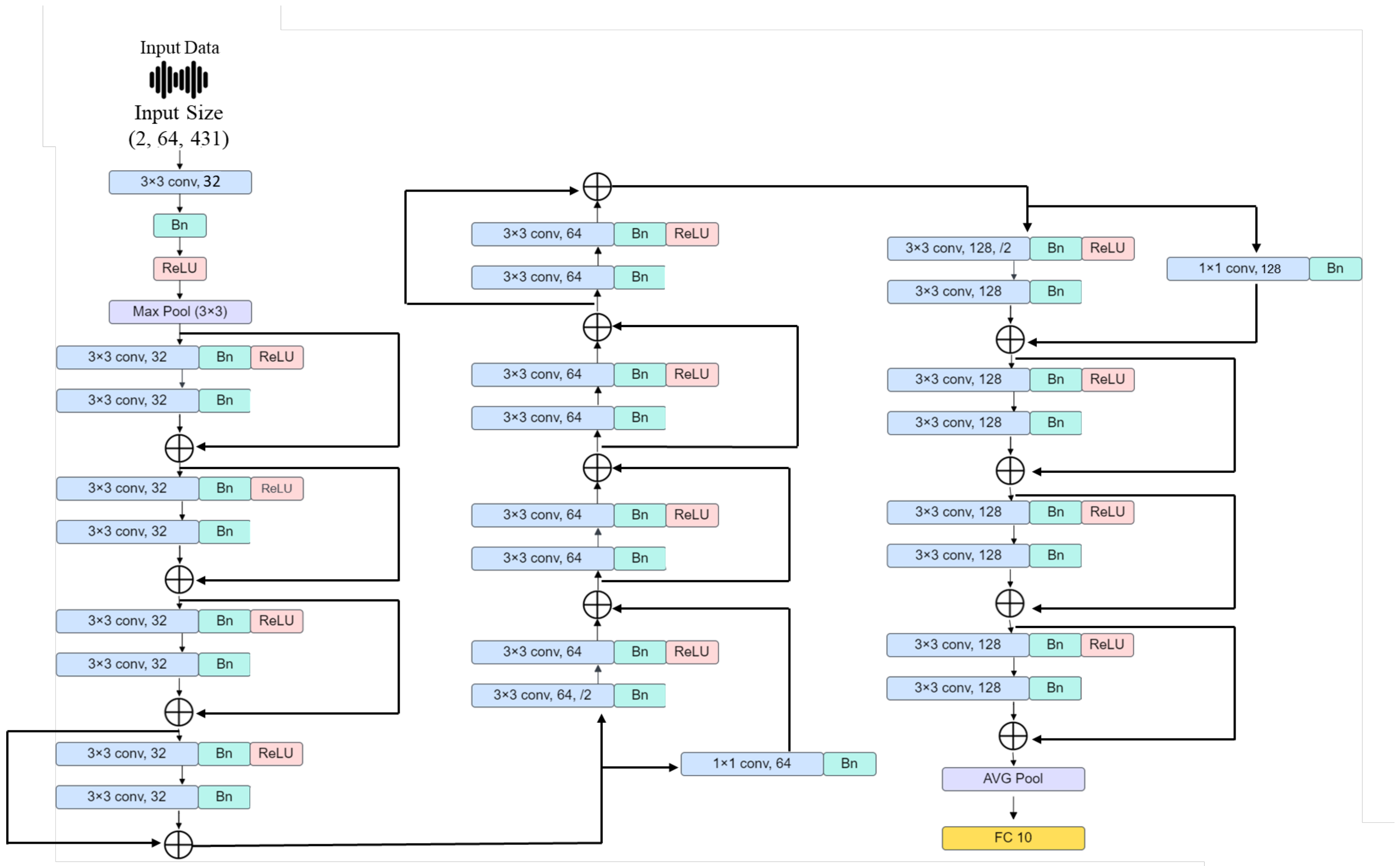


Figure 4.5: The WRN28_2 architecture.

Table 4.2 shows the WideResnets variation architectures used during our thesis.

Table 4.2: Summary of WRN variation architectures.

Layer name	Output Size	**Wideresnet28_2 (WRN2)**	**Wideresnet28_4 (WRN4)**	**Wideresnet28_8 (WRN8)**
Conv1	431 × 431	3×3, 32, stride 1		
Conv2 x	216 × 216	3×3 MaxPool, stride 2		
		$\begin{bmatrix}3\times3 & 32\\3\times3 & 32\end{bmatrix}\times 4$	$\begin{bmatrix}3\times3 & 64\\3\times3 & 64\end{bmatrix}\times 4$	$\begin{bmatrix}3\times3 & 128\\3\times3 & 128\end{bmatrix}\times 4$
Conv3 x	108 × 108	$\begin{bmatrix}3\times3 & 64\\3\times3 & 64\end{bmatrix}\times 4$	$\begin{bmatrix}3\times3 & 128\\3\times3 & 128\end{bmatrix}\times 4$	$\begin{bmatrix}3\times3 & 256\\3\times3 & 256\end{bmatrix}\times 4$
Conv4 x	54 × 54	$\begin{bmatrix}3\times3 & 128\\3\times3 & 128\end{bmatrix}\times 4$	$\begin{bmatrix}3\times3 & 256\\3\times3 & 256\end{bmatrix}\times 4$	$\begin{bmatrix}3\times3 & 512\\3\times3 & 512\end{bmatrix}\times 4$
1 × 1		average pool, 10 fc		
Params		1,472,554	5,877,834	23,486,602

4.2.4 Adversarial Attacks

Recently, a variety of deep learning tasks have relied on adversarial attacks to enhance their performance. For that reason, we have introduced the semi-supervised Deep Co-training technique that makes use of adversarial examples that would be generated using adversarial attacks. The latter offers a variety of attacks for a variety of purposes. We have conducted our work using four well-known attacks, namely:

- **L-BFGS :** Szegedy et al [94]. first introduced adversarial examples against deep neural networks in 2014. They showed that the generated adversarial examples could also be generalized to different models and different training datasets. Which is favorable to our case since we are dealing with audio-type data. The Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) is a non-linear gradient-based numerical optimization algorithm in which the L-BFGS attack aims to generate adversarial examples that look very similar to their real counterparts according to a distance metric, but one that causes a classifier to misclassify it. In other words, it aims to find a perturbation r with the goal of making the classifier misclassify the perturbed input (x + r) as class l, where the loss function used here is the cross-entropy loss. To find a suitable minimum constant c, L-BFGS Attack calculates approximate values of adversarial examples by line-searching c > 0 until an adversary is found.

$$min\, c \parallel r \parallel_2 + L(x^{'}, t), x^{'} \in [0, 1] \tag{4.1}$$

- **FGSM :** L-BFGS Attack used an expensive linear search method to find the optimal value, which was time-consuming and impractical. Goodfellow et al [82]. proposed a fast method called the Fast Gradient Sign Method to generate adversarial examples . This attack is remarkably powerful and yet intuitive in that it searches for the direction in which the loss function increases fastest for a target deep learning model. FGSM is an example of a whitebox attack because the attacker needs to know the model's architecture and parameters to perform backpropagation. It is designed to attack neural networks by leveraging the way they learn gradients. The idea is that rather than working to minimize the loss by adjusting the weights based on the back-propagated gradients, the attack adjusts the input data to maximize the loss based on the same back-propagated gradients. There is no guarantee that the generated adversarial examples by this method are similar to their real counterparts. Practically, one needs to make a tradeoff between small perturbations that are visually similar to the original input and whether the model actually misclassifies the perturbed input.

- **CW :** Carlini Wagner [98] extended the L-BFGS attack by modifying the objective function instead of using the standard cross-entropy loss; they use another loss function that was mentioned in chapter 3 in equation 3.17. The intuition for that objective function is to optimize for the distance between the target class and the most-likely class. If the target class currently has the highest logit value, then the difference between the logit values will be negative, and the optimization will stop when the logit difference between the target class and the runner-up class is at most k. In other words, k controls the desired confidence for the adversarial example (e.g., when k is small, the adversarial example generated will be a low-confidence adversarial example). On the other hand, if the target class does not have the highest logit, then minimizing the loss function brings the gap between the highest class' logit and the target class' logit closer together, i.e., either reducing the highest class' confidence or increasing the target class' confidence. Carlini Wagner actually proposed three different attacks under three different perceptual similarity metrics. We have used the L2 norm in view of the fact that it is more beneficial to our case.

- **Deep Fool :** Moosavi-Dezfooli et al. [97] propose the Deep Fool algorithm, which searches for the shortest distance to cross the decision boundary using an iterative linear approximation of the classifier and orthogonal projection of the sample point onto it, which sets a small vector to perturb input data and push them out of the classification boundary gradually until misclassification occurs. This untargeted attack generates adversarial examples with a smaller perturbation compared with L-BFGS and FGSM.

4.2.5 Evaluation

Evaluation is usually framed as estimating the performance of a system under test when confronted with new data. The system output is then compared to the reference to calculate measures of its performance. We can measure accuracy to reflect how often the system correctly classifies or detects a sound, or we can measure error rates to reflect how often the system makes mistakes. By using the same data and the same methodology to evaluate different systems (perhaps in different places and/or at different times), a fair and direct comparison can be made of system performance [29].

4.2.5.1 Cross Validation

Cross validation is a statistical method used to estimate the effectiveness of a machine learning model on a limited data sample. In other words, it is a procedure used to evaluate a model by learning a hypothesis from a training set and measuring its generalization error on a test set [105]. This approach requires a large amount of data in order to obtain a reliable estimate of the generalization error, which is rare in most situations [106]. Noumoureus resampling techniques such as k-fold cross validation and leave-one-out have been introduced [105].

The K Fold cross validation is an iterative approach during iteration i. It randomly divides the set of observations into K folds of equal size. Fold i is treated as a testing set and the remaining k-1 folds are assigned as a training set. This process would be repeated K times [105]. Figure 4.7 illustrates a 5-cross validation technique.

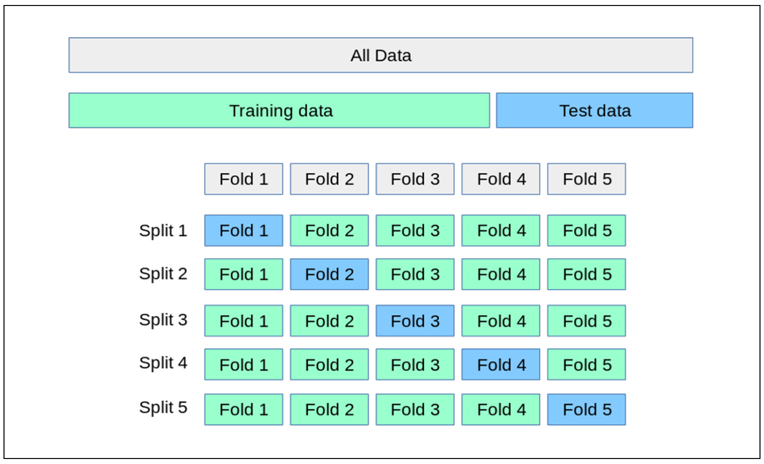


Figure 4.6: 5-Fold Cross Validation .

The performance scores from the k-fold cross validation are then averaged. Generally, 5 or

10 fold cross validation is the most widely used value, but there's no formal rule for determining it [24]. It's highly recommended that the data process a rearrangement in a way that each fold has a good representation of the whole dataset. It forces each fold to have at least m instances of each class and makes sure that one class of data is not overrepresented [105].

4.2.5.2 Evaluation Measures

Evaluation is done by comparing the system output with the reference annotations available for the test data. Metrics used in detection and classification of sound events include mainly accuracy, precision, recall, F-score, There is no consensus over which metric is universally good for measuring the performance of sound event detection, as they each reflect different perspectives on the ability of the system [24].

Metrics

- **Confusion Matrix :**A confusion matrix is a performance measurement for machine learning classification. In other words, it is a simple cross-tabulation of the actual and predicted classes for the data. It is a matrix of $n * n$ when n represents the number of classes. The row dimension contains the actual values,while the column dimension consists of the predicted label. The Figure 4.8 illustrates a representation of the confusion matrix with $n = 2$ [24, 107, 108].

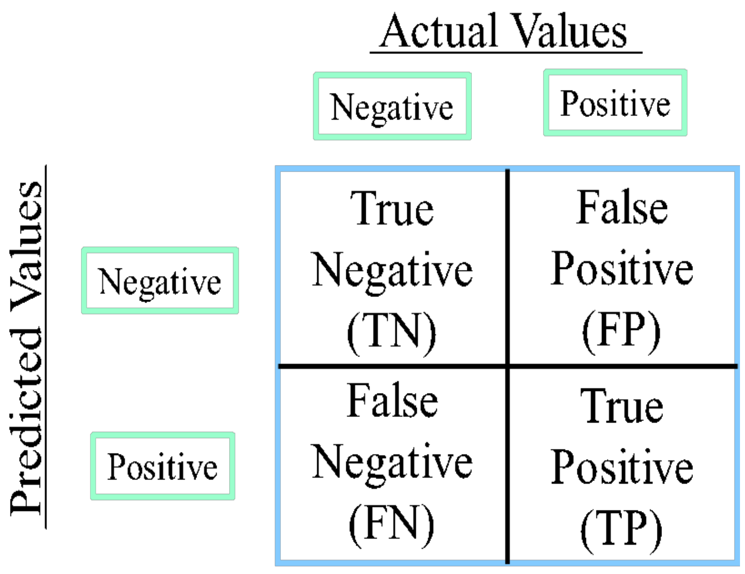


Figure 4.7: Confusion Matrix.

- True Positives (TP) are the cases where the actual class of the data point is 1 (True) and the predicted class is also 1 (True).

- True Negatives (TN) are the cases where the actual class of the data point is 0 (False), while the predicted class is 0 (False).
- False Positives (FP) are the cases where the actual class of the data point is 0 (False), while the predicted class is 1 (True).
- False Negatives (FN) are the cases where the actual class of the data point is 1 (True), while the predicted class is 0 (False).

Accuracy/Error Rate: Accuracy is the ratio between the number of correct predictions made by the model and the total number of instances. calculate the proportion of true positive and true negative in all evaluated cases. Mathematically, this can be stated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.2}$$

F1 Score: The F1 score is the average between precision and recall. It measures how precise the classifier is. The higher the F1 score, the more efficient the model becomes. Mathematically, this can be stated as:

$$F1 = 2 \times \frac{Precision + recall}{Precision \times recall} \tag{4.3}$$

4.3 Conclusion

In this chapter, we have described the overall pipeline used to conduct an Audio Tagging system. We have presented the pre-processing stage and features and have proposed two models, Resnet and Wideresnet along with dvrasarial attacks, that were used to train our system. In the following chapter, we will present the experimental setup along with the results of our experiments and analyze them in order to derive guidelines from multiple statistical comparisons.

Chapter 5

EXPERIMENTAL RESULTS AND DISCUSSION

5.1 Introduction

This chapter covers theused datasets, the development tools and environments used for conducting our work and mainly describes both the experimental design and the results that have been obtained during our research. In the training phase, we aim to compare the performances of different audio tagging systems, varying techniques, their hyperparameters, and amounts of data to observe their impact on different models. We have used accuracy, the F1-score, and the confusion matrix on the test set to evaluate our systems. Moreover, we have relied on various statistical tests to draw our conclusions. Our case study consists of four experiments:

- Experiment 1: Comparison between supervised, DCT and Mean Teacher.
- Experiment 2: Impact of Supervised Ratio.
- Experiment 3: Impact of the network architectures.
- Experiment 4: Adversarial attacks on DCT.

5.2 Dataset

To build our system or any system, the dataset is the core of the entire process since the proposed model, or in our case, the classification model, is trained and tested on it. During this thesis, we have chosen two datasets that both have an environmental background, namely: Environmental Sound Classification (ESC-10), and UrbanSound8K (UBS8K).

5.2.1 ESC-10 Dataset

The ESC-10 is a selection of 10 classes (40 clips per class) from the bigger ESC-50 dataset, representing three general groups of sounds:

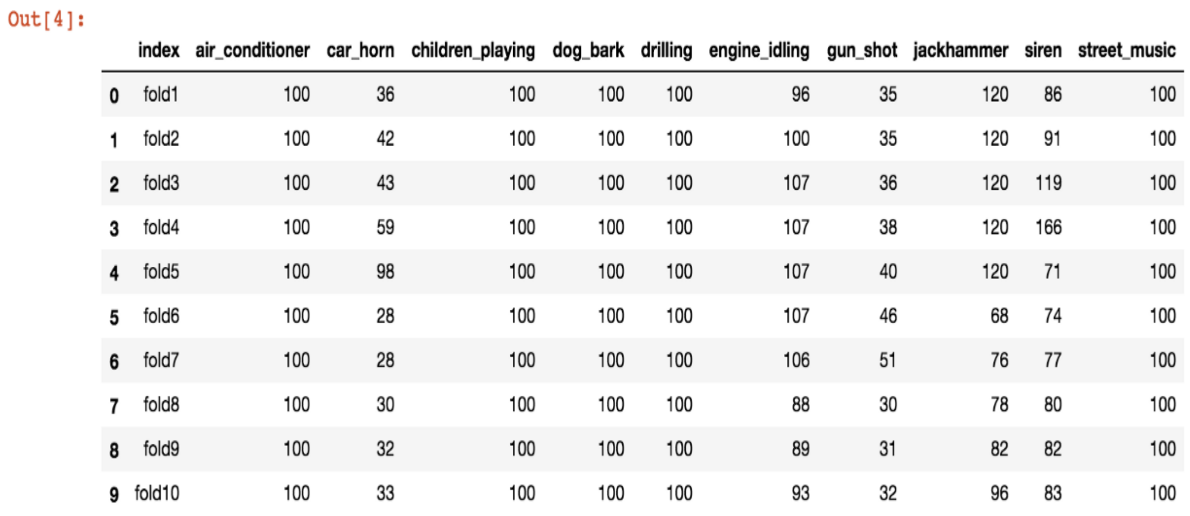
- transient/percussive sounds, with very meaningful temporal patterns (clock ticking, dog barking, sneezing).
- sound events with strong harmonic content (crowing rooster, crying baby).
- more or less structured noise/soundscapes (rain, sea waves, fire crackling, helicopter, chainsaw).

The ESC-10 dataset is a collection of 400 short environmental recordings that are 5-second-long (shorter events were padded with silence as needed). All clips have been extracted from public field recordings available through the Freesound.org project. The extracted samples were reconverted to a unified format (44.1 kHz, single channel, Ogg Vorbis compression at 192 kbit/s). The labeled datasets were consequently arranged into 5 uniformly sized cross-validation folds, ensuring that clips originating from the same initial source are always contained in a single fold [109].

5.2.2 UrbanSound8K

The Urban8k dataset contains 8732 labeled sounds excerpts that range from 0.0008s to 4s, that were extracted from urban sounds. Of these, 1798 are less than 4 s, accounting for 20.59% of the total number of samples (shorter sounds were padded with silence as needed)[110].

The classes of UBS8K are drawn from the urban sound taxonomy including air conditioner (1000), car horn (429), children playing (1000), dog bark (1000), drilling (1000), engine idling (1000), gunshot (374), jackhammer (1000), siren (929) and street music (1000). All excerpts are taken from field recordings uploaded to freesound [111]. The files are pre-sorted into ten folds, the Figure below shows the class distribution of each fold, where it is slightly imbalanced towards the car horn and shotgun classes [112].


Out[4]:

	index	air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music
0	fold1	100	36	100	100	100	96	35	120	86	100
1	fold2	100	42	100	100	100	100	35	120	91	100
2	fold3	100	43	100	100	100	107	36	120	119	100
3	fold4	100	59	100	100	100	107	38	120	166	100
4	fold5	100	98	100	100	100	107	40	120	71	100
5	fold6	100	28	100	100	100	107	46	68	74	100
6	fold7	100	28	100	100	100	106	51	76	77	100
7	fold8	100	30	100	100	100	88	30	78	80	100
8	fold9	100	32	100	100	100	89	31	82	82	100
9	fold10	100	33	100	100	100	93	32	96	83	100

Figure 5.1: The class distribution of each fold of UBS8K

5.3 Development Tools and Environments

Smarter applications are making better use of the insights extracted from data, having an impact on every industry and research discipline. At the core of this revolution lie the tools and the methods that are driving it, from processing the massive piles of data generated each day to learning from and taking useful action [113].

Deep learning systems are one of the most important consumer systems for graphics processing units (GPU) nowadays. Their training algorithm involves many large matrices of production operations. By accelerating the matrix operations via thousands of processing units in parallel, the GPU enables us to train complex deep neural networks (DNN) models efficiently, speeding up the training. Numerous studies have shown that larger and deeper DNNs can significantly increase the model accuracy for computer vision and natural language processing applications [114].

But since there is a high cost associated with these components, making them not accessible to everyone, many companies took the gesture to make them available on online platforms based on the cloud, providing a pay-by-hour manner with GPUs and a runtime fully configured for deep learning [115].

- **Kaggle**
  Kaggle is the world's largest community of data scientists and is owned by Google, Inc. It originally offered data science competitions for business problems, recruiting, and academic research purposes, but now also offers a public data platform, a cloud-based work-

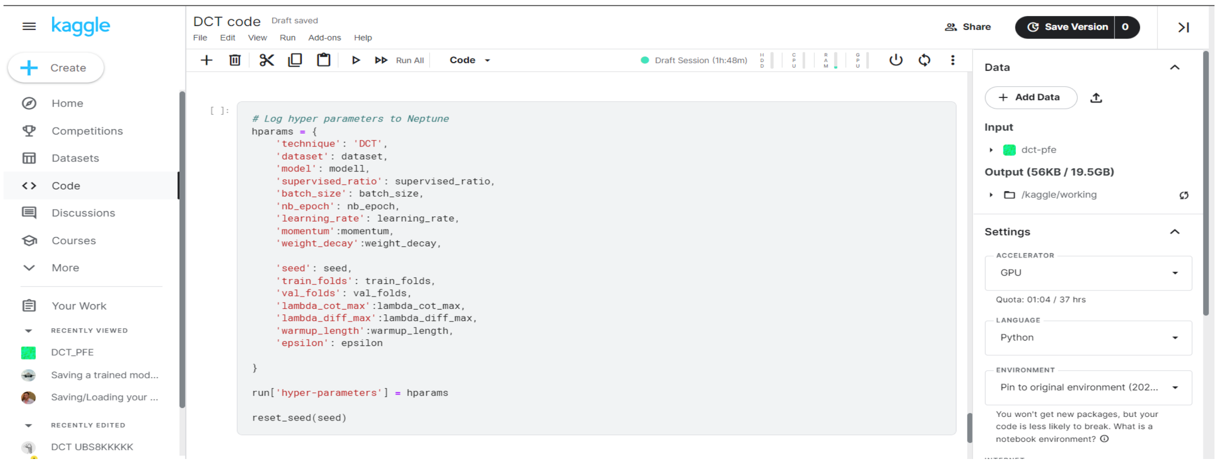
bench on Nvidia Tesla P100 for data science and AI related tasks[116]. Kaggle allows to host datasets where it supports a variety of dataset publication formats (CSVs, JSON, SQLite, HDF5. . . , ect), with a maximum storage capacity of 100GB per dataset. This platform provides its users with two different types of notebooks: scripts and notebooks with an interface based on a Jupyter notebook that allows you to write and execute directly from the browser. This environment has pre installed libraries such as Numpy, Pandas, Skit-learn, Skit-image, Tensorflow, Seaborn, etc.

One of the main advantages of Kaggle is that it provides free access to a certain amount of GPU time. The Table below illustrates the essential information about Kaggle's offers.

Table 5.1: Technical Specifications for a free kaggle notebook.

Number of Cores			**RAM(Gigabytes)**			**Time Execution (Hours)**		
CPU	GPU	TPU	CPU	GPU	TPU	CPU	GPU	TPU
4	2	4	16	13	16	12	12	9

The Figure 5.2 shows a screenshot of Kaggle notebook.



```
# Log hyper parameters to Neptune
hparams = {
    'technique': 'DCT',
    'dataset': dataset,
    'model': modell,
    'supervised_ratio': supervised_ratio,
    'batch_size': batch_size,
    'nb_epoch': nb_epoch,
    'learning_rate': learning_rate,
    'momentum':momentum,
    'weight_decay':weight_decay,

    'seed': seed,
    'train_folds': train_folds,
    'val_folds': val_folds,
    'lambda_cot_max':lambda_cot_max,
    'lambda_diff_max':lambda_diff_max,
    'warmup_length':warmup_length,
    'epsilon': epsilon

}

run['hyper-parameters'] = hparams

reset_seed(seed)
```

ACCELERATOR: GPU

Quota: 01:04 / 37 hrs

LANGUAGE: Python

ENVIRONMENT: Pin to original environment (202...

You won't get new packages, but your code is less likely to break. What is a notebook environment?

Figure 5.2: Screenshot of Kaggle notebook.

5.3.1 Libraires

The amount of data being collected and generated today is massive, and the numbers continue to grow at record rates, increasing the demand for highly efficient and intuitive tools. Python continues to be the most preferred language for scientific computing, data science, and machine learning, and the most common approach for leveraging its strengths while ensuring computational efficiency is to develop efficient python libraries. In recent years, substantial efforts are being spent on the development of such performant yet user-friendly libraries for scientific computing and machine learning [114]. Our work has been conducted using on the following libraries:

Pytorch: *PyTorch* is an open-source tensor library based on Python and Torch, mainly used for applications using GPUs and CPUs. It specializes in automatic differentiation tensor computations, and GPU acceleration, which makes it preferred over other deep learning frameworks like TensorFlow and Keras [117, 118].

Torchaudio: *Torchaudio* is a toolkit that provides building blocks for machine learning applications in the audio and speech domains within the PyTorch ecosystem. It provides important low-level functionalities like audio input/output, spectrogram computation, and a unified interface for accessing datasets[119].

Advertorch: *Advertorch* is a toolbox for adversarial robustness research. It contains many implementations for attacks, defenses and robust training methods. advertorch is built on PyTorch and leverages the advantages of the dynamic computational graph to provide concise and efficient reference implementations [120].

Torchattacks: *Torchattacks* is a PyTorch library that contains adversarial attacks to generate adversarial examples and to verify the robustness of deep learning models [121].
We have also used some other libraries such as numpy that manipulate operations using arrays and stock data as a numpy file , an easy-to-use format[122]. Pandas is mainly used for data analysis and tabular data manipulation, as in csv files [123]. The matplotlib library for data visualization and graphical plotting[124], and the scikit-image library that offers a selection of image processing algorithms [125]. Table 5.2 provides the versions of the libraries used in our research.

Table 5.2: Version of libraries used in our study.

Library	**Version**
Torch	1.8.1
Advertorch	0.2.3
Torchattacks	3.2.7
Torchaudio	0.8.1
Numpy	1.20.3
Scikit-Image	0.17.2
Pandas	1.2.5

5.4 Experimental Design

The steps to build our Audio Tagging system consist of using cross validation on both datasets and transforming them into a Log-mel spectrogram representation to feed the residual and wide residual neural networks while varying their parameters along the way. We have used the F1-score, accuracy metrics, and confusion matrix when evaluating our systems and relied on statistical tests during our analysis and discussion.

5.4.1 Cross Validation

We have performed a 5-cross validation on the Environmental Sound Classification dataset (ESC-10), and a 10-cross validation on the UrbanSound8K dataset (UBS8K). Cross-validation consists of shuffling the dataset randomly in order to generate different combinations to ensure that the events are present in the testing set. Then split the dataset into 5 and 10 folds, and take one fold as a testing set and the remaining as a training set. Repeat this process until we cover all the folds, and finally calculate the overall performance score by taking the average of the results over 5 and 10 folds. We have performed 5-fold and 10-fold cross validation using the Scikit-Learn library.

5.4.2 Feature Extraction

For Audio Tagging, a significant amount of information is contained in the relative distribution of energy in the frequency of an audio signal. The information stored in the frequencies allows for making comparisons between audio files while paying attention to the most relevant characteristics of the audio. The provided audio samples in our UBS8K dataset are resampled at 22kHz since the audio files have different sample rates (varies from 8kHz to 192kHz) while adapting to a mono format. As for the ESC-10 dataset, the files are sampled at 44 kHz in mono format. We handled the problem of different audio time durations by padding silence into each audio file for equal time lengths.

The first step of feature extraction consists of converting an audio recording from an analog to a digital time signal. Afterward, the digital signal is converted from the time domain to the frequency domain by the use of the short time Fourier transform (STFT) with 2048 FFT points to each of the 4-sec and 5-sec UBS8K and ESC-10 datasets, respectively. It has two techniques: framing and applying the hamming window function with a 75% overlap (since the hop size is 512) to specify the number of sample rates a frame should slide across. Next, a 64-mel filter bank is applied to the calculated power spectrum to finally calculate the logarithmic energy of each filter bank output to obtain a Log-mel spectrogram. The Table 5.3 illustrates the Log-Mel Spectrogram setup.

Table 5.3: Log-Mel Spectrogram setup.

Parameters	**Value**	
Sampling Rate	ESC-10 = 44kHz	UBS8K = 22kHz
Frame Length	ESC-10 = 46 ms	UBS8K = 93 ms
Window Size	2048	
Hop Size	512	
Mel Bands	64	

5.4.3 Training Stage

In order to carry out the semi-supervised approach, we have conducted a baseline to compare our SSL techniques with , which does not require much expertise and time to build. Our baseline was trained on 100% of labeled data, using 5 folds of cross validation for the ESC-10 dataset (10 folds of cross validation for UBS8K). Then we applied feature extraction to transform our audio samples into a Log-mel spectrogram (feature vector) and fed it to our WRN28_2 model that would create a probability distribution over 10 classes and produce an output vector. Each element of the vector is the probability that the input belongs to the corresponding class. The most likely class is chosen by selecting the index of that vector having the highest probability. The supervised baseline was trained on a 64-batch size and a 0.003 learning rate on 100 epochs. The Figure below illustrates the pipeline of our baseline for an audio tagging system.

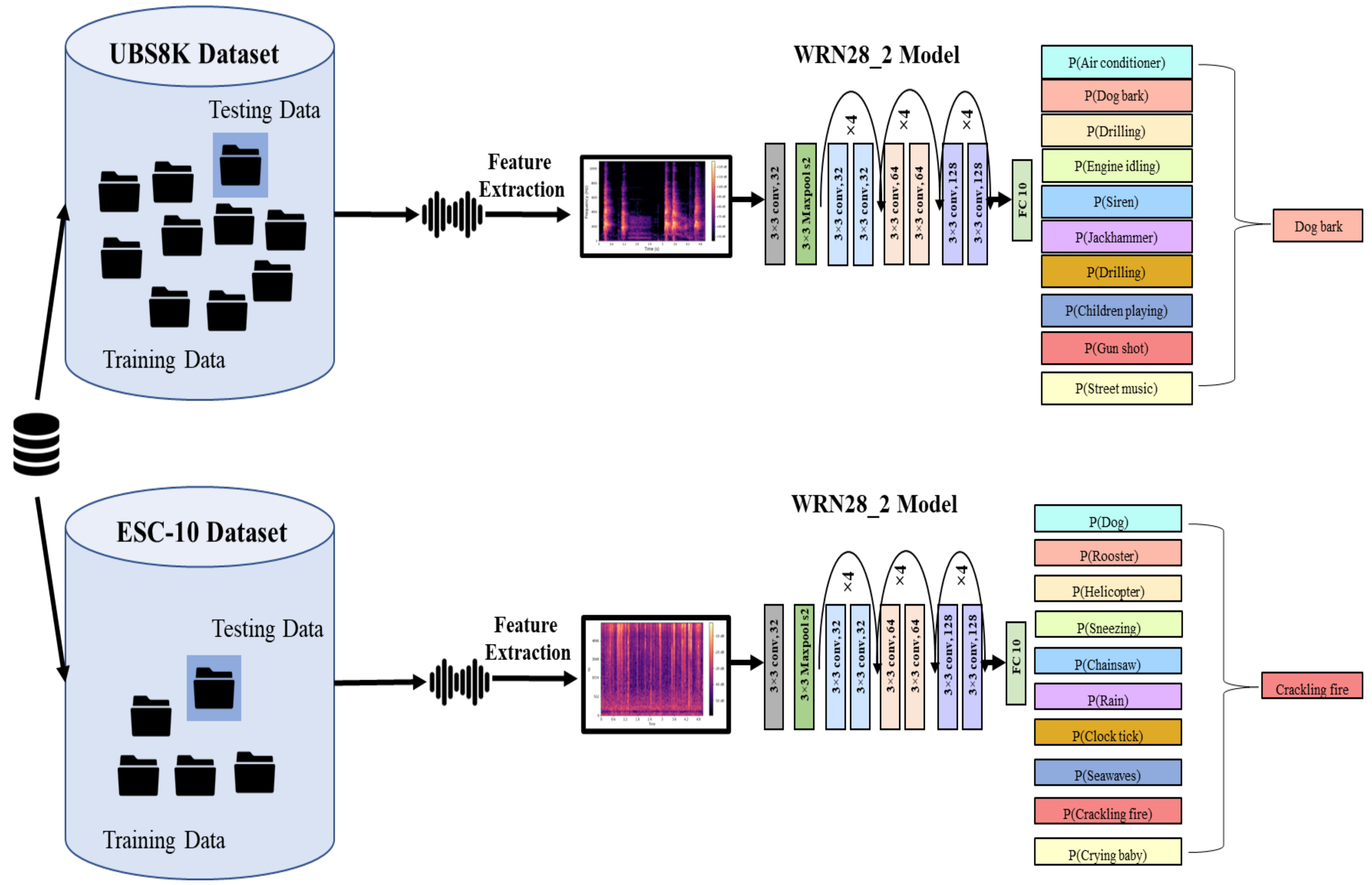


Figure 5.3: The pipeline of our baseline for an Audio Tagging system.

We used two semi-supervised approaches, namely Deep Co-training to train our main Audio Tagging system, and the Mean Teacher Approach to compare it with our main system. Where the whole process can be broken down as follows: To simplify the Deep Co-training method, we have illustrated Figure 5.4 to show a straightforward example. The process starts with taking two audio samples from a labeled data batch and one audio sample from an unlabeled data batch, where we apply feature extraction to get the feature vectors (L_1 , L_2 present the labeled samples and L_U presents the unlabeled sample) to feed to both WRN28_2 models M1 (view1) and M2 (view2). Then we would compute the cross entropy loss (Loss sup) to ensure that network predictions for labeled data are consistent with the ground truth for both models M1 and M2. On the other hand, the Jensen-Shannon divergence loss function (Loss cot) is applied to the unlabeled examples in order to force the two models to agree with each other. For the essential idea of the DCT, we have used the fast gradient sign method (FGSM) to generate adversarial examples with a magnitude of perturbation (epsilon) is equal to 0.02 for most experiments, where it can fool the model but doesn't display any perceivable changes on the sample. The cross entropy loss function (Loss div) then is used to force a network to agree with the predictions of the other network's adversarial examples (ex: P_{L1} and $AdvL_1$ (M2)) [126].

In the mean teacher approach, we make use of labeled (L) and unlabeled data (LU) and feed them to both the student and the teacher model (WRN28_2). In which only the student model is trained. And, a very minimal number of weights from the student model are assigned to the teacher model at every step, called exponential moving average weights (EMA). As shown in Figure 5.4, two cost functions were used, namely : classification cost and consistency cost. Here, classification cost is calculated as a binary cross-entropy between the label predicted by the student model (P_L) and the ground truth. Consistency cost is the mean squared difference between the predicted outputs of the student (P_{L_S}) and the teacher model (P_{L_T}). Consistency cost is actually the distribution difference between two predictions, and the original label is not required. So, instead of labeled data, we may utilize unlabeled data, but no classification cost would be applied. One of the important factors that plays a crucial role in adding robustness to the model is the introduction of noise during training [80].

The Table below shows the different parameters that our SSL approaches used during the experiments.

Table 5.4: MT and DCT parameters setup.

Technique	**Alpha**	**Batch size**	**Lr**	**Epoch**	**Supervised ratio**	**warmup Length**	**Epsilon**	λ_{cot}	λ_{diff}
MT	0.999	64	0.001	100	0.8	50	–	1	–
DCT	–	100	0.0005	100	0.8	100	0.02	1	0.5

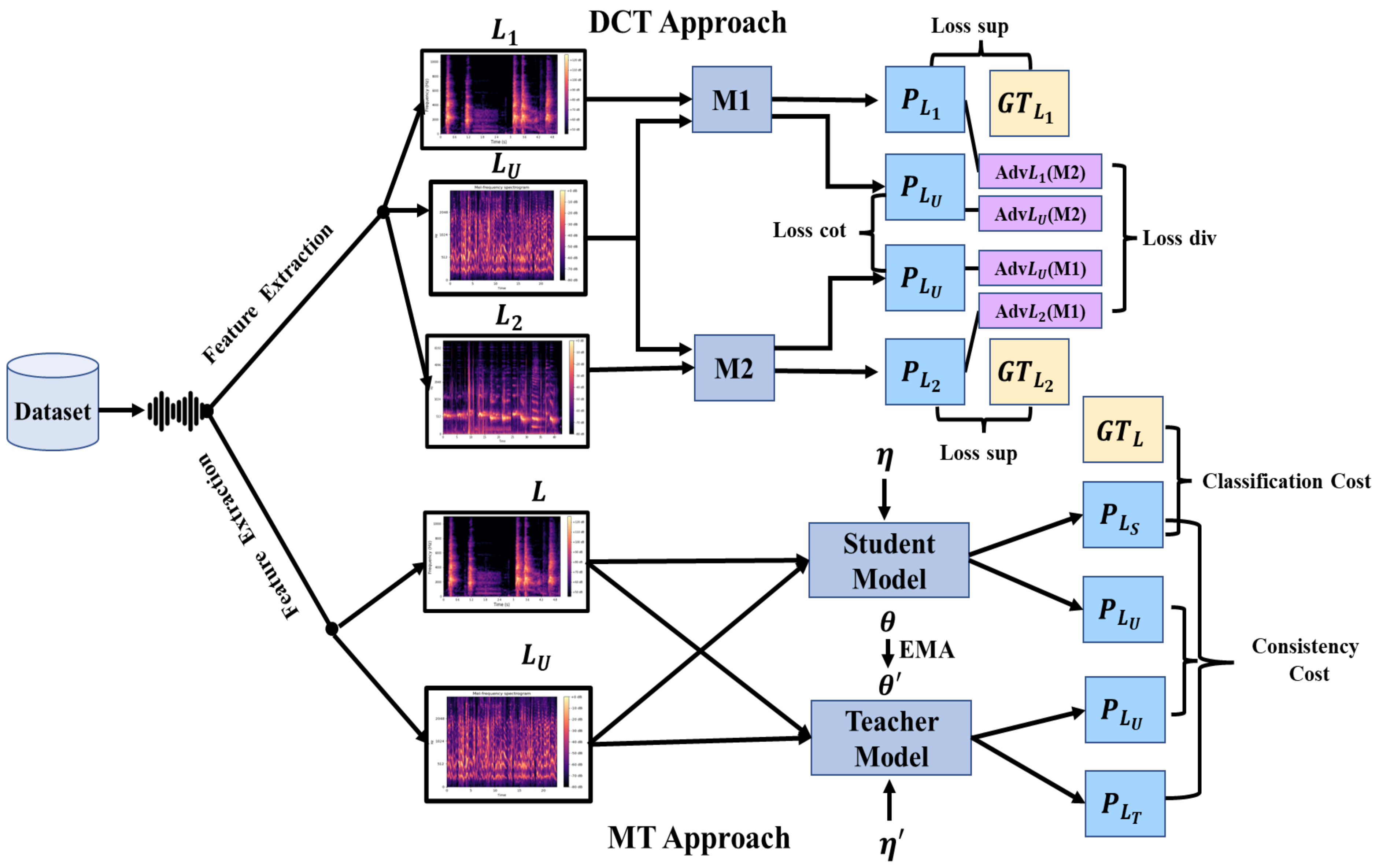


Figure 5.4: The pipeline of DCT and MT approaches for an Audio Tagging system.

5.4.4 Evaluation Procedure

This study aims to compare the performances of different Audio Tagging systems while varying the techniques and their parameters. We have evaluated the overall performance of each system using the accuracy metric and an F1-score when performing class-wise evaluation. Also, we made use of a confusion matrix to define the performance of the classification algorithm and were able to make observations on the model's behavior over all the classes. For some experiments, we have chosen some graphical representations such as a bar plot and a line plot to observe and analyze the evolution of the systems. For other experiments, we have relied on strong statistical tests, namely the Wilcoxon signed-ranks test, which compares the classifiers in a pairwise manner over multiple datasets. The Kruskal-Wallis Test assesses the differences among three or more groups, along with a post hoc test that is the Bonferroni-Dunn Test, to compare all classifiers with a control system.

5.5 EXPERIMENTAL RESULTS

5.5.1 FIRST EXPERIMENT : Comparison between supervised, DCT and Mean Teacher

In this experiment, we used two semi-supervised approaches for Audio Tagging, namely: Mean Teacher (MT) and Deep Co-Training (DCT). We have conducted our work using two residual neural networks, Wideresnet28_2 (28 layers in depth and a width of 2) and a resnet34 that consists of 34 layers deep. Our models have been evaluated against supervised baseline systems built on the same network architecture. We have trained these two models on two well-known datasets: the Urban8k dataset (UBS8K) and the Environmental Sound Classification dataset (ESC-10). For evaluating our models, we have resampled each dataset following K[1]-fold cross validation, resulting in 2 sets of train(i) and test(i), $i \in \{1, ..., k\}$. We have trained our two SS1 approaches on two subsets of data: the supervised subset S and the unsupervised set U. The supervised ratio was fixed at 0.8, which means that the supervised subset consists of 80% of train_i, while the remaining 20% is from the unsupervised set U. whereas, the supervised baseline consists of 100% of train(i) . We have evaluated our systems on test(i) for each fold $i \in \{1, ..., k\}$, and we have reported the mean of these K measurements in Table 5.5.

[1]K=5 is the number of folds in the ESC-10 dataset, K=10 is the number of folds in the UBS8K dataset.

Table 5.5: Average F1-score results of DCT, MT and baseline system on UBS8K and USC-10.

UBS8K							**ESC-10**						
Sound Events	**DCT**		**MEAN TEACHER**		**FULL SUPERVISED**		**Sound Events**	**DCT**		**MEAN TEACHER**		**FULL SUPERVISED**	
	WRN28_2	**resnet 34**	**WRN28_2**	**resnet 34**	**WRN28_2**	**resnet 34**		**WRN28_2**	**resnet 34**	**WRN28_2**	**resnet 34**	**WRN28_2**	**resnet 34**
Air conditioner	0,57	0,52	0,52	0,53	0,57	0,49	**Dog**	0,87	0,91	0,91	0,90	0,85	0,93
Car horn	0,88	0,83	0,90	0,88	0,92	0,86	**Rooster**	0,96	0,93	0,95	0,95	0,94	0,95
Children playing	0,75	0,75	0,82	0,72	0,83	0,75	**Clock Tick**	0,96	0,86	0,86	0,83	0,95	0,77
Dogbark	0,83	0,85	0,88	0,82	0,87	0,82	**Helicopter**	0,84	0,78	0,70	0,70	0,83	0,76
Drilling	0,71	0,66	0,72	0,65	0,75	0,64	**Chainsaw**	0,86	0,88	0,84	0,79	0,87	0,83
Engine idling	0,66	0,60	0,61	0,61	0,62	0,63	**Rain**	0,76	0,78	0,78	0,77	0,78	0,75
Gunshot	0,95	0,95	0,94	0,94	0,98	0,94	**Sea waves**	0,87	0,85	0,87	0,85	0,85	0,87
Jackhammer	0,65	0,60	0,58	0,58	0,55	0,57	**Crackling fire**	0,91	0,88	0,82	0,82	0,90	0,79
Siren	0,85	0,81	0,85	0,80	0,87	0,80	**Crying baby**	0,99	0,95	0,96	0,90	0,96	0,99
Street music	0,75	0,74	0,80	0,74	0,79	0,72	**Sneezing**	0,91	0,89	0,90	0,87	0,90	0,96
Average F1-score	**0,76**	**0,73**	**0,76**	**0,73**	**0,77**	**0,72**	**Average F1-score**	**0,89**	**0,87**	**0,86**	**0,84**	**0,88**	**0,86**

The results in Table 5.5 show that the semi-supervised systems and the baseline yield an average F1-score landing in the range of 72% to 89%. Aside from that, all the compared systems outperform each other across all audio tags. However, our initial analysis does not reveal any 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 [127]. To cope with this shortcoming, appropriate statistical tests should be conducted thoroughly [128]. To this end, we have statistically compared the performances of these techniques using the Wilcoxon signed-ranks test. For further analysis of these results, we have compared the performances of these systems in a pairwise manner based on the Wilcoxon test in Table 5.6 The first row of each cell specifies the number of Win/Tie/Loss of the system in the column over the system 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 10 is statistically significant using the Wilcoxon test.

Table 5.6: Wilcoxon signed-rank test results.

Dataset	Baseline		DCT WRN28_2	DCT Resnet34	MT WRN28_2	MT Resnet34
UBS8K	Sup WRN28_2	W/T/L	3/0/7	**1/0/9**	3/0/7	**1/0/9**
		p-value	0.38	**0.02**	0.16	**0.009**
	Sup Resnet34	W/T/L	**10/0/0**	7/0/3	**9/0/1**	7/0/3
		p-value	**0.005**	0.38	**0.01**	0.38
ESC-10	Sup WRN28_2	W/T/L	**8/0/2**	3/0/7	6/0/4	**2/0/8**
		p-value	**0.047**	0.13	0.57	**0.03**
	Sup Resnet34	W/T/L	6/1/3	5/0/5	5/0/5	3/1/6
		p-value	0.22	0.50	0.95	0.22

According to the above statistical analysis, we can derive the following inferences:

- Taking a closer look at the UBS8K dataset, we can observe that DCT+WRN28_2 and MT+WRN28_2 are significantly better than the supervised WRN28_2 baseline with a confidence level of 0.005 and 0.013, respectively. On the other hand, the ResNet34-based systems exhibit the opposite behavior, as we can see that both DCT+Resnet34 and MT+Resnet34 perform significantly worse than the supervised WRN28_2 baseline with a p-value of 0.02 and 0.009, respectively.
- A close examination of the ESC-10 Table reveals that the DCT+WRN28_2 yields significantly better scores compared to the supervised WRN28_2 baseline, with a p-value of 0.047. But it is worth noting that MT+resnet34 also exhibits very low performance

compared to the supervised WRN28_2 baseline, with a p-value of 0.037.

Based on the above discussion, we can conclude that the semi-supervised methods successfully boost the performance of WRN28_2 -based systems. As compared to ResNet34-based systems, they show totally different behavior.

In order to get better insight on how DCT and MT improve the performance of the WRN28_2 based model, we display in Figure 5.5 (a), (b), and (c) the confusion matrices of and the supervised WRN28_2 baseline ,DCT+WRN28_2 and MT+WRN28_2 , respectively, as computed on the test set.

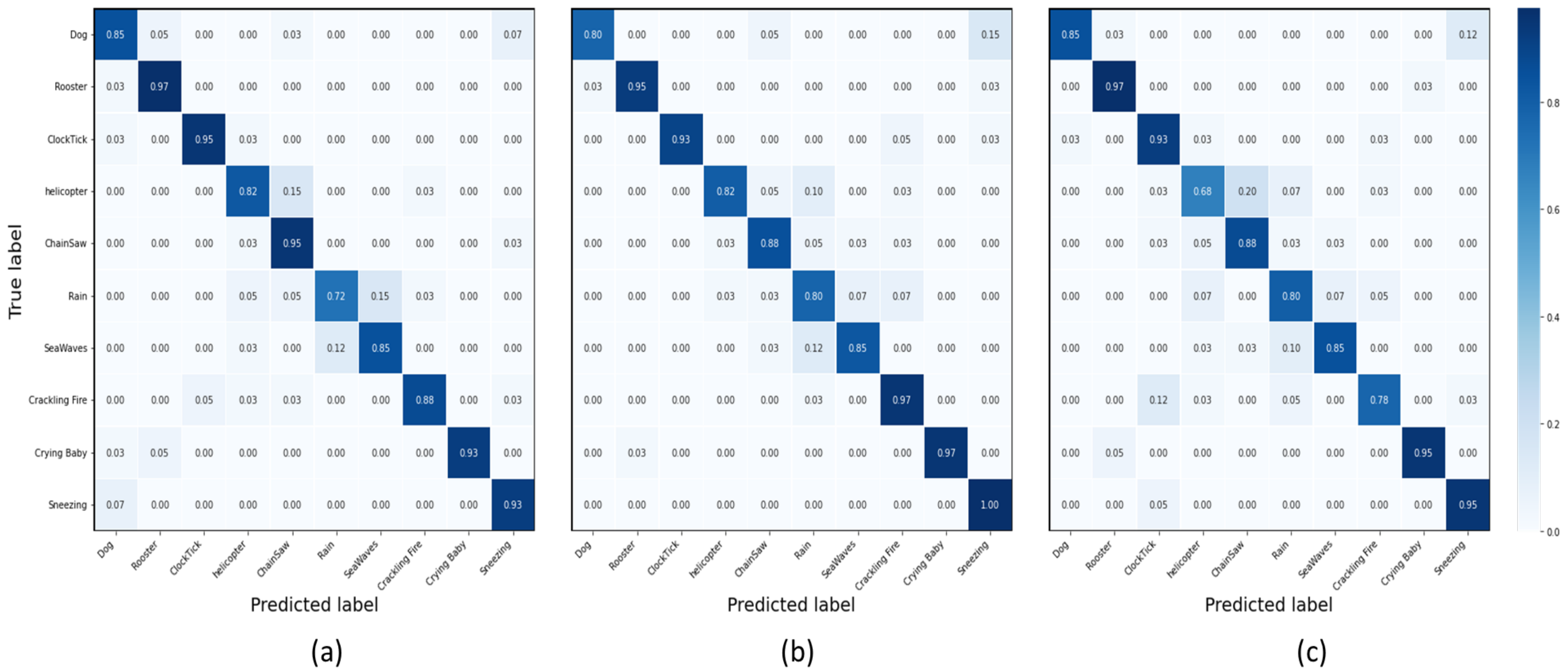


Figure 5.5: Confusion Matrices of the supervised baseline (a) ,DCT (b) and MT (c) on WRN28_2 model.

The matrices indicate appreciable diagonals, meaning that many classes are correctly classified. However, we observe that some classes are easier to classify while others are not; for instance, the matrix of WRN28_2 shows an appreciable diagonal, meaning that several classes are correctly classified. However, some classes are misrecognized as others; for instance, "sea waves" are classified as "rain" and "chansaw" are classified as "helicopter" and vice versa. which indicates that most misclassified classes are quite similar, making it difficult even for humans to distinguish between them. To address this issue, Deep Co-training makes use of adversarial examples to encourage the models to generate complementary representations. Each is resistant to the adversarial examples of the other. It is worth noting that the adversarial examples of some events, such as "cracking fire", "sneezing" and "crying baby", significantly improve the performance of WRN28_2 as illustrated in Figure 5.5. Mean Teacher, however, was unable to produce comparable outcomes.

5.6 SECOND EXPERIMENT : Impact of Supervised Ratio on Audio Tagging Systems

In this experiment, we have built our audio tagging systems while varying the ratios of labeled data. The aim is to explore the impact of the amount of labeled data when training our WRN28_2 models on both the DCT and MT approaches. To be precise, we have changed the supervised ratio each time from a range of 0.1 to 0.9 and trained the models on the first K-1 folds and tested them on the remaining fold (fold K) for both ESC-10 and UBS8K. Note that we have only conducted one iteration of K-fold CV due to a lack of dedicated training resources. The results are depicted in Figure 5.6.

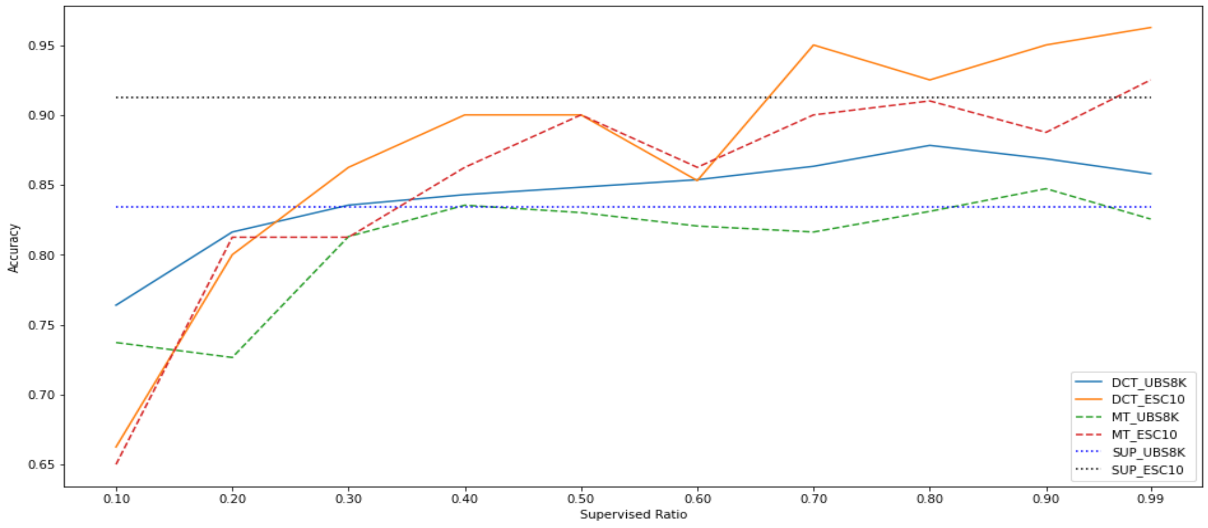


Figure 5.6: Effect of the supervised ratio.

The curves shown in Figure 5.6 indicate that the performance of the semi-supervised approaches progressively increases as the supervised ratio grows. Also, DCT shows superior performance over MT in most cases. It yields better scores than the supervised model while using only a small fraction (30%) of the labeled data on the UBS8K dataset while using about (65%) of the labeled data on the ESC-10K dataset. On the other hand, the MT requires a slightly larger fraction of labeled data from UBS8K to reach a higher performance than the baseline in the range of (38% to 46%) and (80% to 95%), while for the ESC-10 it necessitates a large fraction (95%) of the labeled data.

5.7 THIRD EXPERIMENT : Impact of the network architecture

According to the first experiment, the WRN28_2 model provides better performance scores when compared to Resnet34. In order to further investigate the impact of the network architecture, we have carried out the following experiment. We have trained 6 different network architectures with DCT on both the UBS8K and ESC-10 datasets for the purpose of designating the architecture that yields the highest results. In which the light blue indicates the resnet18 model, the dark blue indicates the resnet34 model, the light green indicates the resnet50 model, the dark green indicates the WRN28_2 model, the pink indicates the WRN28_4 model, and the red indicates the WRN28_8 model. Figure 5.7 gives the obtained accuracy results on the test fold.

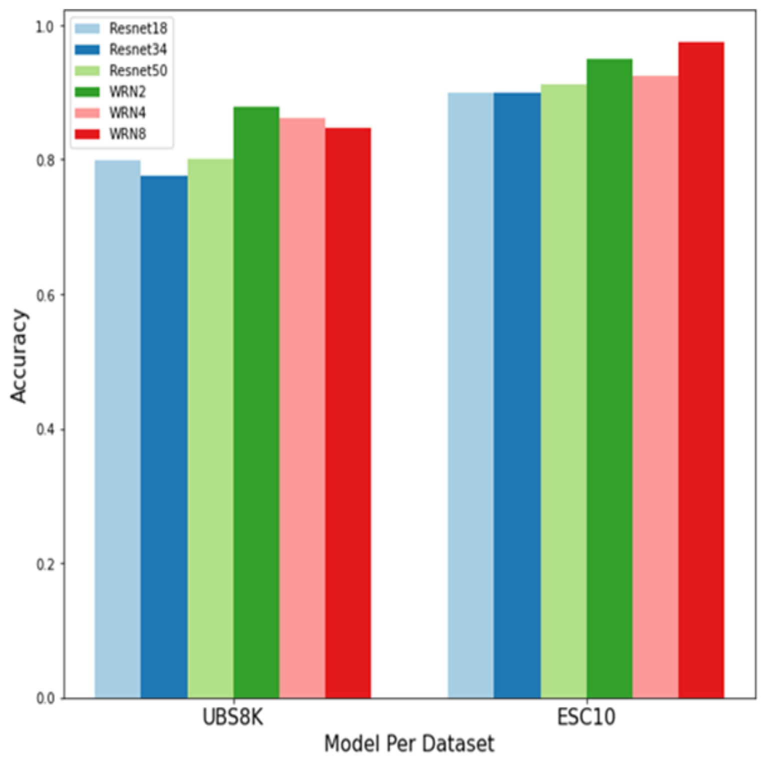


Figure 5.7: Impact of different network architecture.

The analysis of the above bar plot can be summarized by two main observations:

- We observe that the models perform better when trained on the ESC-10 dataset than on the UBS8K dataset, and that's due to the fact that the UBS8K dataset is considered slightly imbalanced, where different classes have an uneven distribution of observations compared to ESC-10, which has the same amount of data in each class. For example, in the UBS8K dataset, the air conditioner class has about 1000 instances (audio files), while the gunshot class hardly reaches 374 instances.

- We note that WRN models yield the best performance as the accuracy scores vary between 0.93 and 0.98 for ESC-10 and 0.84 and 0.87 for UBS8K. Whereas the performance of the Resnet models varies between 0.775 and 0.80 for UBS8K and 0.9 and 0.91 for ESC-10 Therefore, WRNs are far superior to their commonly used thin and very deep counterparts, Resnets. A possible cause of this behavior might be related to the training process of Resnets. As suggested by many authors [103], training very deep residual networks has a problem of diminishing feature reuse, which makes these networks very slow to train.

5.8 FOURTH EXPERIMENT : Analysis of adversarial attacks on DCT

The purpose of this experiment is to explore the impact of adversarial attacks on DCT. This experiment is divided into two parts. The first experiment is devoted to varying the adversarial example magnitude of perturbation (epsilon) using the FGSM attack. The second experiment revolves on comparing different adversarial attacks and to analyze their impact on the behavior of DCT-based systems.

5.8.1 EXPERIMENT 4A : Varying the magnitude of perturbation

We have implemented this experiment with various different values of the perturbation magnitude epsilon $\in [0, 50]$. Figure 5.8 illustrates the change in accuracy scores for each dataset.

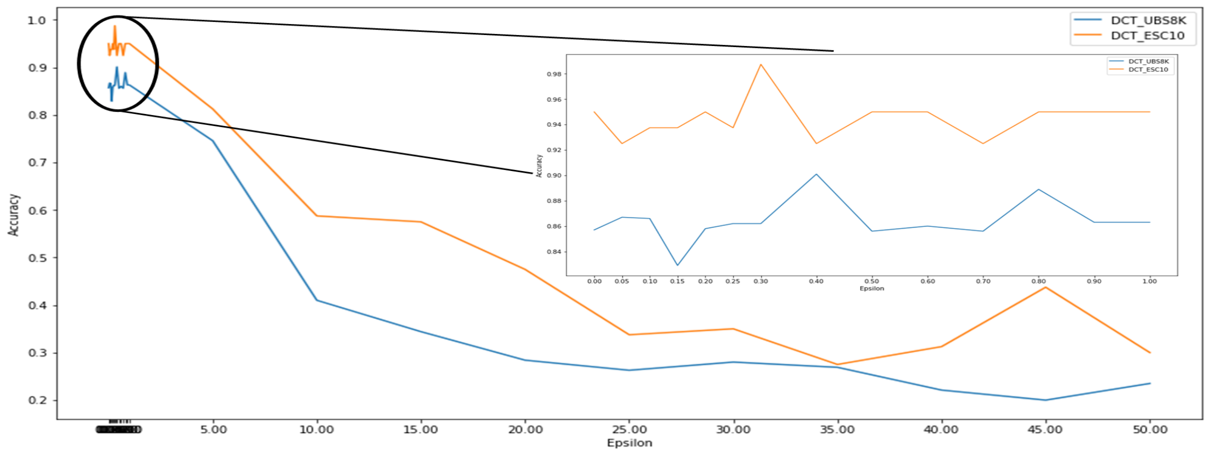


Figure 5.8: Impact of different network architecture.

According to Figure 5.8, both curves display similar observable behaviors. When epsilon < 1, the performance of DCT on both datasets is stable having a small variation here and then.On the other hand, as epsilon gets larger, performance stability is compromised, as we notice a significant decrease in accuracy. which implies that greater values of the magnitude of perturbation badly influence the performance of DCT, whereas setting the magnitude perturbation in range [0,1] generates adversarial examples that fools the models to provide different and complementary information about the data, thus, the models would be resistant to adversarial examples and as a result it would enhance the models performance.

5.8.2 EXPERIMENT 4B : Varying the adversarial attack strategy

We have performed this experiment using four attacks, namely: L-BFGS, FGSM, DEEP FOOL and CW. The hyperparameter settings of each attack are provided in Table 5.7.

Table 5.7: MT and DCT parameters setup.

parameters	L-BFGS	FGSM	DEEP FOOL	CW
Clip min	0	0	–	–
Clip max	1	1	–	–
initial const	0.01	–	–	0.001
Epsilon/Overshoot	–	0.02	0.02	–
kappa	–	–	–	0

According to the previous experiments, FGSM attack has shown remarkable performance along different models. In order to further explore this matter, we have conducted this experiment where we have built numerous audio tagging systems based on DCT approach using different adversarial attacks, where the most common magnitude of perturbation was assigned to each attack. But in fact we are only interested in comparing an unperturbed system with the other alternatives. In other words the unperturbed system would be treated as a baseline against the other perturbed systems to see the impact and the performance of applying an adversarial attack on an audio distribution.

The results of this experiment on ESC-10 and UBS8k are shown in Table 5.8. The last row specifies the average ranks of each system tested on the attack computed using the Kruskal Wallis test.

Table 5.8: Average Ranks and F1-scores of DCT variants.

UBS8K Dataset	**No perturbation**	**FGSM**	**L-BFGS**	**CW**	**DEEP FOOL**	**ESC-10 Dataset**	**No perturbation**	**FGSM**	**L-BFGS**	**CW**	**DEEP FOOL**
Air conditioner	0.56	0.87	0.78	0.00	0.59	**Dog**	0,88	0,94	0,93	0,8	0,94
Car horn	0.92	0.95	0.87	0.56	0.79	**Rooster**	1	0,93	0,89	0,7	0,77
Children playing	0.75	0.87	0.81	0.46	0.81	**Clock Tick**	0,94	1	1	0,71	0,82
Dog bark	0.77	0.87	0.82	0.70	0.75	**Helicopter**	0,8	0,93	0,93	0,77	0,93
Drilling	0.70	0.75	0.74	0.21	0.69	**Chainsaw**	0,8	0,93	1	0,71	0,77
Engine idling	0.75	0.80	0.73	0.64	0.63	**Rain**	0,86	0,89	0,8	0,77	0,8
Gunshot	0.97	0.96	0.96	0.72	0.95	**Seawaves**	0,94	1	0,82	0,82	0,67
jackhammer	0.92	0.90	0.93	0.42	0.84	**Crackling fire**	0,94	0,88	1	0,93	0,93
Siren	0.85	0.93	0.91	0.70	0.90	**Crying baby**	1	1	0,93	0,82	0,7
Street music	0.84	0.91	0.81	0.38	0.70	**Sneezing**	0,82	1	0,94	0,38	0,8
Average F1-score	**0,8**	**0,88**	**0,84**	**0,48**	**0,77**	**Average F1-score**	**0,9**	**0,95**	**0,92**	**0,74**	**0,81**
Average Rank	**28.35**	**37.15**	**31.4**	**7.4**	**23.15**	**Average Rank**	**30**	**36.45**	**32.7**	**11.5**	**16.9**

Starting with the ESC-10 , the Kruskal Wallis test rejects the null hypothesis where H statistic $= 22.1 \succ$ critical chi-square $= 7.78$ for $k - 1 = 5 - 1$ degrees of freedom and $\alpha = 0.1$ and since the p-value $= 1.9.10 - 4 \prec \alpha = 0.1$ where there exists at least one pair of systems with significantly different performances. In other words, the difference between the mean ranks of some groups is big enough to be statistically significant. Because we are only interested in testing whether DCT systems based on different attacks has a significant performance we have conducted a Bonferroni-Dunn test at a 10% significance level with the critical value $q0.10 = 16.65$ and the critical difference CD $= 11.77$. The results of this test are shown by Figure 5.9. On the horizontal axis, we represent the average ranks of each DCT variant and mark using a thick line an interval of 2×CD one on the right and the other to the left of no perturbation system mean rank.

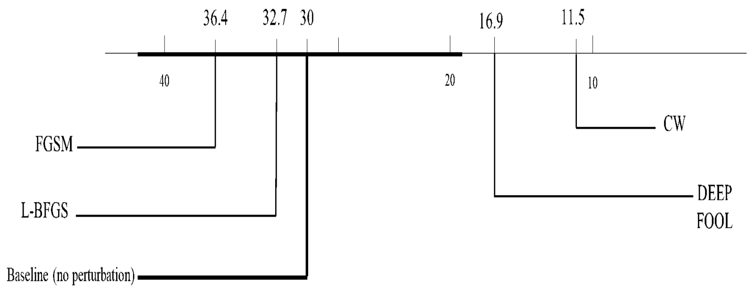


Figure 5.9: Comparison of the baseline system against the 4 other systems with the Bonferroni-Dunn test on the ESC-10 dataset.

Based on the Bonferroni analysis, the following conclusions can be drawn:
DEEPP FFOL and CW fall outside the marked interval and while CW has the lowest rank, these two have significant difference mean ranks compared to the baseline. On the other hand FGSM and LBFGS have a Mean Rank difference of 6.55 and 2.85 outperforming the baseline scores.

For the UBS8K dataset , the Kruskal Wallis test also rejects the null hypothesis where H statistic $= 24.05 \succ$ critical chi-square $= 7.78$ for $k - 1 = 5 - 1$ degrees of freedom and $\alpha = 0.1$ and since the p-value $= 7.10 - 5 \prec \alpha = 0.1$ where it exist at least one pair of systems with significantly different performances. We have conducted a Bonferroni-Dunn test at a 10% significance level with the critical value q0.10 =16.87 and the critical difference CD = 11.86. The results of this test are shown by Figure 5.10 based on the same creatia of Figure 5.9.

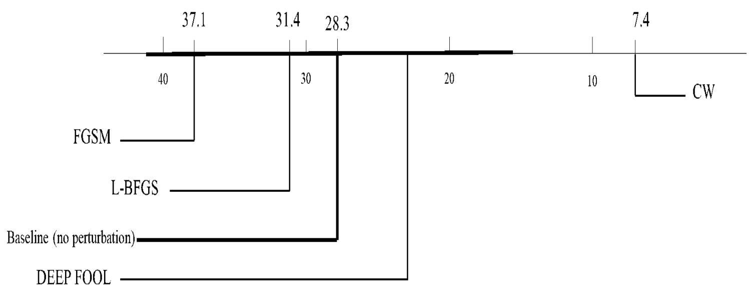


Figure 5.10: Comparison of the baseline system against the 4 other systems with the Bonferroni-Dunn test on the ESC-10 dataset.

Based on the Bonferroni analysis, the significance difference between the baseline and CW is marked outside of the interval and has the lowest rank. whereas DEEP FOOL has increased its performance compared to the ESC-10 dataset since it has a mean rank difference of 5.2. While FGSM and LBFGS have again exceeded the baseline.

In conclusion L-BFGS and FGSM show very promising results on audio based systems. Where for DEEP FOOL we can't make any conclusion since the attack shows different behavior. On the other hand, CW provides the worst performance of them all.
Table 5.9 illustrates the time execution during the training phase for different adversarial attacks for both ESC-10 and UBS8K datasets.

Table 5.9: Time execution for different adversarial attacks.

Dataset/Attack	**FGSM**	**L-BFGS**	**CW**	**DEEP FOOL**
ESC-10	**15m**	**1h 11m**	**17m**	**25m**
UBS8K	**1h 21m**	**11h 24m**	**3h 18m**	**9h 8m**

5.9 Conclusion and summary of experimental findings

In this chapter, we have presented the results of our experimental enquiries. Several lessons can be derived from our analysis:

- Training models based on a semi-supervised learning approach with both labeled and unlabeled data is highly efficient in terms of performance and time.
- Some adversarial attacks improve the prediction performance of an Audio Tagging system, although they are not always beneficial and can negatively affect the performance of such systems.
- The hyperparameter epsilon is a crucial factor that has an extreme influence on the adversarial example and the model's performance. For that reason, it should be defined carefully and values should not surpass 1.
- Wide residual-based models yield better scores than the standard residual networks since they have fewer layers and faster timing.

Conclusion

Contributions and summary of experimental findings

The primary goal of this thesis was to conduct an empirical analysis and make comparisons among Audio Tagging systems. To this end, we built numerous audio tagging systems, and conducted several experiments based on UBS8K and ESC-10 datasets. We made use of two deep learning-based models: Residual Neural Network (ResNet) and Wide Residual Neural Network (WideResNet) to train our audio data. Most of our experiments focused on the WideResnt model. We used two semi-supervised approaches :Deep Co-Training and Mean Teacher, in which they held to account both labeled and unlabeled data. To avoid deriving conclusions affected by chance, we have used three statistical tests, namely: the Wilcoxon signed rank test, the Kruskal-Wallis test, and the Bonferroni-Dunn test.
From this experimental study, we can derive the following conclusions:

- Using both labeled and unlabeled audio data in a semi-supervised approach augments the performance of an Audio Tagging system.
- Wide Residual models outperform the old-fashioned residual models since they are better in terms of complexity and time.
- Applying semi-supervised learning approaches to train an Audio Tagging system demonstrates a noticeable superiority over systems trained in a supervised fashion (100% labeled data).
- Deep Co-training based systems (DCT) outperform Mean Teacher based systems (MT) since DCT works with adversarial examples in which they enhance the models' performance.
- In DCT, the magnitude of perturbation epsilon should be chosen carefully in such a way that it fools the models but isn't perceivable by humans. We highly suggest choosing an $\epsilon < 1$.

- Not all adversarial attacks have a positive impact on the model's performance. For instance, on one hand, FGSM and L-BFGS attacks showed great performance when applied to generate adversarial examples on the model. But Deep fool made a conflicted behavior and where CW had the worst results.
- Analyzing the obtained scores statically is an excellent way of comparing and identifying differences between Audio Tagging systems.

Limits and Future work

This thesis has revealed several interesting areas for improvements, and can be further explored in the near future. We have achieved noticeable improvements in the performance of our audio tagging systems. It is possible, however, to further improve these performances by:

- Applying an augmentation technique to the UBS8K dataset is recommended since it has slightly imbalanced data. The most common data augmentation techniques vary depending on whether we are handling a raw audio augmentation, in which we can simply Time Shift, Pitch Shift, Time Stretch , or apply Noise Injection, but when handling a Spectrogram Augmentation, a method known as SpecAugment is used.
- Another promising solution would be to test the hyperparameters of the adversarial attacks that performed badly and analyze the results to come up with a convincing conclusion.
- Train models on the Universal Adversarial Perturbations and compare the results with an old-fashioned adversarial attack.
- We can use other feature extraction techniques such as Mel-frequency cepstral coefficients (MFCC), In addition, future work should consider also using raw audio signals as inputs to train the audio tagging systems.
- Since WideResnet has shown great results, it would be interesting to compare it with other deep residual networks such as EfficientNet or Alexnet.

During this project, we have encountered a few struggles because of the long training time since we lack robust hardware to perform deep learning tasks. which prevented us from conducting more experiments . Nonetheless, this experience has taught us many useful skills, such as how a machine perceives audio, thesis writing, the main steps of implementing machine learning experiments using Pytorch, and developing systems on a cloud-based platform such as Kaggle.

Bibliography

[1] Eduardo Fonseca, Manoj Plakal, Frederic Font, Daniel PW Ellis, and Xavier Serra. Audio tagging with noisy labels and minimal supervision. *arXiv preprint arXiv:1906.02975*, 2019.

[2] Jonathan Foote. Automatic audio segmentation using a measure of audio novelty. In *2000 ieee international conference on multimedia and expo. icme2000. proceedings. latest advances in the fast changing world of multimedia (cat. no. 00th8532)*, volume 1, pages 452–455. IEEE, 2000.

[3] Silvia Allegro, Michael Büchler, and Stefan Launer. Automatic sound classification inspired by auditory scene analysis. In *Eurospeech, Aalborg, Denmark*, 2001.

[4] Emre Cakır, Toni Heittola, and Tuomas Virtanen. Domestic audio tagging with convolutional neural networks. *IEEE AASP challenge on detection and classification of acoustic scenes and events (DCASE 2016)*, 2016.

[5] Stanley Mneney. *An Introduction to Digital Signal Processing.* CRC Press, 2022.

[6] Xavier Durot and Jean-Bernard Rault. A new noise injection model for audio compression algorithms. In *Audio Engineering Society Convention 101*. Audio Engineering Society, 1996.

[7] Yong Xu, Qiuqiang Kong, Wenwu Wang, and Mark D Plumbley. Large-scale weakly supervised audio classification using gated convolutional neural network. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 121–125. IEEE, 2018.

[8] Il-Young Jeong and Hyungui Lim. Audio tagging system using densely connected convolutional networks. In *DCASE*, pages 197–201, 2018.

[9] Svilen Dimitrov, Jochen Britz, Boris Brandherm, and Jochen Frey. Analyzing sounds of home environment for device recognition. In *European Conference on Ambient Intelligence*, pages 1–16. Springer, 2014.

[10] Zhouyu Fu, Guojun Lu, Kai Ming Ting, and Dengsheng Zhang. A survey of audio-based music classification and annotation. *IEEE transactions on multimedia*, 13(2):303–319, 2010.

[11] Veronica Morfi and Dan Stowell. Deep learning for audio event detection and tagging on low-resource datasets. *Applied Sciences*, 8(8):1397, 2018.

[12] Theodoros Giannakopoulos. pyaudioanalysis: An open-source python library for audio signal analysis. *PloS one*, 10(12):e0144610, 2015.

[13] Daniele Barchiesi, Dimitrios Giannoulis, Dan Stowell, and Mark D Plumbley. Acoustic scene classification: Classifying environments from the sounds they produce. *IEEE Signal Processing Magazine*, 32(3):16–34, 2015.

[14] Toni Heittola, Annamaria Mesaros, Antti Eronen, and Tuomas Virtanen. Context-dependent sound event detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2013(1):1–13, 2013.

[15] S Chandrakala and SL Jayalakshmi. Environmental audio scene and sound event recognition for autonomous surveillance: A survey and comparative studies. *ACM Computing Surveys (CSUR)*, 52(3):1–34, 2019.

[16] Ella Browning, Rory Gibb, Paul Glover-Kapfer, and Kate E Jones. Passive acoustic monitoring in ecology and conservation. 2017.

[17] Zeenat Tariq, Sayed Khushal Shah, and Yugyung Lee. Speech emotion detection using iot based deep learning for health care. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4191–4196. IEEE, 2019.

[18] Emre Cakir, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Multi-label vs. combined single-label sound event detection with deep neural networks. In *2015 23rd European signal processing conference (EUSIPCO)*, pages 2551–2555. IEEE, 2015.

[19] Yuan Gong, Yu-An Chung, and James Glass. Psla: Improving audio tagging with pretraining, sampling, labeling, and aggregation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3292–3306, 2021.

[20] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. *arXiv preprint arXiv:1606.00298*, 2016.

[21] Yong Xu, Qiang Huang, Wenwu Wang, Philip JB Jackson, and Mark D Plumbley. Fully dnn-based multi-label regression for audio tagging. *arXiv preprint arXiv:1606.07695*, 2016.

[22] Seongkyu Mun, Sangwook Park, David K Han, and Hanseok Ko. Generative adversarial network based acoustic scene training set augmentation and selection using svm hyperplane. In *DCASE*, pages 93–102, 2017.

[23] Peter E Hart, David G Stork, and Richard O Duda. *Pattern classification*. Wiley Hoboken, 2000.

[24] Tuomas Virtanen, Mark D Plumbley, and Dan Ellis. *Computational analysis of sound scenes and events*. Springer, 2018.

[25] James Hillenbrand. The physics of sound sound and vibration. pages 1–44. 2016.

[26] Francesco Camastra and Alessandro Vinciarelli. Audio acquisition, representation and storage. In *Machine Learning for Audio, Image and Video Analysis*, pages 13–55. Springer, 2015.

[27] John Price and Terry Goble. Signals and noise. In *Telecommunications Engineer's Reference Book*, pages 10–1. Elsevier, 1993.

[28] Jeroen Belleman. From analog to digital. pages 281–315, 2009.

[29] H Schmickler. Time-domain and frequency-domain signals and their analysis. *arXiv preprint arXiv:2009.14544*, 2020.

[30] Dimitris G Manolakis and Vinay K Ingle. *Applied digital signal processing: theory and practice*. Cambridge university press, 2011.

[31] Meinard Muller. The fourier transform in a nutshell. *Fundamentals of Music Processing, Section*, 2:40–56, 2015.

[32] Ezeude Anayo Kingsley et al. Blind source separation using frequency domain independent component analysis, 2007.

[33] David Gerhard. *Audio signal classification: History and current techniques*. Citeseer, 2003.

[34] Peter Mckeon, Slah Yaacoubi, Nico Declercq, and Salah Ramadan. Issues concerning using mode conversion of guided waves to size defects in plates. 04 2012.

[35] Hohyub Jeon, Yongchul Jung, Seongjoo Lee, and Yunho Jung. Area-efficient short-time fourier transform processor for time–frequency analysis of non-stationary signals. *Applied Sciences*, 10(20):7208, 2020.

[36] Tom Mitchell, Bruce Buchanan, Gerald DeJong, Thomas Dietterich, Paul Rosenbloom, and Alex Waibel. Machine learning. *Annual review of computer science*, 4(1):417–433, 1990.

[37] Alexander Lerch. *An introduction to audio content analysis: Applications in signal processing and music informatics*. Wiley-IEEE Press, 2012.

[38] Romain Serizel, Victor Bisot, Slim Essid, and Gaël Richard. Acoustic features for environmental sound analysis. In *Computational analysis of sound scenes and events*, pages 71–101. Springer, 2018.

[39] Francesc Alías, Joan Claudi Socoró, and Xavier Sevillano. A review of physical and perceptual feature extraction techniques for speech, music and environmental sounds. *Applied Sciences*, 6(5):143, 2016.

[40] Karthikeyan Umapathy, Sridhar Krishnan, and Raveendra K Rao. Audio signal feature extraction and classification using local discriminant bases. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4):1236–1246, 2007.

[41] Elham Babaee, Nor Badrul Anuar, Ainuddin Wahid Abdul Wahab, Shahaboddin Shamshirband, and Anthony T Chronopoulos. An overview of audio event detection methods from feature extraction to classification. *applied artificial intelligence*, 31(9-10):661–714, 2017.

[42] Qingyang Xu, Wanqiang Zheng, Xiaoxiao Liu, and Punan Jing. Deep learning technique based surveillance video analysis for the store. *Applied Artificial Intelligence*, 34(14):1055–1073, 2020.

[43] Bhanu Prasad and SR Mahadeva Prasanna. *Speech, audio, image and biomedical signal processing using neural networks*, volume 83. Springer, 2007.

[44] Ronaldus Maria Aarts. Audio signal processing device. *Acoustical Society of America Journal*, 120(6):3445, 2006.

[45] Bhiksha Raj Ramakrishnan. *Reconstruction of incomplete spectrograms for robust speech recognition.* PhD thesis, Carnegie Mellon University, 2000.

[46] Quan Zhou, Jianhua Shan, Wenlong Ding, Chengyin Wang, Shi Yuan, Fuchun Sun, Haiyuan Li, and Bin Fang. Cough recognition based on mel-spectrogram and convolutional neural network. *Frontiers in Robotics and AI*, page 112, 2021.

[47] Hao Meng, Tianhao Yan, Fei Yuan, and Hongwei Wei. Speech emotion recognition from 3d log-mel spectrograms with deep learning network. *IEEE access*, 7:125868–125881, 2019.

[48] BZJLS Thornton. Audio recognition using mel spectrograms and convolution neural networks. 2019.

[49] Tom Bäckström, Okko Räsänen, Abraham Zewoudie, Pablo Perez Zarazaga, Sneha Das, et al. Introduction to speech processing, 2020.

[50] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. A comparison of audio signal preprocessing methods for deep neural networks on music tagging. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1870–1874. IEEE, 2018.

[51] Hari Charan Vemula. *Multiple drone detection and acoustic scene classification with deep learning.* PhD thesis, Wright State University, 2018.

[52] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning.* Cambridge University Press, 2020.

[53] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms.* Cambridge university press, 2014.

[54] Qiong Liu and Ying Wu. *Supervised Learning*, pages 3243–3245. Springer US, Boston, MA, 2012.

[55] S.Dridi. UNSUPERVISED LEARNING - A SYSTEMATIC LITERATUREREVIEW. 2021.

[56] YCAP Reddy, P Viswanath, and B Eswara Reddy. Semi-supervised learning: A brief review. *Int. J. Eng. Technol*, 7(1.8):81, 2018.

[57] Ludmila I Kuncheva. *Combining pattern classifiers: methods and algorithms.* John Wiley & Sons, 2014.

[58] S Neelamegam and E Ramaraj. Classification algorithm in data mining: An overview. *International Journal of P2P Network Trends and Technology (IJPTT)*, 4(8):369–374, 2013.

[59] Yong Xu, Qiuqiang Kong, Qiang Huang, Wenwu Wang, and Mark D Plumbley. Attention and localization based on a deep convolutional recurrent model for weakly supervised audio tagging. *arXiv preprint arXiv:1703.06052*, 2017.

[60] Kele Xu, Boqing Zhu, Qiuqiang Kong, Haibo Mi, Bo Ding, Dezhi Wang, and Huaimin Wang. General audio tagging with ensembling convolutional neural networks and statistical features. *The Journal of the Acoustical Society of America*, 145(6):EL521–EL527, 2019.

[61] Luiz Zaniolo and Oge Marques. On the use of variable stride in convolutional neural networks. *Multimedia Tools and Applications*, 79(19):13581–13598, 2020.

[62] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[63] Alessandro Maccagno, Andrea Mastropietro, Umberto Mazziotta, Michele Scarpiniti, Yong-Cheol Lee, and Aurelio Uncini. A cnn approach for audio classification in construction sites. In *Progresses in Artificial Intelligence and Neural Systems*, pages 371–381. Springer, 2021.

[64] grong Cheng, Zhiguang Qin, Chaosheng Feng, Yong Wang, and Fagen Li. Conditional mutual information-based feature selection analyzing for synergy and redundancy. *Etri Journal*, 33(2):210–218, 2011.

[65] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30, 2006.

[66] Salvador Garcia and Francisco Herrera. An extension on" statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of machine learning research*, 9(12), 2008.

[67] Nathalie Japkowicz and Mohak Shah. *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

[68] Ishan Misra, Abhinav Shrivastava, and Martial Hebert. Watch and learn: Semi-supervised learning for object detectors from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3593–3602, 2015.

[69] Julien IE Hoffman. *Biostatistics for medical and biomedical practitioners*. Academic press, 2015.

[70] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[71] Yassine Ouali, Céline Hudelot, and Myriam Tami. An overview of deep semi-supervised learning. *arXiv preprint arXiv:2006.05278*, 2020.

[72] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[73] Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. A survey on deep semi-supervised learning. *arXiv preprint arXiv:2103.00550*, 2021.

[74] Sina Ardabili, Amir Mosavi, and Annamária R Várkonyi-Kóczy. Advances in machine learning modeling reviewing hybrid and ensemble methods. In *International Conference on Global Research and Education*, pages 215–227. Springer, 2019.

[75] Erik Englesson and Hossein Azizpour. Consistency regularization can improve robustness to label noise. *arXiv preprint arXiv:2110.01242*, 2021.

[76] Kunfeng Wang, Chao Gou, Yanjie Duan, Yilun Lin, Xinhu Zheng, and Fei-Yue Wang. Generative adversarial networks: introduction and outlook. *IEEE/CAA Journal of Automatica Sinica*, 4(4):588–598, 2017.

[77] Mamshad Nayeem Rizve, Kevin Duarte, Yogesh S Rawat, and Mubarak Shah. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. *arXiv preprint arXiv:2101.06329*, 2021.

[78] Chih-Fong Tsai and Ming-Lun Chen. Credit rating by hybrid machine learning techniques. *Applied soft computing*, 10(2):374–380, 2010.

[79] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Advances in neural information processing systems*, 32, 2019.

[80] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

[81] Leo Cances and Thomas Pellegrini. Semi-supervised audio tagging with deep co-training and augmentations. 2020.

[82] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[83] Yuxuan Gu, Qixin Chen, Kai Liu, Le Xie, and Chongqing Kang. Gan-based model for residential load generation considering typical consumption patterns. 11 2018.

[84] Zhixiao Wang, Mingyu Chen, Wenyao Yan, Wendong Wang, Ang Gao, Gaoyang Nie, Feng Wang, and Shaobo Yang. Revisiting recent and current anomaly detection based on machine learning in ad-hoc networks. In *Journal of Physics: Conference Series*, volume 1288, page 012075. IOP Publishing, 2019.

[85] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013.

[86] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[87] Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. Deep co-training for semi-supervised image recognition. In *Proceedings of the european conference on computer vision (eccv)*, pages 135–152, 2018.

[88] Zhi-Hua Zhou and Ming Li. Semi-supervised learning by disagreement. *Knowledge and Information Systems*, 24(3):415–439, 2010.

[89] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[90] Yaoshiang Ho and Samuel Wookey. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, 8:4806–4813, 2019.

[91] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[92] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

[93] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[94] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[95] Rey Reza Wiyatno, Anqi Xu, Ousmane Dia, and Archy de Berker. Adversarial examples in modern machine learning: A review. *arXiv preprint arXiv:1911.05268*, 2019.

[96] Jing Lin, Laurent L Njilla, and Kaiqi Xiong. Secure machine learning against adversarial samples at test time. *EURASIP Journal on Information Security*, 2022(1):1–15, 2022.

[97] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[98] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[99] Jizong Peng, Guillermo Estrada, Marco Pedersoli, and Christian Desrosiers. Deep co-training for semi-supervised image segmentation. *Pattern Recognition*, 107:107269, 2020.

[100] Lamia H Shehab, Omar M Fahmy, Safa M Gasser, and Mohamed S El-Mahallawy. An efficient brain tumor image segmentation based on deep residual networks (resnets). *Journal of King Saud University-Engineering Sciences*, 33(6):404–412, 2021.

[101] Olarik Surinta and Thananchai Khamket. Recognizing pornographic images using deep convolutional neural networks. In *2019 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT-NCON)*, pages 150–154. IEEE, 2019.

[102] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[103] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[104] Letícia Cordeiro. Wide residual network for the tiny imagenet challenge, 2017.

[105] Max Kuhn, Kjell Johnson, et al. *Applied predictive modeling*, volume 26. Springer, 2013.

[106] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Tut database for acoustic scene classification and sound event detection. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 1128–1132. IEEE, 2016.

[107] Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European conference on information retrieval*, pages 345–359. Springer, 2005.

[108] Alireza Baratloo, Mostafa Hosseini, Ahmed Negida, and Gehad El Ashal. Part 1: simple definition and calculation of accuracy, sensitivity and specificity. 2015.

[109] Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018, 2015.

[110] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1041–1044, 2014.

[111] Frederic Font, Gerard Roma, and Xavier Serra. Freesound technical demo. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 411–412, 2013.

[112] Xifeng Dong, Bo Yin, Yanping Cong, Zehua Du, and Xianqing Huang. Environment sound event classification with a two-stream convolutional neural network. *IEEE Access*, 8:125714–125721, 2020.

[113] Sebastian Raschka, Joshua Patterson, and Corey Nolet. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, 11(4):193, 2020.

[114] Junzhe Zhang, Sai Ho Yeung, Yao Shu, Bingsheng He, and Wei Wang. Efficient memory management for gpu-based deep learning systems. *arXiv preprint arXiv:1903.06631*, 2019.

[115] Tiago Carneiro, Raul Victor Medeiros Da Nóbrega, Thiago Nepomuceno, Gui-Bin Bian, Victor Hugo C De Albuquerque, and Pedro Pedrosa Reboucas Filho. Performance analysis

of google colaboratory as a tool for accelerating deep learning applications. *IEEE Access*, 6:61677–61685, 2018.

[116] Casper Solheim Bojer and Jens Peder Meldgaard. Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*, 37(2):587–603, 2021.

[117] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[118] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[119] Shanshan Yao, Baoning Niu, and Jianquan Liu. Enhancing sampling and counting method for audio retrieval with time-stretch resistance. In *2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM)*, pages 1–5. IEEE, 2018.

[120] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. Advertorch v0. 1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

[121] Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020.

[122] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[123] Wes McKinney et al. pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9, 2011.

[124] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03):90–95, 2007.

[125] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[126] Elizabeth D Liddy. Natural language processing. 2001.

[127] Masashi Sugiyama. *Introduction to statistical machine learning*. Morgan Kaufmann, 2015.

[128] Salvador García, Alberto Fernández, Julián Luengo, and Francisco Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information sciences*, 180(10):2044–2064, 2010.