Université de BLIDA 1

Faculté des Sciences

Département d'Informatique

MASTER THESIS

Option: Ingénierie des Logiciels

EXPERIMENTAL DESIGN AND ANALYSIS OF AUDIO TAGGING SYSTEMS: CASE STUDIES

By

Bouchra AMIROUCHE

Ilhem MOUSSA

In front of a jury composed of:

Ms. Siham BACHA	President
Ms. Dalila GUESSOUM	Examiner
Ms. Hadjer YKHLEF	Supervisor
Mr. Farid YKHLEF	Supervisor

2019/2020

ABSTRACT

The goal of general-purpose audio tagging is to create systems capable of recognizing a variety of sounds. Including musical instruments, vehicles, animals, sounds generated by some sort of human activity etc. The motivation for research in the field of artificial sound understanding can be found in potential applications such as security, healthcare (hearing impairment), improvement in smart devices and various music related tasks. The main contribution of this work entails conducting extensive studies and comparisons between audio tagging systems using a huge dataset made of 11 073 audio recordings. In this thesis, we have carried out two sets of experiments. First, we have examined Deep Convolutional neural networks (CNN) and 3 of its variants (Convolutional Recurrent Neural Network (CRNN), Gated Convolutional Recurrent Neural Network (GCRNN) and Gated Convolutional Neural Networks (GCNN)) using Log-Mel Spectrogram features. We have supported our analysis and discussion with numerous statistical tests to analyze and compare the effect of the above-mentioned features and models on the tagging performance. Our experimental findings indicate that our systems capture diverse set of sound events, with various confidences. Moreover, Convolutional Recurrent Neural Network (CRNN) significantly outperforms the other models. Second, motivated by the fact that the individual models produce diverse predictions, we have investigated the effect of ensemble learning using a technique known as stacking. Our analysis shows that stacking provides a proper amalgamation of the individual learners, resulting in better handling the diverse nature of the events.

Keywords: Audio Tagging, Deep Learning, Machine leaning, Ensemble Learning, Stacking, Feature Extraction, Statistical Tests.

RESUME

L’étiquetage audio est une technique qui permet de créer des systèmes capables d’identifier un ensemble de sons tel que : les sons des instruments musicaux, les sons générer par une activité humaine et le son des véhicules etc. Ce qui a motivé la présente recherche, c’est sa potentielle application dans divers domaines tel que la sécurité, la santé ainsi que l'amélioration des appareils intelligents. La principale contribution de ce travail consiste à mener une étude approfondie qui consiste à analyser et comparer les performances de plusieurs systèmes d’étiquetage audio en utilisant une base de données volumineuse constituée de 11 473 enregistrements audio. Dans ce travail, nous avons effectué deux séries d'expériences: dans un premier temps, nous avons comparé les performances de nos systèmes selon les caractéristiques du log-mel spectrogramme en examinant le réseau de neurones convolutifs (CNN) et trois de ses variants : le réseau de neurones convolutifs récurrents (CRNN), le réseau de neurones convolutifs récurrents à portes (GCRNN) et le réseau de neurones convolutifs à portes (GCNN). Nous avons appuyé notre analyse dans ce document par des tests statistiques afin d’interpréter et de comparer les résultats obtenus. Cela nous a permis de démontrer que nos systèmes capturent plusieurs types d’évènements sonores. De plus, la performance du réseau de neurones convolutifs récurrents (CRNN) a surpassé les autres. Deuxièmement, motivé par le fait que les modèles individuels produisent des prédictions diverses. Nous avons étudié l'effet de l'apprentissage ensembliste en invoquant une technique connue sous le nom de «stacking». Notre analyse démontre que cette méthode a une capacité de généralisation considérablement meilleure que les classifieurs uniques. Plus important encore, cette dernière a fourni une fusion appropriée de leurs diverses prédictions, ce qui a permis de mieux gérer la diversité des évènements.

Mot clés : Etiquetage Audio, Apprentissage Profond, Apprentissage automatique, Apprentissage Ensembliste, Extraction des Caractéristiques, Testes Statistique.

ملخص

تهدف أنظمة وضع العلامات الصوتية العامة إلى إنشاء أنظمة قادرة على التعرف على مجموعة متنوعة من الأصوات بما فيها الآلات الموسيقية، المركبات، الحيوانات والأصوات الناتجة عن أنشطة بشرية مختلفة... إلخ. الدافع الأساسي للبحث في مجال فهم تحليل الصوت يكمن في التطبيقات المحتملة مثل الأمن، الرعاية الصحية (ضعف السمع)، تحسين الأجهزة الذكية ومختلف المجالات المتعلقة بالموسيقى. المساهمة الرئيسية لهذه الأطروحة هي إجراء دراسات ومقارنات مكثفة بين أنظمة وضع العلامات الصوتية باستخدام قاعدة بيانات تتضمن على تسجيلات صوتية من مجالات متعددة ومختلفة. في هذه الأطروحة، قمنا بتنفيذ مجموعتين من التجارب، حيث قمنا اولا بفحص أداء أنظمتنا الناتجة عن استعمال نموذج الشبكات العصبونية الالتفافية عميقة (CNN) و 3 نماذج ممتددة منها ; نموذج الشبكات العصبونية الالتفافية التكرارية (CRNN)، نموذج الشبكات العصبونية الالتفافية التكرارية البوابية (GCRNN) ونموذج الشبكات العصبونية الالتفافية البوابية (GCNN) بالإضافة إلى استعمال تقنية استخراج خصائص "Log-Mel Spectrogram ". من أجل تقييم أداء هذه الأنظمة دعمنا تحليلاتنا بعدة اختبارات إحصائية. حيث تشير نتائجنا التجريبية أن أنظمتنا تلتقط أحداث صوتية متنوعة. علاوة على ذلك، الشبكات العصبونية الالتفافية التكرارية عرضت دقة وفعالية في الأداء مقارنة بالنماذج الشبكات العصبونية الأخرى. في المجموعة الثانية من التجارب، وبناءا على أن النماذج الفردية حققت نتائج متنوعة. لقد تطرقنا لدراسة تأثير المصنفات المتعددة على أداء أنظمة وضع العلامات الصوتية باستخدام تقنية "Stacking". كشفت نتائج هذه الدراسة أن "Stacking" يوفر اندماج ملائم لنتائج النماذج الفردية مما أدى إلى التعامل بشكل أفضل مع الطبيعة المتنوعة للأحداث الصوتية.

كلمات المفاتيح: أنظمة وضع العلامات الصوتية، تعلم عميق، التعلم الآلي، المصنفات المتعددة، تقنيات استخراج خصائص، اختبارات إحصائية.

ACKNOWLEDGMENTS

First and foremost, we would like to take this opportunity to extend our thanks to our supervisors Hadjer YKHLEF and Farid YKHLEF for creating a stimulating and positive working environment even in this difficult and challenging times, without their help, encouragement, support and guidance it would be impossible for us to complete this thesis. A special thanks also goes to the examiners for the time they spent on carefully reading this thesis and for their constructive comments.

Moreover, this work could not be completed without the effort and cooperation of our group. Last but not least we would like to thank our family and friends for their constant source of support.

Finally, we would like to thank Google for making the GPU computing accessible to masses for free with Colaboratory.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATION AND ACRONYMS

BPPTT	: Backpropagation Through Time
BN	: Batch Normalization
CNN	: Convolutional Neural Network
CRNN	: Convolutional Recurrent Neural Network
DCT	: Discrete Cosine Transform
DFT	: Discrete Fourier Transform
FFT	: Fast Fourier Transform
FN	: False Negative
FP	: False Positive
GAP	: Global Average Pooling
GCNN	: Gated Convolutional Neural Network
GCRNN	: Gated Convolutional Recurrent Neural Network
GLU	: Gated Linear Units
GRU	: Gated Recurrent Units
LSTM	: Long Short Term Memory Network
MAP	: Mean Average Precision
MFCC	: Mel Frequency Cepstral Coefficients
RNN	: Recurrent Neural Network
TP	: True Positive
TN	: True Negative
VGG	: Visual Geometry Group

INTRODUCTION

1. Context and problem statement

Nowadays, machines are able to efficiently handle a wide variety of multimedia content including images, audios, videos etc. As the amount of data is constantly increasing, the analysis and the recognition of certain patterns out of it has become of paramount importance. Sound analysis [1] is a subfield concerned about execution of such tasks for audio signals such as speech, music, acoustic events etc.

We can classify sound analysis tasks into 3 categories: Acoustic Scene Classification, Audio Tagging and Event Detection. In scene classification, the goal is to categorize an audio recording into one of a set of (predefined) categories; for instance: home, street, and office. Similarly, audio tagging assigns a given audio recording with one or several pre-defined tags. The motivation behind audio tagging systems is to foster research towards more general machine listening systems capable of recognizing and discerning a wide range of acoustic events and audio scenes. Furthermore, there is a large amount of user-generated audio content that is available on the web, which can be a resource of great potential for sound recognition related research. Audio tagging has many applications such as audio information retrieval [2], audio classification [3], acoustic scene recognition [4], industry sound [5] and music tagging [6]. Finally, event detection locates in time the occurrences of a specific type of sound or sounds, either by finding each instance when the sound(s) happen or by finding all the temporal positions when the sound(s) are active. Here, the term sound event refers to a specific sound produced by a distinct physical sound source, such as a car passing by, a bird signing, or a doorbell. By contrast, the term of sound scene refers to the entirety of sound that is formed when sounds from various sources, typically from real scenarios, combine to form a mixture.

The process of Audio Tagging consists of two main stages (Figure 1): **Feature Engineering** and **Machine Learning**. First, the main role of sound pre-processing step is to enhance certain characteristics of the incoming audio file in order to optimize audio analysis performance in the later phases of the analysis system. Then, feature extraction is applied on the resulting preprocessed data; here, we divide the audio signal into equal frames in order to perform feature extraction and obtain a feature vector per frame. Each vector of data is associated to its corresponding event label. The most common types of features used in the literature include: Mel-frequency cepstral coefficients [7], Log Mel band Energy [8] and spectral centroid. Next, the classification model takes the feature vector of each frame and outputs the event presence predictions for each sound event class. It worth underscoring a classifier or learner enables us to predict the events labels, known as class labels, present in a sound recording. Many classification algorithms have been introduced in the literature, such as Deep Neural Networks [9], Gaussian Mixture Models (GMMs) [10], and Support Vector Machine [11].

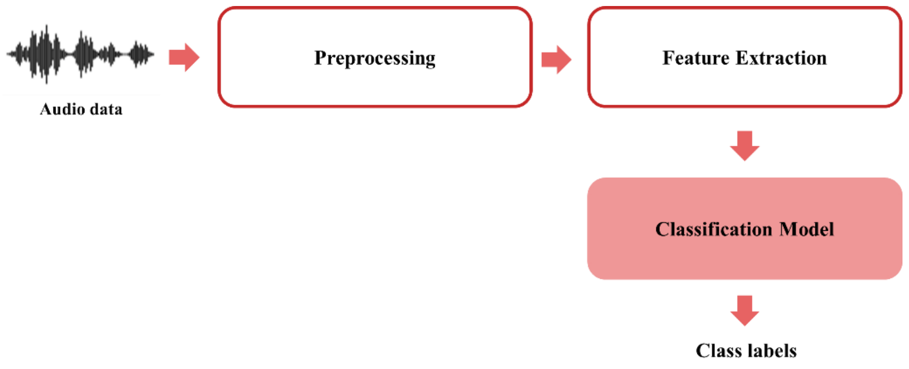


Figure 1: The mechanism of audio tagging system.

2. Contributions

With the increasing attention geared towards multimedia content, the research community has recently become more motivated to perform extensive studies and comparisons between sound analysis systems. The design and evaluation of such systems is actually a more complicated task, and should be conducted properly in order to ensure significance of results (i.e. avoid deriving conclusions affected by chance). To date, the problem of Audio Tagging has been addressed by the research community using various methodologies. Most of them have focused on extracting relevant features and finding suitable classifiers to improve the overall performance. Nevertheless, *only a few attempts have reviewed and studied the proper manner required for the design and analysis of Audio Tagging systems* [12] [13] [14]. Furthermore, several seminar papers have been published recently; most of them have invoked recent and hybrid deep learning architecture, for instance: **Convolutional Recurrent Neural Network** and **Gated Convolutional Recurrent Neural Network** [15]**.** Therefore, an adequate review of the newly proposed techniques has become necessary. Additionally, extensive comparisons among these methods should be conducted in order to acquire the best practices for addressing the Audio Tagging task. In what follows, we summarize our main contributions:

1. We have carried out extensive experiments on the FSDKaggle2018 dataset [16]. This dataset presents several challenges. It contains user-generated audio clips retrieved from **Freesound** [17], which are very diverse in terms of acoustic content, recording techniques, clip duration, etc. Furthermore, these audio clips could feature incomplete

and inconsistent user-provided metadata. Some audio clips were manually labeled, while a smaller set of clips were automatically categorized on the basis of their user-provided metadata. Therefore, the dataset is unbalanced and contains a large fraction of reliable annotations that can be trusted. It also contains an amount of non-verified annotations that could include a small amount of label noise.

2. To cope with the above challenges, we have designed our Audio Tagging systems using well-known deep neural network architectures, which have been successfully used in audio-related tasks [15] [3]. Specifically, we have studied Convolutional Neural Networks (CNN), Convolutional Recurrent Neural Network (CRNN), Gated Convolutional Recurrent Neural Network (GCRNN) and Gated Convolutional Neural Networks (GCNN). In addition, we have supported our analysis and discussion with numerous statistical tests.
3. We have invoked Stacking ensemble learning technique by taking the noisy label into account. The aim of Stacking is to combine the prediction result from different models in order to improve the accuracy and robustness of the system. Furthermore, to address the effect of the non-verified examples on the performance of the system, we have used a combination re-weight strategy along with stacking to handle the potential noisy label of the non-verified annotations in the dataset.
4. We have reviewed recent Audio Tagging schemes and discussed the major steps involved in the proper design and evaluation of such systems.

3. Thesis structure

This thesis consists of **two primary parts.** The first part covers the state-of-the-art notions that are necessary for understanding the ideas developed in this thesis. **Chapter 1** is also divided into two parts; the first one gives an overview of acoustic features used to represent audio signals. Specifically, we present the different feature extraction techniques that are frequently used in literature as for the second part of this chapter we review some relevant classification concepts, providing a brief description of the supervised classifiers, evaluation metrics and statistical tests invoked in this work. In **Chapter 2**, we describe the architecture of some basic and hybrid deep learners. The second half of this thesis describes the methodology that we have followed for comparing Audio Tagging systems. We provide in **Chapter 3** detailed description of the experimental setup, including preprocessing, feature extraction and parameters setting. In **Chapter 4**, we present the obtained results through performance tables and plots. Finally, we conclude by summarizing the contributions of this thesis, the lines of limitations and future work.

PART I: FUNDAMENTALS OF AUDIO TAGGING

In this part we explain the notions that are necessary for understanding the ideas developed in this thesis. It is composed of two chapters. Chapter 1 is divided into two main parts the first one gives an overview of acoustic features used to represent audio signals. Specifically, we describe data required for the development of Audio Tagging systems and highlight the importance of feature engineering to transform the signal into a suitable representation. Furthermore, we present the different feature extraction techniques that are frequently used in literature as for the second part of this chapter we review some relevant concepts of classification providing a brief description of the fundamentals of classification, feature selection techniques, evaluation metrics and statistical tests invoked in this work. In Chapter 2, we present the main deep neural network used in our work. In addition, we provide a brief description of a data augmentation technique and ensemble method. Moreover, we discuss some empirical and theoretical findings on the differences of the architectures presented in this chapter. Finally, we have discussed the challenges related to Audio Tagging research.

CHAPTER 1: GENERALITIES ON AUDIO TAGGING

1.1 Introduction

Audio Tagging is about predicting the types of sound events occurring in audio clips. It is comprised of two main stages: (1) Feature Engineering, (2) Machine Learning. Feature engineering consists of transforming the signal into a representation which maximizes the sound recognition performance of the analysis system. The acoustic features provide a numerical representation of the signal content relevant for machine learning, characterizing the signal with values which have connection to its physical properties, for example, signal energy, its distribution in frequency, and change over time. On the other hand, machine learning is an application of artificial intelligence that provides systems the ability to automatically learn and improve from data. The result of the learning process is known as machine learning model. This latter takes as an input a set of features extracted from a sound event and assigns a label to it.

This chapter is divided into **two primary parts**: In the first part, we begin by introducing the process of sound acquisition in Section 1.2. Then, in Section 1.3, we highlight the importance of feature engineering, we briefly discuss time and frequency representations and explain the different feature extraction techniques widely employed in the literature and some preprocessing techniques. Finally, in Section 1.4 we shortly describe some existing datasets for sound analysis. In the second half of this chapter, we first provide a short introduction to the relevant concepts of classification in Section 1.5. Then, in Section 1.6 we present model evaluation techniques. In Sections 1.7 and 1.8 we explain feature selection and statistical tests. Finally, in Sections 1.9 and 1.10, we conclude this chapter by reviewing related work on Audio Tagging research and summarizing the main concepts that we have learned.

PART I: FEATURE ENGINEERING

1.2 Sound Acquisition

Data acquisition is an important stage of developing an Audio Tagging system, as its performance highly depends on the data in the process [1]. Essentially, the aim is to collect as realistic as possible acoustic signals in conditions which are as close as possible to the intended target application. **The metadata** should include a ground truth information which is often manually annotated during the data collection; metadata can be defined as “data about data”. In the case of audio, it usually refers to textual information that is used to describe and to index an audio file or a segment.

Collected data should also have sufficient number of representative examples of all sound classes for increasing the generalization ability of the acoustic models. The audio must represent the modeled phenomena such that the models learned from it will represent the variability in the acoustic properties expected in the application. The presence of all categories relevant to the Audio Tagging task (Figure1.1) will provide the required coverage, making a dataset suitable for the given task. These are the main properties that a well-built dataset should have:

• **Coverage:** The dataset should contain as many different categories as are relevant to the task.

• **Variability:** For each category, there should be examples with variable conditions of generation, recording, etc.

• **Size:** For each category, there should be sufficiently many examples. Otherwise, the training of a system results in a weak model.

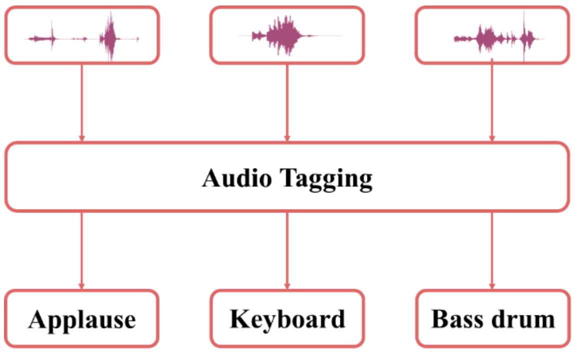


Figure 1.1: System input and output characteristics for Audio Tagging [1].

1.3 Time and frequency representation

1.3.1 Frequency

Frequency is the measurement of the number of times that a repeated event occurs per unit of time. The frequency of **wave-like** patterns including sound expresses the number of cycles of the repetitive waveform per second. For humans, hearing is limited to frequencies between about 20 Hz and 20 000 Hz [18].

1.3.2 Fourier transform

Signal is defined as any physical quantity that varies with time. It conveys information in its patterns of variation. The manipulation of this information involves the acquisition, storage, transmission, and transformation. In order to find the different frequencies that are present in a signal we apply the Fourier transform [19].

The Fourier analysis is the main mathematical tool which 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. It is used to decompose a signal into sinusoidal elements. Each sinusoid represents a frequency, which makes it possible to obtain information on the frequency distribution rather than a temporal distribution. The resulting sinusoids of Fourier Transform on a signal represented as a function of time is a complex value, whose imaginary part represents the phase off-set of the pure sinusoid and its absolute value represents value of the corresponding frequency component.

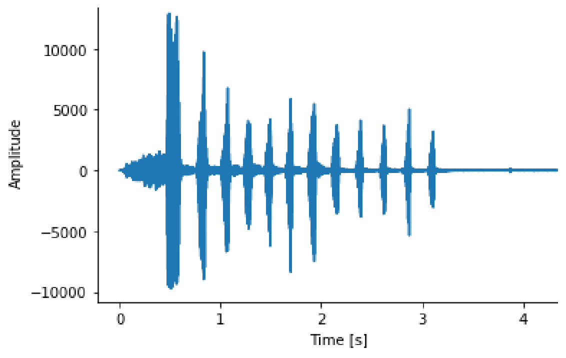
The exact form of the Fourier transform used to determine the spectrum from the discrete time signals is known as the Discrete Fourier Transform (DFT).

The mathematical equation of the DFT is:

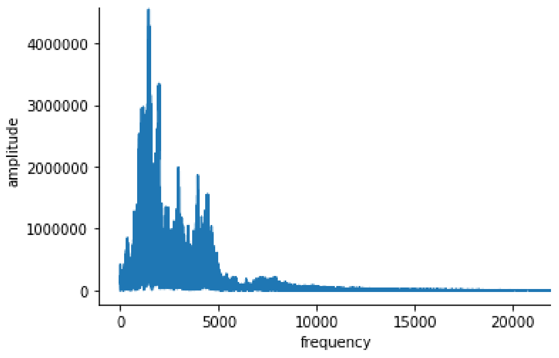
$$\mathrm{X}(\mathrm{k}_i) = \sum_{n=0}^{N-1} x(n_i)e^{-2j\pi i \frac{kn_i}{N}}, \tag{1.1}$$

where K is a set of possible frequencies and N the total number of samples in a given sound signal, furthermore, $K_i \in K$ denotes the i^{th} frequency and $n_i \in N$ represents the i^{th} sample of the signal. We define $x(k_i)$ as the amount of the i^{th} frequency in the signal and $x(n_i)$ as the amplitude of the signal at the i^{th} sample.

Here are two plots that show the effect of the FFT function applied to a simple raw audio waveform, it shows the frequency domain representation of a time domain signal (Figure 1.2).



(a) Raw audio waveform



(b) Frequency domain representation

Figure 1.2: Frequency representation of a raw audio waveform.

It is worth mentioning that the DFT algorithm has a complexity of $O(N^2)$, whereas, the Fast Fourier Transform (FFT) implementation has a quasi-logarithmic complexity $O(N\ log_2\ N)$ [19]. The savings in terms of computation are enormous, it is for this reason that FFT is commonly used in practice.

1.3.3 Audio Processing

In order to develop a robust and appropriate signal representation, audio is prepared and processed for machine learning algorithms in the audio processing phase of the overall system design. This phase consists of two main stages **preprocessing**, and **acoustic feature extraction**.

A Preprocessing

Pre-processing is applied to the audio signal before acoustic feature extraction if needed. The main role of this stage is to enhance certain characteristics of the incoming signal in order to optimize audio analysis performance in the later phases of the analysis system. This is achieved by diminishing the effects of noise [20], emphasizing the target sounds in the signal [1] or segmenting the original audio signal into audio and silent events to be used in feature extraction [20].

Silence removal

Silence removal is used to eliminate the silent portion of the audio signal *x*. The process of Silence removal consists of dividing an input signal into small segments (frames) and thresholding the root mean square (RMS) energy of those frames. The total length of each individual segment is equal to product of time duration and sampling frequency of segment (Fs).

$$\text{Segment}_{\text{length}} = \text{Segment}_{\text{duration}} * \text{Fs} \tag{1.2}$$

The RMS value of each segment is calculated and compared with threshold value R_{th}. RMS value of each individual segment can be calculated from equation (1.3)

$$\text{RMS}_{\text{segment}} = \sqrt{mean(segment^2)} \tag{1.3}$$

If $\text{RMS}_{\text{segment}}$ of individual segment is less than R_{th} then the segment is removed. The function of silence removal block is given in equation (1.4).

$$f(x) = \left\{ \begin{matrix} \text{RMS}_{\text{segment}} \le R_{th}\ \ , silent\ segment \\ \text{RMS}_{\text{segment}} > R_{th}\ \ , non\ silent\ segment \end{matrix} \right\} \tag{1.4}$$

Figure 1.3 shows the silence removal process on an audio file, which is 13s in length. The non-silent sections that are extracted encased in a black rectangle.

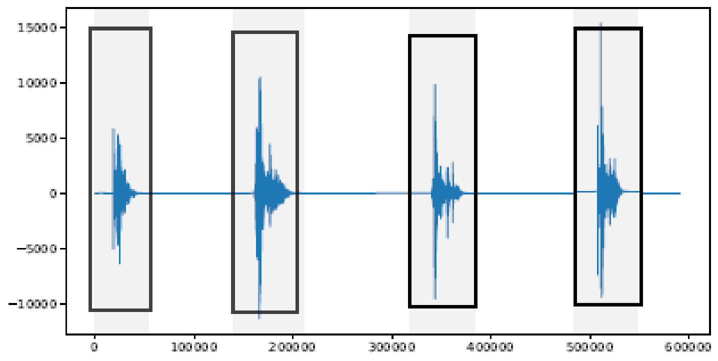


Figure 1.3: Silence removal process.

B Feature extraction

Feature extraction is an important signal processing task. It refers to the process of computing a numerical representation of the acoustical signal. The representation can be used to characterize the audio segment with values which have connection to its physical properties; for example, **signal energy**, **its distribution in frequency**, **and change over time t**. The role of feature extraction is to transform the signal into a representation which maximizes the sound recognition performance of the analysis system. It also requires less amount of memory and computational power than direct use of audio signal in the analysis [1].

The process of feature extraction is similar for many types of acoustic features used in analysis. It consists of frame blocking, windowing, spectrum calculation, and other computation depending on which type of feature extraction is being used. Figure 1.4 depicts the processing pipeline for feature extraction.

Traditional methods for spectral evaluation are reliable in the case of a stationary signal (i.e. a signal whose statistical characteristics are invariant with respect to time). However, audio signals are most of the time non-stationary throughout the whole audio recording, but stationary within short time frames [1]. For this reason, we use frame blocking and windowing to be able to use the audio signal and interpret its characteristics in a proper manner.

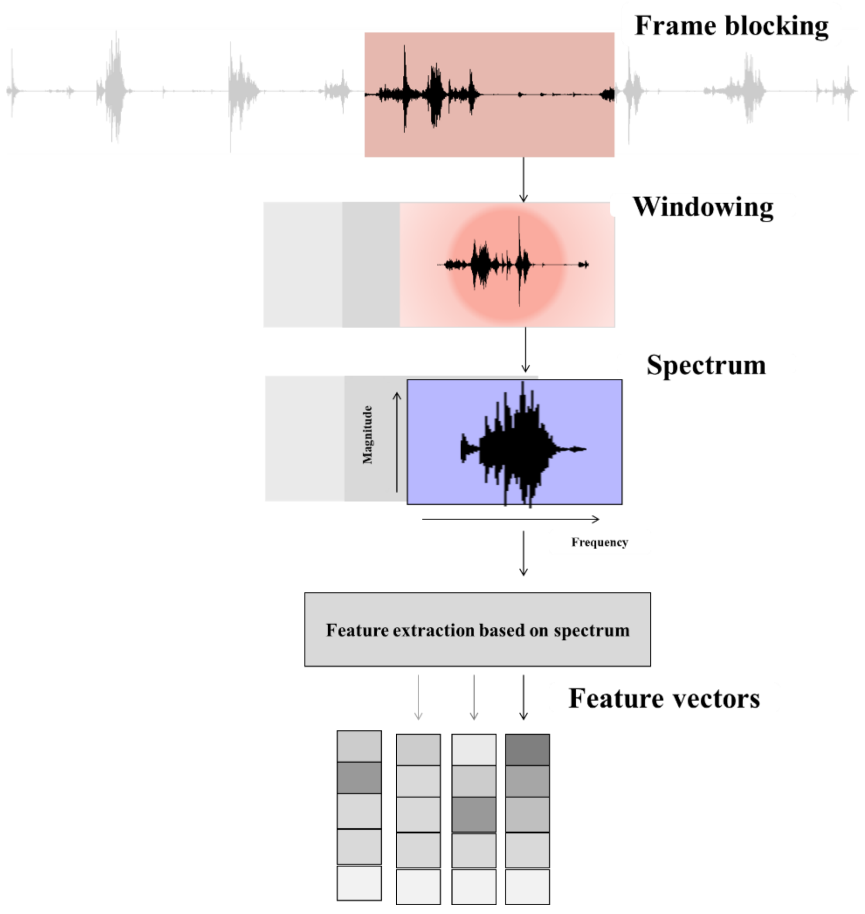


Figure 1.4: **The processing pipeline of feature extraction [1].**

Frame blocking consists of decomposing the audio signal into a series of overlapping frames. These frames have to be short enough so that it can reasonably be assumed to be stationary [1]. The selection of frame length is dependent on the machine hearing task at hand it usually varies between 20 and 60 ms [1].

Windowing is often applied after framing [1] in order to avoid the discontinuities at the borders of the frame which would cause distortions in the spectrum (corrupt the frequency spectrum estimation). The windowing process consists of multiplying each frame with a window function; hence, attenuating the signal near the edges and emphasizing the central portion. Hamming, Hann and Blackman functions are often used for windowing [21].

Hamming window (raised cosine window): the role of the hamming window is to minimize the spectral distortion by using the window to taper the signal to zero at the beginning and end of each frame, it is defined as:

$$w(m_i) = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi m_i}{M}\right), 0 \leq m_i \leq M \\ 0, othrwise \end{cases}, \tag{1.4}$$

where M is the total number of samples in each frame and m_i the i^{th} sample of the frame. The resulting value $w(m_i)$ represents the windowed value.

The Figure below shows the shape of this function:

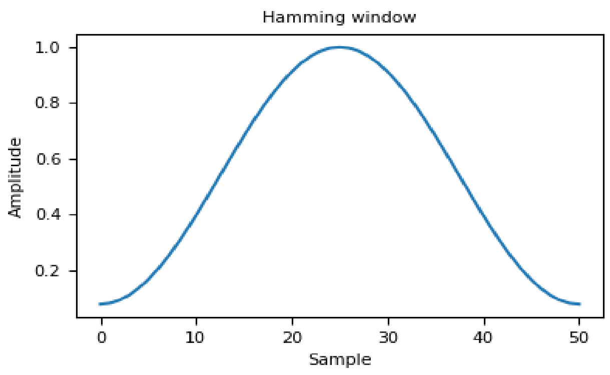


Figure 1.5: Hamming window [22].

Acoustic features can be divided into 3 main categories: temporal features, spectral features and perceptual features (prosodic features). In what follows, we describe some well-known features from each category.

SPECTRAL FEATURES

LOG-MEL SPECTROGRAM represents an acoustic time-frequency representation of a sound. In the calculation of Log-Mel Spectrogram, firstly Fast Fourier Transform is calculated over pre-processed audio signal.

The **filter bank** is used to map its spectral amplitude to the Mel-scale of the perceptual excitation, and the mel filter bank converts the spectrum to the mel spectrum. Mel-scale is based on the perception of human hearing frequencies [23]. Thus, the Mel-scale is used to measure the tone of a subjective frequency or pitch.

The filter bank energy is obtained after mel filtering. Finally, the logarithmic conversion of the mel energy is calculated and then the Log Mel Spectrum is generated from the filter bank. The flow of Log-Mel spectrogram extraction is shown in Figure 1.6.

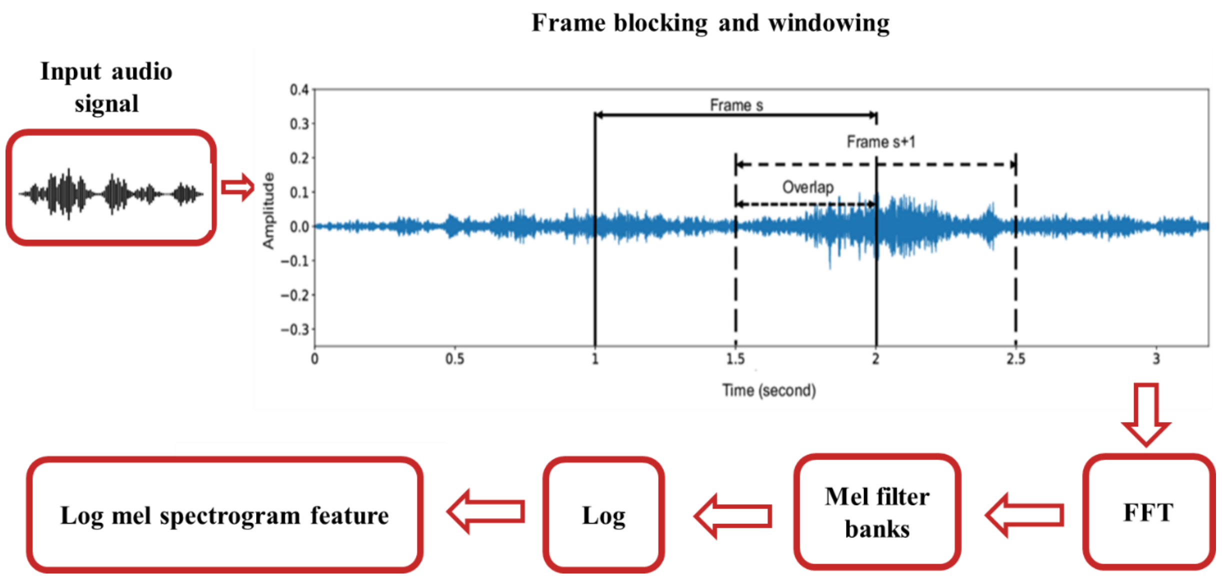


Figure 1.6: **Flow of Log-Mel Spectrogram extraction.**

MEL-SCALE as shown in Figure 1.7 is the psycho acoustic representation of frequency in linear scale processing. Stevens, Volkmann, and Newman in 1937 proposed a unit of pitch called 'Mel' [24] [25]. 'Mel' is defined as the perceptual scale of pitches judged by listeners to be equal distance from each other. During the series of experiments, it was observed that when the frequency of the signal is less than 1000Hz, human auditory system perceives signals on a linear scale and for the frequency, over 1000Hz it was recognized on a logarithmic scale. The essence of Mel-scale is to bring this feature into perspective.

Converting frequency domain to mels domain is done using formula:

$$\begin{aligned} mel &= 2595 \log\left(1 + \frac{frequency}{700}\right) \\ frequency &= 700(10^{mel/2595} - 1) \end{aligned} \tag{1.5}$$

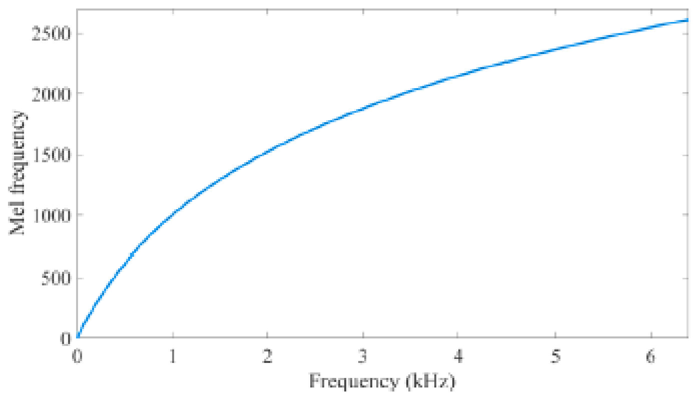


Figure 1.7: **Kilo Hertz vs Mel-scale** [22].

MEL FREQUENCY CEPSTRAL COEFFICIENTS (MFCC) is a type of cepstral representation of audio signals [26]. The steps involved for their extraction are similar to the **Log-Mel Spectrograms** steps but with an additional step that consists of computing the Discrete Cosine Transform (DCT).

DCT is a mathematical technique applied to the Log-Mel Spectrogram resulting in Mel Frequency Cepstral Coefficients. The operation of DCT is similar to DFT, and the critical difference is unlike DFT, DCT consists of only cosine terms which are real.

PERCEPTUAL FEATURES

Prosodic features, or perceptual frequency features, indicate information with semantic meaning in the context of human listeners. Therefore, they are organized according to semantically meaningful aspects of sounds including pitch, fundamental frequency, loudness, intensity and sharpness.

- **Loudness** is a psychoacoustic property of the sound. It represents our human perception of how loud or soft sounds of various intensities are. The loudness of a sound is subjective, it varies from person to person and is measured by sone and phon units [27].
- **Pitch** is a perceptual property of sounds that allows their ordering on a frequency-related scale. More commonly, pitch is the quality that makes it possible to judge sounds as «higher» and «lower» in the sense associated with sound recording.
- **Sharpness** can be interpreted as a spectral centroid based on psychoacoustic principle. It is commonly estimated as a weighted centroid of specific loudness [28].

TEMPORAL FEATURES

Temporal features are directly extracted from the audio raw data without any transformation. Such features normally suggest a simple tactic to investigate audio signals. Although, it is generally necessary to combine them with spectral features. Representative instances of temporal features are: zero-crossing rate, amplitude-based features, and power-based features [27].

OTHER APPROACHES

Alternative cepstral decompositions can be obtained similarly to MFCC from other frequency-domain representations. This had led to the introduction of features such as the Linear Prediction Cepstral Coefficients (LPCC) based on LPC coefficients [29], the Gammatone Feature Cepstral Coefficients (GFCC) [30] or Constant-Q Cepstral Coefficients (CQCC) [31]. None of these features are as popular as the MFCC but GFCC. For example, have been applied to sound scene analysis [32].

1.4 Datasets

Recently, general-purpose sound event recognizers have gained attention [33]. In this case, a wide range of sound events are considered, not tied to a specific domain. However, the majority of available datasets are domain specific and are usually small in size [33]. In addition, the existing general-purpose datasets often contain many unlabeled data sounds, which can make the learning process a challenging task.

Various data resources have been gathered for the sound analysis task; some of these are provided in Table 1.1.

Table 1.1: Various audio analysis datasets.

	Dataset name	Classes	Examples	Clip length	Dataset duration	Ref
Domain specific	**CHIME-Home**	7 (balanced)	6137	4s	≈6.82h	[34]
	GTZAN (2002)	10 (balanced)	1000	30s	8.33h	[35]
	ESC-50	50 (balanced)	2 000	5s	2.78h	[36]
	TUT Acoustic Scenes 2016	15 (balanced)	1560	30s	13h	[37]
	UrbanSound8k	10 (balanced)	8732	≤4s	8.75h	[38]
General purpose	**FSDKaggle2018**	41 (unbalanced)	11073	≤ 30s	18h	[16]
	Audio set 2017	525 (unbalanced)	≈2.1M	10s	≈5833h	[33]
	FSDKaggle2019	80 (unbalanced)	29266	≤ 30s	≈103.4h	[39]

Part II: Machine learning for Audio Tagging

1.5 Fundamentals of classification

Classification belongs to the category of Supervised Learning, where the input data is **labeled**. It is the process of predicting the class labels of a given data point (also called sample or instance), the data point is characterized by a **feature vector** $x \epsilon X$ and by its class label $y \epsilon Y$ [40].The classification algorithms take in a set of m data samples of input-output association $\Gamma\{(x_1, y_1), (x_2, y_2), \dots \dots, (x_m, y_m)\}$, where $x_i\ \epsilon X$ and $y_i \epsilon Y$, and learns a mapping function f from a feature vector $x \epsilon X$,some parameters τ and produces an output $\hat{y}$.

$$\hat{y} = f(x, \tau) \tag{1.6}$$

Furthermore, in Supervised Learning we can distinguish **single-label classification** and **multi-label classification**.

- **Single-label classification** when the classifier learns from a set of samples that are associated with a single label. It can be further divided into two categories based on the number of classes in a set of labels: binary and multi-class classification.

- **Multi-label classification** is different from the traditional single label classification. It refers to a task of associating each learning example with multiple labels the output is a vector of N labels. The multi label classifier associates an instance with one of 2^N possible output vectors. A common approach to multi label classification is to preform problem transformation, whereby a multi label problem is transformed into one or more single label (i.e. binary or multi-class) problems and the single label predictions of the single-label classifiers are transformed into multi label predictions. Although this method is easy to implement, it can be computationally inefficient when the number of classes is large [41]. Instead of this approach several studies [21], [42] has explored deep learning method (more details can be found in Chapter 2) to address the multi -label classification problem. Both methods are illustrated in Figure 1.8.

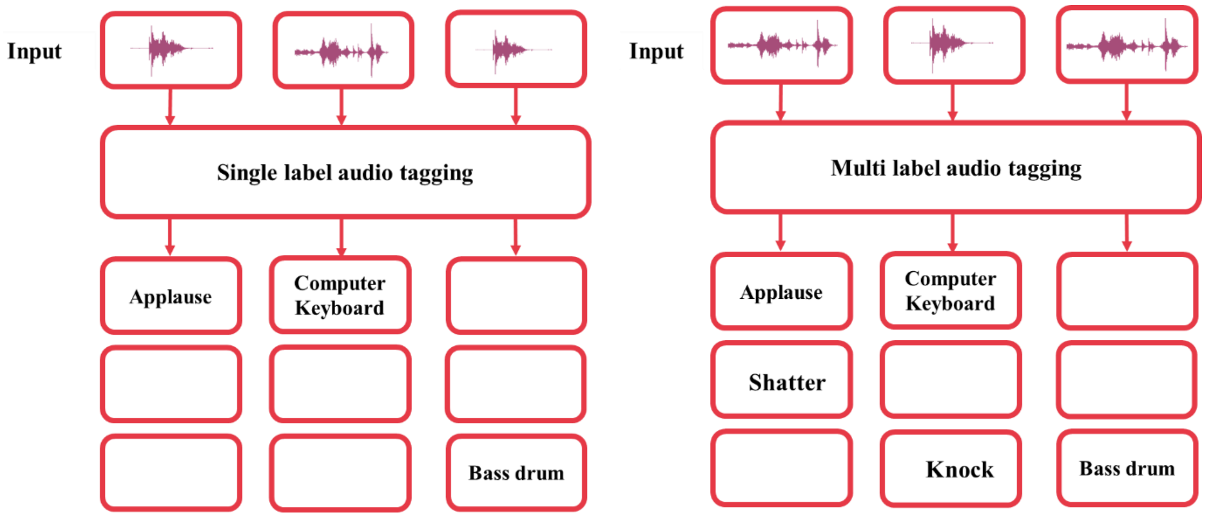


Figure 1.8: **System input and output characteristics for single-label and multi-label Audio Tagging systems** [28].

Usually a classifier is seen as a two-step algorithm: training stage and testing stage. The first stage whereby the model learns a hypothesis from the training data. Learning is the process of optimizing the loss function that calculates the difference between the actual and the predicted outputs. The model is updated according to a learning algorithm so that the loss value is minimized and to increase the generalization ability of the model (i.e. the ability to act properly on unseen samples). The widely used optimization approach for supervised learning, namely gradient descent (more detail in Section 2.3.2). In the second stage, the resulting model is used to predict the class label of unseen examples drawn from the testing set. Numerous learning models have been introduced by the machine learning community, such as **Neural Networks** [21], **Adaboost** [43] and **Support Vector Machine** [9].

Audio tagging aims to assign one or a set of tags to a clip. In machine learning terminology tagging would be equivalent to multi-label classification [29]. However, in the scope of this thesis we will focus on building a single-tag Audio Tagging system, and hence it is considered to be multi-class classification problem. We will focus on neural networks with deep architecture due to their effectiveness when dealing with the Audio Tagging problem. Chapter 2 provides an extended treatment on some widely used deep neural networks architectures.

It is widely acknowledged when a model fits the training data perfectly, it usually leads to poor generalization ability [44]. This problem is known as overfitting. To cope with this shortcoming, data augmentation is applied with deep neural network architectures [28]. Mix-up technique has been extensively invoked for building sound analysis systems, particularly Audio Tagging [45], [46], [47]. More details can be found in Section 2.10.

We measure the quality of the predictions in multiple ways with the most common being the error rate (i.e. the ratio between the number of misclassified samples to the total number of samples). Model evaluation provides metrics for measuring the performance of learners. In addition, several authors have introduced statistical tests for performance comparison such as “Friedman Test”, “Nemenyi Test” and “Wilcoxon signed Rank Test” [48].

When applying a learning algorithm there is no assurance that the chosen parameters τ yields the best performance [44]. In addition, there is no learning algorithm that produces the most accurate classifier on a given problem [44]. The rational approach is to try many learners and select the one with the best performance on a different sample set.

Ensemble learning adopts an alternative strategy to address Model Selection by amalgamating multiple learners [49]. The combination can also reduce overfitting, while providing sufficient expressive power to learn complex hypothesis [49]. We use in our experiments a popular technique called stacking (more details are given in Section 2.11).

1.6 Evaluation and comparison

1.6.1 Cross Validation

Cross validation is a resampling technique used to estimate the test error of models with limited data points [44]. It is commonly used in machine learning to compare and select a model based on the model evaluation scores. Several resampling techniques have been used such as k-fold cross validation and leave one out [50].

k-fold cross validation refers to the task of taking the available data and partitioning it randomly into k subsets or folds, of approximately equal size. Then, the k-1 of the folds are used to train a set of models that are then evaluated on the remaining fold. This Procedure is repeated for all the K times. Figure 1.9 illustrates a 4-cross validation technique.

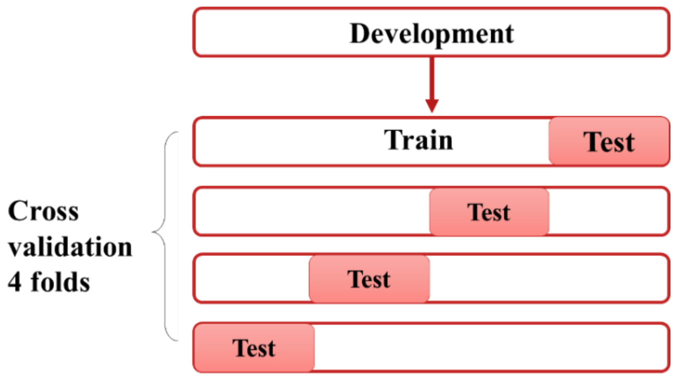


Figure 1.9: 4-fold cross validation [37].

The performance scores from the K-fold cross validation are then averaged. The choice of *k* is generally preferred to be 5 or 10, but there is no formal rule and it could take any value. The satisfaction of the data is recommended; this means that each fold has to have a good representation of the entire dataset. Thus, ensuring that the data partitioning is balanced, with all the classes present in all the folds, with approximately the same amount of data for each class [28].

1.6.2 Evaluation Metrics

There exists numerous perfomance metrics in the machine learning literature [51]. The choice of the right metrics in one of the curcial steps in defining the solution to the problem. Furthermore, when dealing with a single-tag Audio Tagging task which is considered to be a multi-class classification problem. The evaluation involves measuring the performance of the proposed methods based on the accumulated values of the intermediate statistics, denoted by TP, TN, FP, and FN the sums of the true positives, true negatives, false positives, and false negatives accumulated throughout the test data, resulting in overall metrics calculated accordingly as instance-based or class-based. In instance based (i.e. micro-averaging), intermediate statistics are accumulated over the entire data. Overall performance is calculated based on these, resulting in metrics with values that are most strongly affected by the performance on the most common classes in the considered problem. Whereas, in class-based (i.e. macro-averaging), intermediate statistics are accumulated separately for each category (scene or event class). Overall performance is then calculated as the average of class-wise performance, resulting in values that emphasize the system behavior on the smaller classes in the considered problem [28].

A Confusion matrix

Confusion matrix (Figure 1.10) is a performance measurement for machine learning classification. It is a matrix of $n \times n$ where n represents the number of classes. The row dimension contains the actual values, whereas, the column dimension consists of the predicted label. This Figure provides a representation of the confusion matrix with $n = 2$.

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

The metrics that can be computed from the Confusion matrix includes precision, recall, F1-score, the mean average precision and the classification accuracy.

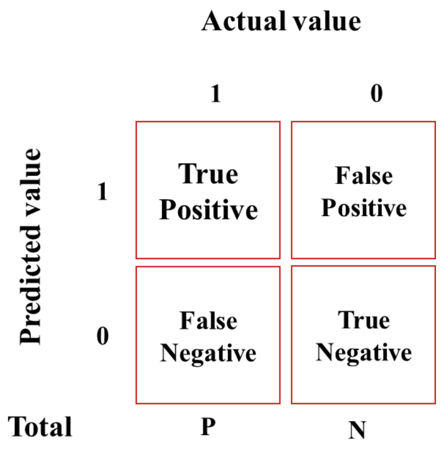


Figure 1.10: Confusion matrix [52].

B Accuracy

Accuracy is the ratio between the number of correct predictions made by the model and the total number of examples. Accuracy can be computed using the previous metrics (TP, TN, FP, FN) as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.7}$$

C Precision and Recall

Precision is the Fraction of predicted positives which are actually positive. Recall is the fraction of actual positives which are correctly predicted:

$$Precision = \frac{TP}{TP + FP} \tag{1.8}$$

$$Recall = \frac{TP}{TP + FN} \tag{1.9}$$

D F1-score

F1-score is the average between precision and recall. It measures how many examples the model classifies correctly. The greater the F1-score is, the better the performance of the model. It is given by the equation below.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{1.10}$$

E The mean average precision (MAP)

Map is based on larger set of measurements. It is typically more stable (less noisy) than point measures such as F1-score. MAP is the mean of all the average precision across all the class labels.

$$MAP@x = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{\min\{K,x\}} P(i), \tag{1.11}$$

where N is the number of data samples, K is the number of classes, whereas we are calculating the mean average precision at the top x of labels. $P(i)$ is the precision at a cutoff i.

F Averaging single-label scores

When multiple class labels are to be retrieved, averaging the evaluation measures can give a view on the general results. It can be obtained through two averaging operations depicted in the equations below.

Let k be the number of possible class labels and $B(tp, \mathrm{tn}, \mathrm{fp}, \mathrm{fn})$ represent some specific binary evaluation measure $B \in \{Accuracy,\ Precision,\ Recall,\ \mathrm{F1-score}\}$ that is calculated based on the number of the true positives (tp_i), true negatives (tn_i), false positives (fp_i), and false negatives (fn_i) after a binary evaluation for a label i.

Macro-averaging measure

$$B_{macro} = \frac{1}{k}\sum_{i=1}^{k} B(tp_i, fp_i, tn_i, fn_i) \tag{1.12}$$

Micro-averaging measure

$$B_{micro} = B(\sum_{i=1}^{k} tp_i\,, \sum_{i=1}^{k} fp_i\,, \sum_{i=1}^{k} tn_i\,, \sum_{i=1}^{k} fn_i) \tag{1.13}$$

1.7 Feature Selection

Feature Selection is the process of determining what inputs should be presented to a classification algorithm. It aims to reduce the number of features by eliminating the redundant and irrelevant features from the feature set [53]. In addition, shrinking the feature set improves the generalization ability of the system and reduces the potential presence of overfitting [54]. There exist three main feature selection paradigms: **filters**, which select features based upon a statistical measure of correlation; **wrappers**, which select features based upon the performance of classification algorithms; and **embedded methods**, a wide group of algorithms which select features as part of the classification process [53]. In the scope of this thesis, we have focused on the embedded methods, we selected Neural Network models that contain **built-in** feature selection. Meaning that the model will only include features that help maximize the generalization ability of the system. In these cases, the model can pick and choose which representation of the data is best by integrating feature selection in network engineering. In Convolutional Neural Networks, features are dynamically selected by tuning the weights associated with the kernels (filters) [55]. In the Gated Convolutional Neural Network, the gating mechanism allows the model to select which features are relevant for predicting the class label [56].

1.8 Statistical Tests

Recently, the machine learning community has become increasingly aware of the need for statistical validation of the published results [57]. Various researchers adopt different statistical and common-sense techniques to decide whether the differences between the algorithms are real or random. In this section we shall examine several statistical tests used in our thesis.

1.8.1 Friedman Test

The Friedman test is a non-parametric statistical test used to test for differences between several algorithms over the class labels (i.e. Sound events or tag). It first ranks the techniques for each class label separately according to the chosen evaluation metric (i.e. accuracy, F1-score). The best performing technique gets the rank 1, the second best gets rank 2, etc. In case of ties, average ranks are assigned. Let r_i^j be the rank attributed the j^{th} system on the i^{th} class label; and let $R_j = \frac{1}{N}\sum_{i=1}^{N} r_i^j$ denote *the average rank* of system $j \epsilon \{1, \dots, t\}$ over N class labels. Under the null hypothesis, it is assumed that all algorithms are equivalent and so their ranks their average rank should be equal. The Friedman statistic is distributed according X_F^2 with $t - 1$ degree of freedom for sufficiently large N and t (usually $N > 10$ *and* $t > 5$). It is given by:

$$X_F^2 = \frac{12N}{t(t+1)}\left[\sum_j R_j^2 - \frac{t(t+1)^2}{4}\right] \tag{1.14}$$

In their study Iman and Davenport reported that X_F^2 is conservative and derived a new statistic F_F which is distributed to the F-distribution with $(t-1)$ and $(t-1)(N-1)$ degrees of freedom.

$$F_F = \frac{(N-1)X_F^2}{N(t-1) - X_F^2}. \tag{1.15}$$

If the null-hypothesis is rejected, we can proceed with a post-hoc test such as *the Nemenyi test* or *the Bonferroni-Dunn test* in order to precisely identify the differences between the algorithms.

1.8.2 Friedman Aligned test

The Friedman Aligned Test is a modified version of the Friedman test [57]. The Friedman test offers intra-set comparability only, however, in some cases comparability among class labels is required. The Friedman Aligned test employs the method of aligned ranks, where a value of location is computed as the average performance achieved by all algorithms in each class label. Then, it calculates the performance obtained by an algorithm and the value of location. This step is repeated for algorithms and class labels. The resulting differences are called aligned observations, which are then ranked from 1 to kn relative to each other. The ranks assigned to the aligned observations are called aligned ranks. The Friedman Aligned

Ranks test statistic T is compared for significance with a chi-square distribution for $t-1$ degrees of freedom.

$$T = \frac{(t-1)\left[\sum_{j=1}^{t} \widehat{R_j^2} - (tn^2/4)(tn+1)^2\right]}{\{[tn(tn+1)(2tn+1)]/6\} - \left(1/_t\right)\sum_{i=0}^{n} \widehat{R_i^2}}, \tag{1.16}$$

where $\widehat{R_\iota}$ is equal to the rank total of the i^{th} class label and $\widehat{R_J}$ is the rank total of the j^{th}algorithm. If the null-hypothesis is rejected, we can proceed with a post-hoc test such as *the Nemenyi test* or *the Bonferroni-Dunn test.*

1.8.3 Quade test

The Quade test offers an improvement for some specific cases where the data samples are more difficult or the differences registered between various algorithms over the data samples is larger [57]. The Quade test conducts a weighted ranking analysis over the class labels. The procedures start by finding the ranks r_i^j in the same way as the Friedman test does. The next step requires the original values of performance of the classifiers x_{ij}. Ranks are assigned to the class labels according to the size of the sample range in each class label. The Quade statistic is then calculated, which is distributed according to the F-distribution with $t-1$ and $(t-1)(n-1)$ degrees of freedom. If the null-hypothesis is rejected, we can proceed with a post-hoc test. A detailed description of the mathematical process of the Quade test can be found in [57].

1.8.4 Nemenyi test

Nemenyi test is a post-hoc test invoked when the Friedman test rejects its null hypothesis and it is used when all methods are compared to each other. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference (CD).

$$CD = q_\alpha \sqrt{\frac{t(t+1)}{6N}}, \tag{1.17}$$

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

1.8.5 The Bonferroni-Dunn test

The Bonferroni-Dunn test is a post-hoc test, invoked after the Friedman test. It is used when we are interested in comparing one technique against the other alternatives. It adjusts the significance level $\propto$ in a single step by dividing the value of $\propto$ by the number of comparisons performed $t-1$. The alternative way to compute the same test is to compute the CD (i.e.

Critical Differences) using the same equation as for the Nemenyi test, however using the critical values for $\propto/(t-1)$.

1.8.6 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 [57]. The formulation of this test is the following. We designate by d_i the difference between the performance scores of two techniques on N datasets. $i \in \{1, \dots, N\}$. We first rank these differences according to their absolute values; in case of ties average ranks are attributed. Then, we compute the sum of ranks for the positive and the negative differences, which are denoted as R^+ and R^-, respectively. Their formal definitions are given by:

$$R^+ = \sum_{d_i<0} \operatorname{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \operatorname{rank}(d_i)$$
$$R^- = \sum_{d_i<0} \operatorname{rank}\,(d_i) + \frac{1}{2}\sum_{d_i=0} \operatorname{rank}\,(d_i). \tag{1.18}$$

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 value for T_w can be found in any textbook on general statistics [48], whereas for larger N, the statistics:

$$z = \frac{T_w - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{1.19}$$

1.9 Related work

Table 1.2 presents some literature works related to approaches that employ machine learning and deep learning methods along with various input representations for Audio Tagging. The list expedites a general overview of the different classifiers and features pertaining to their characteristics. It also consists of the latest matters surrounding the development of Audio Tagging systems. These studies analyze the behavior of different learning methods to extract high-level representations of input features. For instance, In [58], the audio clip is treated as an image. Unlike the object in the image, in audio clips from real life, some events, such as “Bark” may last a few minutes, while other events, such as “gunshot,” may only last for hundreds of milliseconds. This characteristic of sound events increases the difficulty of Audio Tagging based on CNN. To better use CNN to extract high-level representations, Convolutional Recurrent Neural networks have been suggested to amalgamate convolutional and recurrent layers such as LSTMs [59] and GRUs [60] in a single deep learning architecture in order to emphasize the benefits of both. In addition, attention mechanisms [56] [61] [62] has been proposed in CNN such as the GLU. The benefits of such mechanisms have been shown across a range of tasks, from Audio Tagging [63], to language modeling [56], which shows that the attention method can alleviate the overfitting problem.

Furthermore, Ensemble learning which consists of building a classification model by integrating multiple classifiers. The combination of different models can improve the accuracy and robustness for the classification [49] using the complementary prediction result from different models. However, the ensemble learning such as Stacking [64] has been under-explored for Audio Tagging. Most of the previous methods simply compute the average of the predictions [65].

Table 1.2: Various audio analysis classification approaches.

Reference	Type of classifier	Feature representation	Performance
[66]	Hidden Markov Models (HMM)	MFCC	Accuracy=30.1%
[67]	Gaussian Mixture Models (GMM)	MFCC	F-score=13.08%
[68]	Deep Neural Networks (DNN)	Short time Fourier transform, Log-Mel energy, MFCC	EER[1]=0.1785%
[69]	Convolutional Recurrent Neural Network (CRNN)	Log-Mel Spectrogram	F-score=69.1%
[63]	Convolutional Gated Recurrent Neural Network (CGRNN)	Mel-Filter Banks (MFB)	EER[1]=0.11%

1.10 Conclusion

In this Chapter, we have provided an outline of the basic concepts of sound representation that are essential to understand the ideas treated in this work. We have reviewed some important concepts of classification in general. Many classification paradigms have been applied for developing audio tagging systems. Most importantly, deep learning-based systems have attracted a wide spread attention from the research community due to their effectiveness. Therefore, in the next chapter, we will give an overview of some deep learning notions, including several well-known deep architectures.

EER [1]: Event Error Rate.

CHAPTER 2: DEEP LEARNING FOR AUDIO TAGGING

2.1 Introduction

In the previous chapter we have discussed the notions of sound acquisition and representation required to prepare the audio signals for machine learning and deep learning approaches. In addition to some concepts of machine learning. Deep learning is a modern machine learning method, known for its ability to express highly non-linear relationships between the input and the output [70]. Deep learning techniques are now the state of the art in many audio applications [21] due to their capacity to learn the mapping between the target labels and a lower level representation such as the magnitude spectrogram or even the raw audio signals. Moreover, Deep Neural Network is among the recently proposed deep learning techniques in context of sound analysis. These models require their own kind of engineering effort in order to find the appropriate architecture for the target task; for instance, tuning the hyper-parameters, choosing the right training algorithm and regularization techniques. Deep learning models benefits from larger datasets, in order to expose the models to a larger and varied training samples. Data augmentation includes a set of techniques that enhance the size and quality of the dataset such as mixup.

2.2 Artificial Neural Networks

Artificial neural networks (ANNs), also known simply as neural networks (NNs), are considered to be a machine learning method that is based on the inner workings of the human brain [21]. The objective of NNs is no different from other models, (i.e. to approximate a function). NNs are composed of stacks of inter-connected artificial **neuron** blocks (also called **layers**) that aim to find a mapping between the input and the target output. Each network has a set of hyper-parameters that determine the network architecture (number of neurons in each layer, number of layers etc.) and a network training procedure (optimization method parameters, regularization parameters etc.). The most common components of NNs are presented in the following subsections.

2.2.1 Neuron

A neuron is the basic unit of a NN. Each neuron receives inputs through its incoming weighted connections from other neurons and possibly itself. For each connection the input is received as the transmitted signal multiplied by the connection weight w. The sum of the weighted received signals and a bias term b is computed, then passed through an activation function F and finally output through an outgoing weighted connection (Figure 2.1).

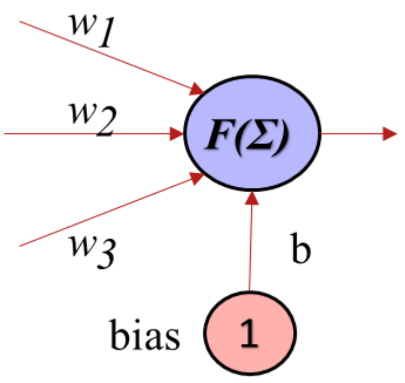


Figure 2.1: A simple model of a neuron [71]**.**

2.2.2 Layer

Neurons in the network are grouped into layers. There is one input layer, a variable number of hidden layers, and one output layer. NNs that include more than one hidden layer are often grouped under the name deep learning. Each layer receives inputs from the proceeding layer (possibly itself) and delivers outputs to the following. The input layer is composed of D nodes, where D is the dimensionality of the input data. Each node reads one of the components of an input vector $x_i \in X$ and outputs it to the following layers neurons. Hidden layers are between the input and output layer that perform intermediate computations of the network. The output layer in classification tasks typically consists of a neuron for each class. For a given output neuron k, its computed value is usually interpreted after normalization in the range [0, 1]. The incoming connections for the neurons in one layer form a matrix W. Together with **a bias vectors *b***, the **weight matrices** for all layers W represent the parameters θ of the model (Figure 2.1).

2.2.3 Activation function

The activation function scales the activation of a neuron into an output signal. The most commonly used in NNs are:

- **Sigmoid function** scales the input value between 0 and 1, as we know that probability lies between 0 and 1. This is why it is used for predicting the probability of the output. It can be defined as follows.

$$\sigma(x) = \frac{1}{1 + e^{x}} \tag{2.2}$$

- **Rectified linear unit (ReLu)** is a thresholding function that returns the same input x as long as it is greater than zero otherwise it returns zero.

$$ReLU = \max(0, x) \tag{2.3}$$

- **SoftMax** makes it possible for the network to output among all the possible classes, the class with the highest probability. It is also used to normalize the outputs of multiclass classification tasks.

  Given a vector X of inputs to the output layer where j indexes the output units $j = 1,2,3, \dots, m$, it is defined for each of its components x_j as:

$$softmax(x_j) = \frac{e^{x_j}}{\sum_m e^{x_m}}, \tag{2.4}$$

  where x is a vector of inputs to the output layer, j indexes the output units.

- **Tanh function (tanh)** is a hyperbolic tangent function outputs the value in the range $[-1,1]$. This function shows derivatives that can reach higher values than the sigmoid derivatives and is expressed as follows.

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.5}$$

2.3 Training Algorithms

2.3.1 Back propagation algorithm

The main idea behind Back propagation (BP) is to adjust the networks weights so that the output values for the training data are as close as possible to the desired target output. BP is a method used to compute the gradients that will be used for the update of each neuron's weights. Furthermore, BP can be split into two fundamental steps.

- **Forward pass:** an input is applied to the first layer and propagated through the network so that all the weights are computed.

- **Backward pass:** based on the desired target output, derivative of the cost function is back-propagated from the last layer to the first.

The mathematical description of the backpropagation algorithm can be found in [72].

2.3.2 Gradient Descent (GD)

GD is the most common optimization algorithm in *machine learning* and *deep learning* [73]. It is used to update the parameters of a function such that it minimizes the difference between the function output and the desired output. It is achieved by updating the parameters in the opposite direction of the gradient of the hypothesis function $J(\theta)$. The size of the step it takes for each iteration to reach the local minimum is determined by the **learning rate** α, and $O\theta J(\theta)$ as the gradient of the hypothesis function. GD has three variants that differ based on the amount of data utilized to compute the gradient of the hypothesis function.

Stochastic Gradient Descent (SGD) uses a single training dataset at a time (one row after another) and then iteration adjustment of weights for each row.

Batch Gradient Descent (BGD) uses the entire dataset rows for training at the same time and then makes adjustments to the weights.

Mini-Batch Gradient Descent is a hybrid of BGD and SGD, uses more than one training example at a time.

2.4 Network regularization

There are several techniques proposed to address issues typically encountered in Machine Learning such as overfitting [44] and vanishing (or exploding) gradients [74]. These techniques are grouped under regularization techniques. Below we explained two of the techniques specifically designed for Neural Networks.

2.4.1 Dropout

The term "dropout" refers to dropping units (hidden and visible) in a neural network. Dropping a unit out means temporarily removing it from the network, along with all its incoming and outgoing connections. The Dropout algorithm temporarily removes randomly selected neurons by setting the selected neuron weight to zero with a certain probability. Therefore, these neurons do not have any effect on the output of the network. Dropout has been shown to address two problems, it prevents overfitting and provides a way of efficiently approximately combining exponentially many different Neural Network architecture [21].

2.4.2 Batch Normalization (BN)

Due to the vanishing (or exploding) gradients problem (Explained in section 2.7), the distribution of the activations for each layer becomes very diverse for deeper Neural Networks, which slows down the learning as each layer is updated with the same **learning rate**. Batch normalization is used as an intermediate layer that will take care of the normalization of the hidden units activations at each layer to zero mean and unit standard deviation [75]**.** It has been introduced to reduce the co-variance shift in the network, and to accelerate the training of neural networks rather than to properly regularize the model by counteracting overfitting. However, in [76] the authors argue that BN provides similar regularization benefits as dropout.

2.5 Hyper-parameters of the Network

The neural network training is dependent on numerous hyper-parameters which are capable of determining the capacity and the complexity of the model. The hyper-parameters employed in the scope of this study are as follows:

Learning rate (α) is the scalar that determines the amount of change in the gradient towards the proper direction. Higher values of α leads to overshooting the optimal solution in the hypothesis function, while lower values of α leads to too many iterations to towards the best value. There is no consensus on the ideal value of the learning rate [21]. Thus, it should be selected by examining the performance of the model by varying it.

Number of hidden layers determines the depth of the network. The higher the value is, the deeper the network.

Number of units in each layer can be different for each layer. These values determine the number of weights in total.

Batch size is the number of samples processed before the weights of the model are updated.

Number of epochs is a hyper-parameter that defines the number of times that the dataset is passed forward and backward through the neural network. The BP algorithm decreases the training error with the increasing number of epochs.

2.6 Convolutional Neural Network

CNNs are hierarchical NNs that have been designed initially for image classification. A typical CNN is characterized by the repetition of convolution layers, an activation function, followed by the pooling layers which are partially connected. Due to these layers CNN can achieve a complete overview of the input with a good invariance to patterns shifts. The network usually ends with a fully connected layer with a SoftMax output. However, several studies [77] have proposed an alternative to the fully connected layer since the latter is prone to overfitting, namely **The Global Average Pooling**. The actual difference, when compared to other types of neural networks, lies in the introduction of a combination of convolution and pooling operations. Moreover, CNNs use a modified version of the back-propagation algorithm to ensure the shared weight constraint. The sample architecture of the convolutional neural networks is shown in the figure below.

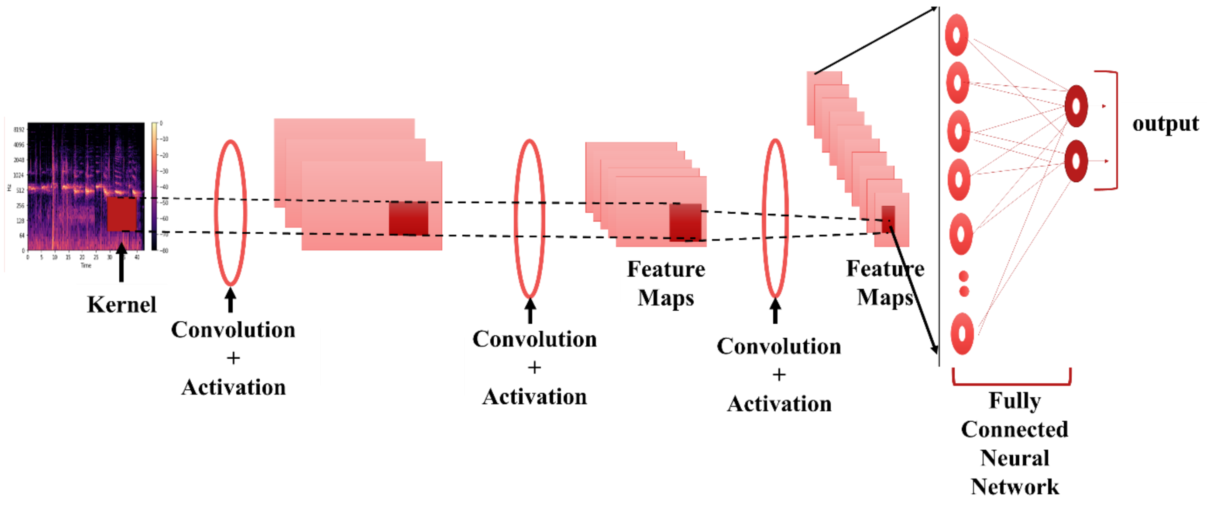


Figure 2.2: A simple architecture of CNN [78].

2.6.1 The convolution layer

The convolution layer introduces a special way of receiving the input. Instead of being connected to all the inputs coming from the previous layer, it takes a small portion of the input space (i.e. The receptive field), the weights of this portion create a convolutional kernel (filter). It consists of sliding the filter over the input space by a certain stride value and apply the dot product between the filter and a portion of the input matrix, resulting in a **convolved feature matrix** or **the feature map**. This mathematical operation is called **convolution**, which will vastly reduce the number of the parameter and helps the model learn the relevant features only. The operation is shown in Figure 2.3.

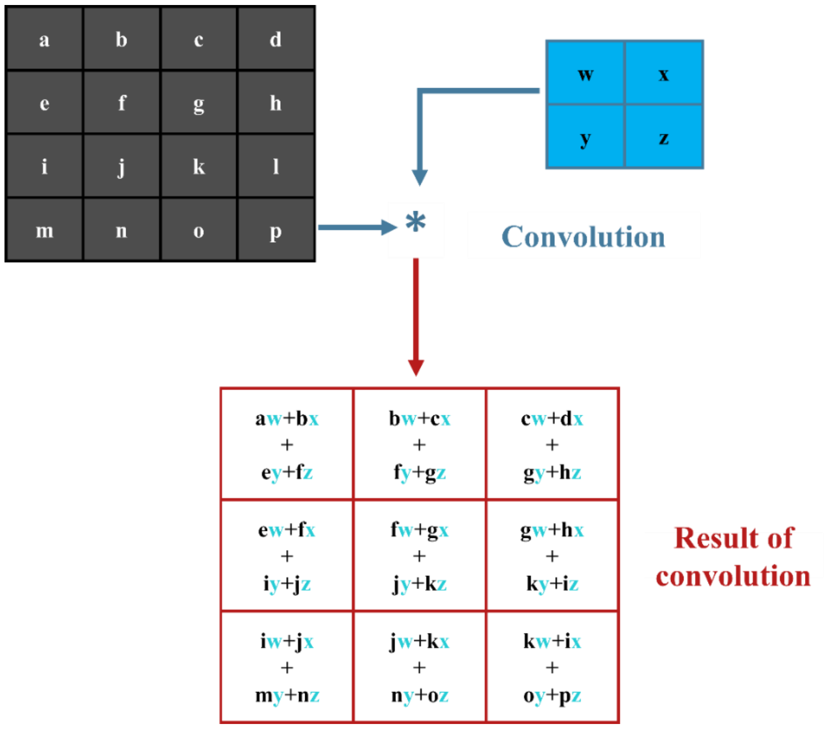


Figure 2.3: Convolution in CNN [79].

2.6.2 The Activation layer

The activation layer applies an activation function over each feature map returned from the convolution layer, to create a nonlinear relationship between the inputs and the outputs. The most commonly used one is the **Rectified Linear Units activation function** (ReLu) [80] (Section 2.2.3).

2.6.3 The Pooling layer

The main idea behind the pooling layer is down-sampling in order to reduce the computational load by progressively reducing the spatial size of the representation. Max-pooling is one of the most common types of pooling. It takes small rectangular blocks from the feature map and subsamples it to produce a single maximum output from the block then slides to the next block with a specific stride value. The most commonly used size of max pooling is 2x2 [81]. An illustration of this operation is depicted in Figure 2.4.

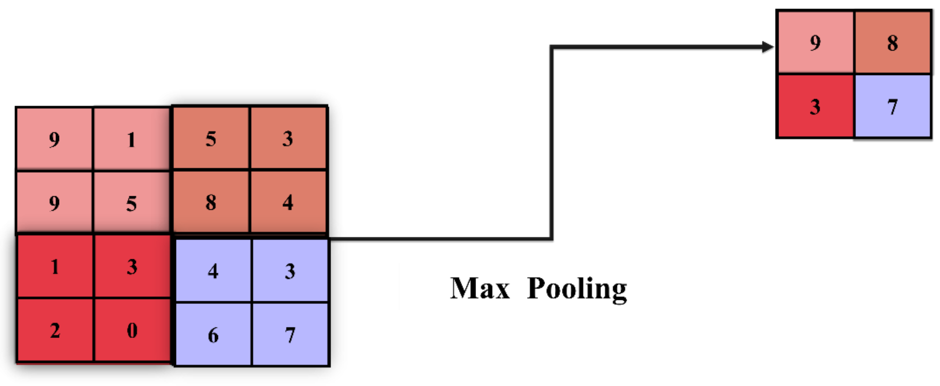


Figure 2.4: **Max Pooling [20].**

2.6.4 Fully Connected layer

The neurons in a fully connected layer are arranged in a way that each node is directly connected to every node in both the previous and in the next layer as shown in Figure 2.2 The hidden layers of the network which is composed of a stacked layer of convolution and pooling output the feature maps, which are fed to the Dense layers. The Fully connected dense layers flattens the feature matrix into a one-dimensional vector, the vector is then fed to the final dense layer with SoftMax activation function (Section 2.2.3) to output a vector of probabilities ∈ [0,1].

2.6.5 The Global Average Pooling layer (GAP)

The idea is to generate one feature map for each corresponding category of the classification task. GAP has no parameter to optimize thus the overfitting problem is avoided. The global average pooling mechanism computes the mean value for each feature map and supplies it to the final dense layer with a SoftMax activation function. The SoftMax function takes each value and converts it to a probability (with the probability of all values summing to 1.0). However, in [82] the authors have used a fully connected layer along with the GAP layer. The GAP computes the mean value for each feature map and supplies the result to the input of each unit in a single fully connected layer.

2.7 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of neural networks that are beneficial to use with sequential data [83]. The structure of RNN is similar to that of the standard neural network, with a distinction that RNNs allow their neurons to share their outputs with pervious layer neurons, creating a feedback cycle. This indicates that an RNN may sustain the temporal activations even in the absence of input [84]. Therefore, RNNs are dynamical systems with a dynamical memory over time that can compute sequences of different lengths. However, the complexity of RNN structure makes it hard to train properly due to the vanishing gradient and exploding gradient problems. RNNs uses a straightforward extension of the backpropagation algorithm, denoted Back Propagation Through Time (BPPTT) [85]. There are various variations of RNNs, we note among them Long Short-Term Memories (LSTMs) and Gated Recurrent Units (GRUs). An example of the RNN architecture is illustrated in Figure 2.5.

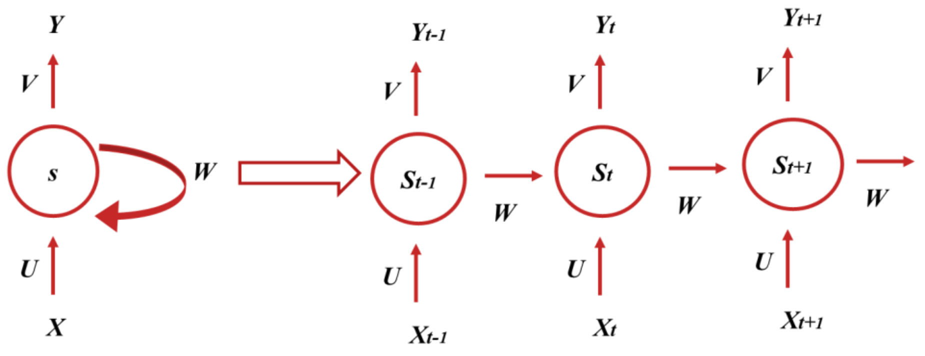


Figure 2.5: Left: Visual illustration of the RNN recurrence relation, Right: The RNN states are recurrently unfolded over the sequence t-1, t, t+1 [86].

2.7.1 Recurrent layer

The recurrence relationship defines how the state evolves step by step over the sequence via a feedback loop over previous states. The recurrent layer applies the same function f over a sequence recurrently. The recurrence relation given by:

$$S_t = f(S_{t-1}, X_t), \tag{2.6}$$

where f is a differentiable function, S_t is a vector of values called the internal network state at a step *t* of both the current input as well as the previous state, X_t input at step *t*, and S_{t-1} is the network's summary of all of the previous inputs.

The recurrent layer through the backpropagation algorithm updates a set of three parameters (weights), namely **U, W** and **V**. The vector **U** transforms the input X_t into the state S_t, **W** transforms the previous state S_{t-1} into the current state S_t and **V** maps the newly computed internal state S_{t-1} to the output Y_t. They apply a linear transformation over their respective input. The internal state and the output of the network can be defined as follows:

$$S_t = f(WS_{t-1}, UX_t), \tag{2.7}$$

$$Y_t = VS_t, \tag{2.8}$$

where f is the non-linear activation function such as tanh, sigmoid, or ReLU and X_t denotes the input vectors *(* $x_1,\dots,x_t$*)*. The state S_t represents a sequence of state vectors *(*$s_1,\dots,s_t$*)*. Finally, the output Y_t is a sequence of probability vectors *(*$y_1,\dots,y_t$*)* of the next input in the sequence. Through this recurrence relation, each state is dependent on all of the previous computations, which allows RNN to have memory over time and to compute sequences of different lengths.

2.7.2 The exploding and vanishing problem

In theory, RNNs can remember information for arbitrarily a long period of time. However, in practice, they are limited to looking back only a few steps [86]. This issue is known as *vanishing and exploding gradients problem*. These problems arise during the training of a deep network when the gradients are being propagated back in time all the way to the initial layer. The gradients coming from the deeper layers have to go through continuous matrix multiplications because of the chain rule, and as they approach the earlier layers, if they have small values (less than 1), they shrink exponentially until they vanish and make it impossible for the model to learn , this refers to the *vanishing gradient problem*. On the other hand, if they have large values (more than 1) they get larger and eventually blow up and crash the model, this refers to the *exploding gradient problem*. To cope with these shortcomings, Gated Recurrent Layer Methods such as Gated Recurrent Units (GRU) and Long-Short Term Memory Networks (LSTM) [59], have been introduced.

2.7.3 Long Short-Term Memory (LSTM)

Hochreiter and Schmidhuber have studied the problems of vanishing and exploding gradients extensively and have proposed a solution called **Long Short-Term Memory** network [59]. LSTMs can handle long-term dependencies due to a specially crafted memory. It contains special units called **memory blocks** in the recurrent hidden layer. The memory blocks contain memory cells with self-connection which stores the temporal state of the network; in addition to a special multiplicative unit called **Gates** to control the flow of information. Each memory block contains an input gate, output gate and a forget gate. The **input gate** controls the flow of input activations into the memory cell. The **output gate** controls the output flow of cell activations into the rest of the network. Finally, **the forget gate** scales the internal state of the cell before adding it as input to the cell through the self-recurrent connection of the cell; Therefore, adaptively forgetting or resetting the cell memory. The forget gate addresses a weakness of LSTM models preventing them from processing continuous input streams that are not segmented into subsequences [87].

2.7.4 Gated Recurrent Unit (GRU)

GRU is a type of recurrent method that was introduced as an improvement over LSTM [60]. The idea behind GRU and LSTM is very similar, which is to allow the relevant information from the previous timesteps to be stored in the cell state, and to control the cell state through the gates that learns which information is relevant for the given task. The main difference is that GRUs combine the forget and the external input gates of LSTM in a single gate called the **update gate**; hence, has less parameters compared to LSTM. In addition to the update gate, GRU has an additional cell called the **reset gate**. Both gates are composed of weights and an activation function. Each cell includes a cell state, which consists of the accumulated information from the previous timesteps. During training, the gate weights learnt by which proportions to combine the cell state and the input for the current timestep to produce the gated unit output for the current timestep. The **reset gate** adjusts the incorporation of new input with the previous memory and the update gate controls how much to preserve of the previous memory. Furthermore, an enhanced version has been introduced, namely **bi-directional GRU** [88] allows to process the sequence input in two directions including forward and backward ways, which can increase the model capacity and flexibility [60].

2.8 Convolutional Recurrent Neural Network

Another increasingly common hybrid architecture is to follow one or more convolutional layers by recurrent layers. This approach is alternately known as **Convolutional Recurrent Neural Networks (CRNN)**. Combining convolutional and recurrent layers in a single deep learning architecture integrates the strength of both CNNs and RNNs, which has shown excellent performance in sound analysis applications [69] [17], while overcoming their individual weaknesses.

Convolutional neural networks are able to extract higher level features that are invariant to local spectral and temporal shifts. Furthermore, convolutional layers can be used to learn filters (i.e. weight kernels) that are shared among the input and shifted in both time and frequency. However, the temporal context that can be modeled using convolutional layers is limited [1]. Recurrent layers, with a gated structure such as GRUs and LSTMs can be used to extract long term temporal information among the consecutive time frames by utilizing information from the earlier time frames as a feedback for the calculation of the higher-level representation for the current frame.

In order to find the optimal hyper-parameters for the deep neural networks, a grid search is performed including the combinations of some of the hyper-parameters such as the number of layers, number of units in each layer etc. Figure 2.6 depicts the effect of the number of network parameters on the performance of RNN, CNN and CRNN when tested on TUT-SED synthetic 2016 dataset [69]. For the same number of parameters, CRNN shows a better performance than CNN and RNN methods in most cases. This observation confirms that combining the CNNs and RNNs into CRNN classifier is a more efficient and powerful way of utilizing the network parameters compared to CNN and RNN. The effect of sequence length, (i.e. the number of frames per input example) has been investigated by Cakır et al. [69]. Their results indicate that CRNNs can model the whole event in a single sequence, which results in improving the performance. In addition, the longest temporal context that RNN can model is not sufficient to model the events as a whole in a single sequence. This can be explained with the role of the convolutional layer in CRNN architecture. Convolutional layers learn filters that are invariant to short term temporal variations and they effectively pre-process the features to be used in longer temporal context in the following recurrent layers.

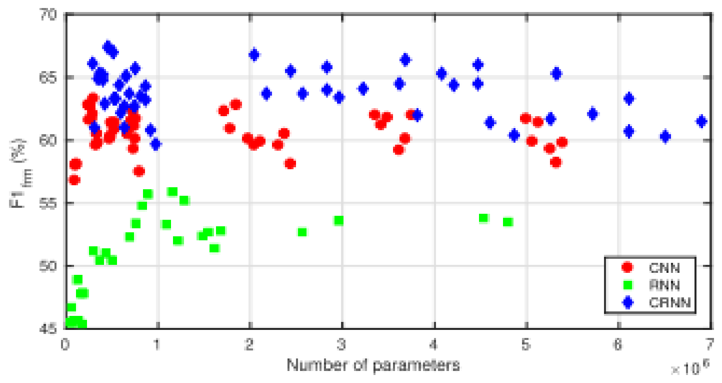


Figure 2.6: Number of parameters vs F1-score for CNN, RNN and CRNN [69].

2.9 Gated Convolutional Neural Network and Gated Convolutional Recurrent Neural Network

Gated Convolutional Neural Network (GCNN) and Gated Convolutional Recurrent Neural Network (GCRNN) are variants of CNN and CRNN, respectively. The difference is that each convolutional layer is replaced with a Gated Convolutional layer. Gating mechanism has been shown to be essential for Recurrent Neural Networks to reach state-of-the-art performance [21]. This mechanism has shown to produce better results for several task such as audio classification [89], language modeling [56].

2.9.1 Gated Linear Units

The Gated Linear Unit (GLU) [22] is used as an activation function to replace the Rectified linear activation function (ReLu) in CRNN and CNN. The structure of GLU is shown in Figure 2.7 GLU can reduce the gradient vanishing problem for deep networks [22] by providing a linear path for the gradients propagation while keeping nonlinear capacities through the sigmoid operation. Similar to the gating mechanisms in Long-Short Memories (LSTM) or Gated Recurrent Units, GLU can control the amount of information of a time-frequency representation unit flow to the next layer. GLU are defined as:

$$Y = (W * X + b) \odot \delta(V * X + c), \quad (2.9)$$

where δ denotes the sigmoid function, the symbol $\odot$ is the element-wise product and $*$ is the convolution operator. W and V are convolutional filters, b and c are biases. X denotes the input tensor in the first layer or the feature maps in the interval layers in the model. The value of sigmoid function ranges from 0 or 1, so if a GLU gate value is close to 1, then the time-frequency unit is attended. Whereas, if a GLU gate value is near 0, then the corresponding time-frequency unit is ignored. Thus, the network learns sound events and ignore the unrelated sounds.

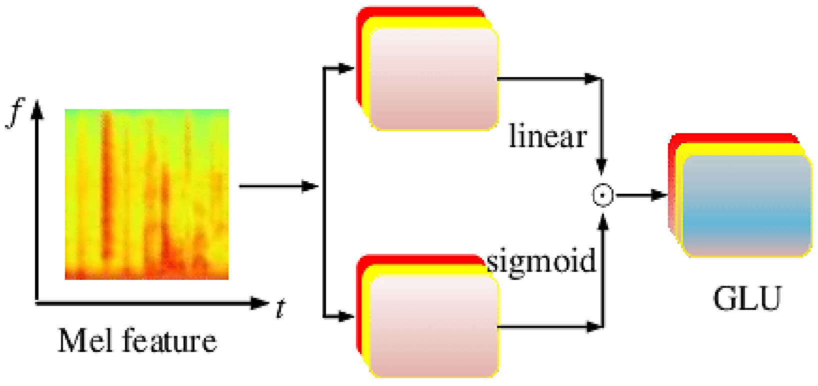


Figure 2.7: GRU structure [15].

2.10 Mixup

Data Augmentation encompasses a suite of techniques that enhance the size and quality of the training datasets [32]. It is widely used along with Deep Learning models in order to address the overfitting problem [30]. Numerous data augmentation techniques have been introduced in the literature such as: Mixup, Time Stretching, and Pitch Shifting [90]. In our study, we have chosen Mixup approach due to its simplicity and its significant improvements of audio classification systems [33].

Mixup is a method that randomly mixes a pair of inputs and their associated target values [91]. Consider a pair of inputs, $x_1$and x_2, and its corresponding binary label, y_1 and y_2. To mix these, a parameter, $\alpha \in [0, 1]$ is used as a mix ratio.

$$x = \alpha x_1 + (1 - \alpha)x_2 \tag{2.10}$$

$$y = \alpha y_1 + (1 - \alpha)y_2 \tag{2.11}$$

The outputs x, y are then used as the training examples.

2.11 Ensemble Methods

2.11.6 Stacked generalization

Stacked generalization (stacking) is a general method which uses a high-level model to combine lower-level models to achieve greater predictive accuracy [92] (see Figure 2.8),it first creates T level-1 classifiers, $C_1, \ldots, C_T$, based on a cross-validation partition of the training data. To do so, the entire training dataset is divided into B blocks, and each model-1 classifier is first trained on a different set of B-1 blocks of the training data. Each classifier is then evaluated on the B^{th} block (i.e. not seen during training). The outputs of these classifiers on their pseudo-training blocks constitute the training features for the level-2 (meta) classifier, which effectively serves as the combination rule for the level-1 classifiers. Note that the meta-classifier is not trained on the original feature space, but rather on the predictions of level-1 classifiers.

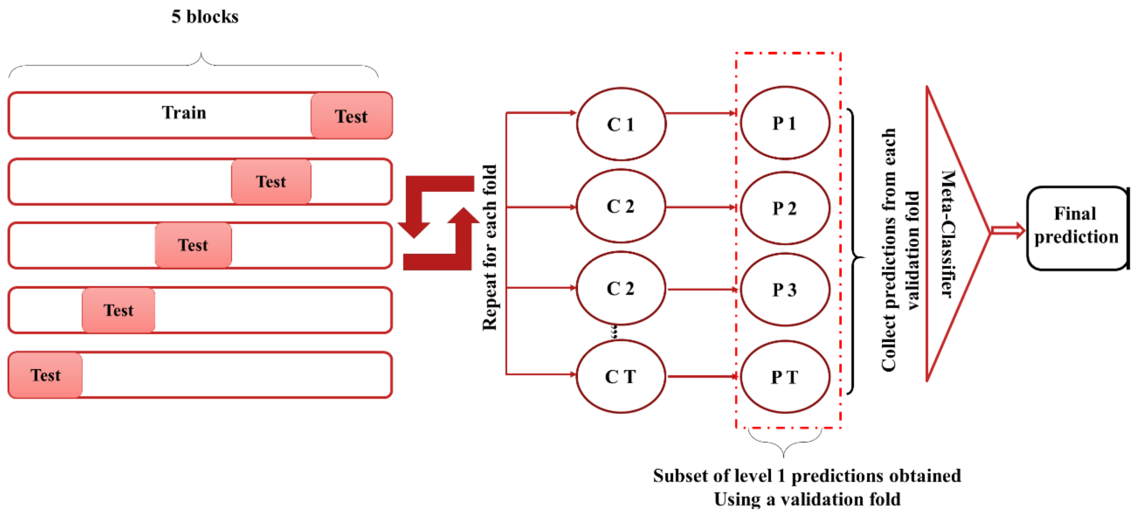


Figure 2.8: The stacking mechanism.

2.12 Summary of empirical and theoretical findings on CNN, CRNN, GCNN and GCRNN

Empirical studies have shown that different deep architectural designs such as CNN, CRNN, GCNN and GCRNN often have an advantage over shallow architectures when dealing with complex learning problems [1] [55] [21].

In recent studies, the focus of research shifted from parameter optimization and connections readjustment towards the improvement of the architectural design of the deep CNN networks [55]. This shift resulted in many new architectural designs such as CRNN. Cakir et al. have evaluated CRNN on three datasets of real-life sound recordings (i.e. TUT Sound Events Synthetic 2016, TUT-SED 2009 and TUT-SED 2016) and compared its performance to CNN and RNN. Their results show an improvement in performance of CRNN method over CNN and RNN [8]. The performance of CRNN indicates an architectural advantage compared to the rest. It gathers the capabilities of both CNN and RNN in one classifier. However, their proposed CRNN strongly depends on the amount of available annotated data. Specifically, when the performance of CRNN for TUT-SED 2016 (78 minutes) is compared to the performance on TUT-SED 2009 (1133 minutes) and TUT-SED Synthetic 2016 (566 minutes), there is a clear performance drop both in the absolute performance and in the relative improvement with respect to other methods. Dependency on large amounts of data is a common limitation of current deep learning methods. Similar findings have been reported in [93].

Recently, attention-based neural networks have been applied to a wide variety of tasks, such as speech recognition [94] [95], visual object classification [96]. The term attention means to focus on specific parts of the input. Xu et al. have proposed an attention based neural network for audio tagging that can automatically select the important frames for the targets, while ignoring the unrelated parts (e.g. the background noise segments) [63]. They have compared the proposed method with two state-of-the-art systems that used CNN as a classifier, Lidy-CNN [97] and Cakir-CNN [14]. The results indicate that the attention-based method reduces the Event Error Rate from 0.13 to 0.11 on average. In addition, the gated network performs better in detecting the long-term patterns of the "child speech" which occur frequently in the whole dataset. Similar results have been reported by Xu et al. [3]. In their study, they have applied the learnable Gated Linear Unit (GLU) to replace the ReLU activation after each layer of the Convolutional Recurrent Neural Network for audio tagging and weakly supervised sound event detection. The audio tagging results show that the gated CRNN gains effective improvement with a F1-score of 54.2 compared the DCASE2017 baseline [98] with F1-score of 18.2.

2.13 Challenges

2.13.1 Intra-class variability

Sound event classes for Sound analysis tasks are often defined broadly such as phone ringing, doorbell etc. This presents a challenge for sound analysis methods in the form of intra-class variability. For instance, doorbell class can be used to represent all types of doorbells, whose acoustic characteristics can vary significantly among the examples of this class. Therefore, in order to claim that a sound analysis system can robustly detect doorbells, it should be able to do so on a wide variety of doorbells. This requires the sound analysis method to be able to detect or extract the acoustic features that are found in common among different examples of the same class [22].

2.13.2 Noisy labels

Although the dataset is labeled only a portion of it is verified and the rest is not guaranteed to have the true labels. This problem can be formulated as a form of label noise. These mislabeled instances are considered to be outliers which are generally the result of four potential sources. Firstly, the information which is provided to the expert may be insufficient to perform reliable labelling. Secondly, since collecting reliable labels is time consuming and costly task, there is increasing interest in using less reliable labels provided by non-expert such as using automated classification methods. Thirdly, when the labeling task is subjective, the problem of inter-expert variability might occur. Inter-expert variability is defined broadly as the presence of important variability in the labeling by several experts. Eventually, label noise can also simply come from data encoding or communication. Furthermore, these noisy labels may lead to lower classification problem and slower optimization, thus they should be taken into account in learning problems [99] [66].

2.14 Conclusion

Throughout this chapter, we have reviewed some important concepts of deep learning methods. First, we have presented the deep neural networks architectures used in our work. We have also presented Mixup, a data augmentation technique and Stacking ensemble learning approach, highlighting the importance of these techniques in order to obtain a reliable robust Audio Tagging system. Furthermore, we have summarized some empirical and theoretical findings on the differences of the architectures presented in this chapter. Finally, we have discussed the challenges related to Audio Tagging research.

PART II: EXPERIMENTS

In this part we describe the methodology that we have followed for evaluating and comparing different deep learning approaches. It is composed of two chapters. In the first chapter we present the experimental setup defined to evaluate the performance of our Audio Tagging system, whereas in the second chapter, we discuss the results of our experiments.

CHAPTER 3: DESIGN AND ANALYSIS OF AUDIO TAGGING: EXPERIMENTAL SETUP

3.1 Introduction

This chapter presents the experimental setup used to conduct our experiments. First, in Section 3.2 we present our dataset. Next, in Section 3.3, we present the tools that we have used to conduct our experiments. Then, in Section 3.4 we present the setup for the proper steps for design of experiments in context of sound analysis.

3.2 Dataset

We have conducted series of experiments on **Freesound Dataset Kaggle 2018** (FSDKaggle2018). consisting of audio samples from Freesound annotated using a vocabulary of **41 labels** from Google's AudioSet Ontology [33]:

- Tearing
- Bus
- Shatter
- Gunshot, gunfire
- Fireworks
- Writing
- Computer keyboard
- Scissors
- Microwave oven
- Keys jangling
- Drawer open or close
- Squeak
- Knock
- Telephone
- Saxophone
- Oboe
- Flute
- Clarinet
- Acoustic guitar
- Tambourine
- Glockenspiel
- Gong
- Snare drum
- Bass drum
- Hi-hat
- Electric piano
- Harmonica
- Trumpet
- Violin, fiddle
- Double bass
- Cello
- Chime
- Cough
- Laughter
- Applause
- Finger snapping
- Fart
- Burping, eructation
- Cowbell
- Bark
- Meow

FSDKaggle2018 is a reduced subset of **FSD** [17], which is a large-scale, general-purpose open audio dataset that is currently under development. It is composed of audio content collected from Freesound [100] (i.e. Freesound is a sound sharing site developed and maintained by the Music Technology Group in Barcelona). It contains a total of 11 073 files provided as uncompressed waveforms with 16-bit bit depth and 44.1 kHz sample rate.

The ground truth data provided in this dataset has been obtained after a data labeling process which resulted into two types of annotations:

Manually-verified annotations represent the part of the dataset that was manually verified by human annotators that manually assessed the presence/absence of an automatically assigned sound category. In most cases, there is no additional acoustic material other than the labeled category. In few cases, there may be some additional sound events, but these additional events will be out-of-domain (i.e. they do not belong to any of the 41 AudioSet categories of FSDKaggle2018).

Non-manually verified annotations are mainly composed of un-rated candidate annotations, and complemented with a small amount of rated annotations. These annotations are most probably not accurate. Some of the audio clips annotated as non-verified could present several sound sources (even though only one label is provided as ground truth). These additional sources are typically out-of-domain, but in few cases, they could be within the domain. Figure 3.1 shows the distribution of manually-verified and non-verified annotations per category in the training set.

FSDKaggle2018 dataset was split into two sets a train set and a test set: The train set is meant to be for system development and includes 9473 audio clips unequally distributed among 41 categories. The minimum number of audio clips per category in the train set is 94, and the maximum is 300. The total duration of the train set is almost 18h. Out of the 9473 clips from the train set, 3710 have manually-verified annotations and 5763 have non-verified annotations. Figure 3.1 shows the distribution of manually-verified and non-verified annotations per category in the train set. The test set is composed of 1600 clips with manually-verified annotations and with a similar category distribution to that of the manually-verified portion of the train set. The minimum number of manually-verified audio clips per category in the test set is 25, and the maximum is 110. These annotations are complemented with 7800 *non annotated clips* which are also included in the test set but that will not be used for evaluating our systems.

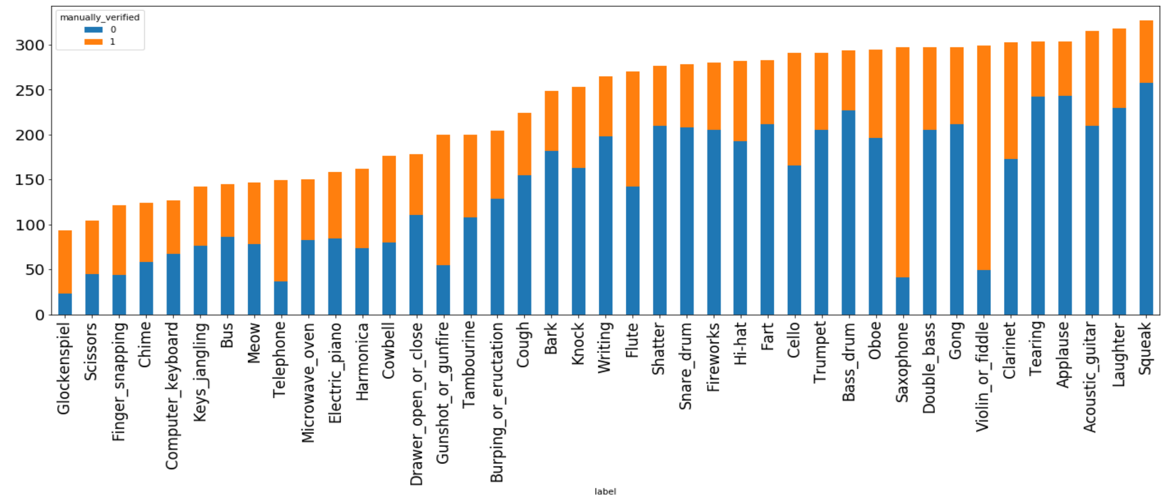
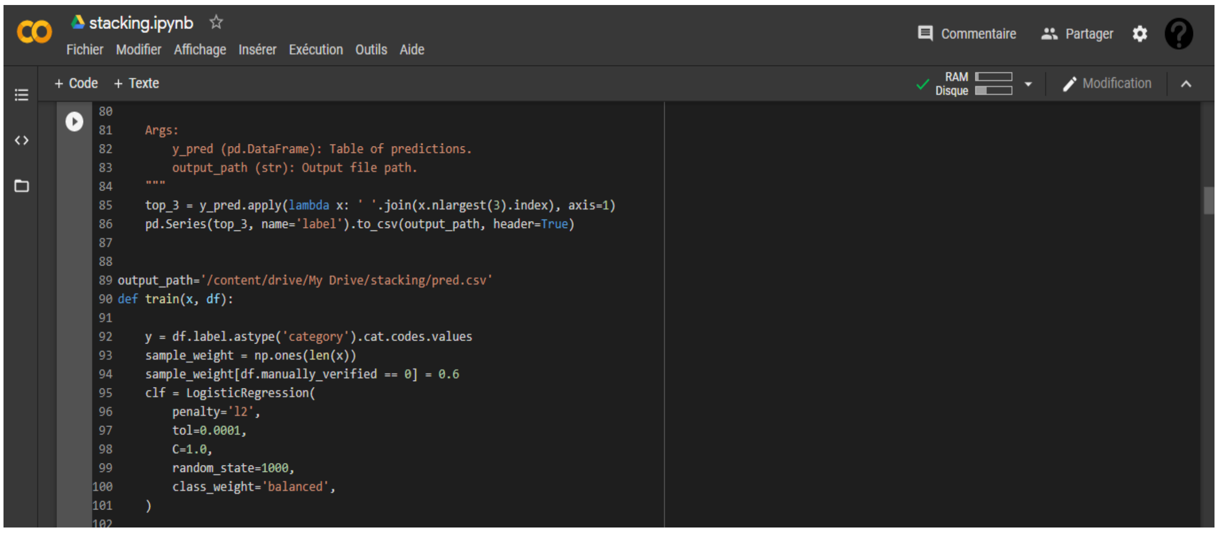


Figure 3.1: **Distribution of manually-verified and non-verified annotations per category in the train set.**

3.3 Tools

We have carried our experiments using **Python** which is an object-oriented open source programming language [101]. First, we have performed feature engineering using Spyder which is a scientific environment based on Python [102]. We have displayed our features using Librosa 0.7.2 which is a Python package for signal processing [103]. Moreover, the deep learning process was performed using a set of Python packages such as Tensorflow and Keras which are high-level APIs for building and training deep learning models. Other libraries that were invoked include Numpy, Seaborn and Pandas, etc [78] [79].

We have trained our models using **Google Colaboratory** which is a free Cloud service. it consists of executable Python notebooks stored within Google Drive and connected to a Cloud based runtime to perform the execution of the Python code on Nvidia Tesla K80 GPU. Figure 3.2 shows a screenshot of Colab notebook.



```

    Args:
        y_pred (pd.DataFrame): Table of predictions.
        output_path (str): Output file path.
    """
    top_3 = y_pred.apply(lambda x: ' '.join(x.nlargest(3).index), axis=1)
    pd.Series(top_3, name='label').to_csv(output_path, header=True)


output_path='/content/drive/My Drive/stacking/pred.csv'
def train(x, df):

    y = df.label.astype('category').cat.codes.values
    sample_weight = np.ones(len(x))
    sample_weight[df.manually_verified == 0] = 0.6
    clf = LogisticRegression(
        penalty='l2',
        tol=0.0001,
        C=1.0,
        random_state=1000,
        class_weight='balanced',
    )
```

Figure 3.2: **Screenshot of Colab notebook.**

3.4 Design and analysis of Audio Tagging systems

We aim at building different **Single-tag Audio Tagging systems** that are able to recognize an increased number of sound events of very diverse nature (including musical instruments, human sounds, domestic sounds, animals, etc.). Furthermore, we use data with annotations of varying reliability (i.e. Manually-verified annotations and Non-manually verified annotations). Specifically, we illustrate the principles of designing machine learning experiments for building Audio Tagging systems. To this end, we have carried out two sets of experiments.

First, we have examined Log-Mel Spectrogram feature extraction technique using two different sets of parameters along with four deep learning architectures (VGG13, GCNN, CRNN and GCRNN), while varying their parameters. Giving eight models in total, these models involve the use of preprocessing techniques and data augmentation in order to improve their performance and reduce overfitting. We have utilized the F1-score along with Accuracy and MAP@3 metrics to perform the evaluation of our systems.

Second, we have investigated the effect of combining, the predictions of the previously trained models. In order to consolidate their strengths, to achieve this, we used a popular technique called stacking. Figure 3.3 and Figure 3.9 shows the General schema for the first and second case study respectively.

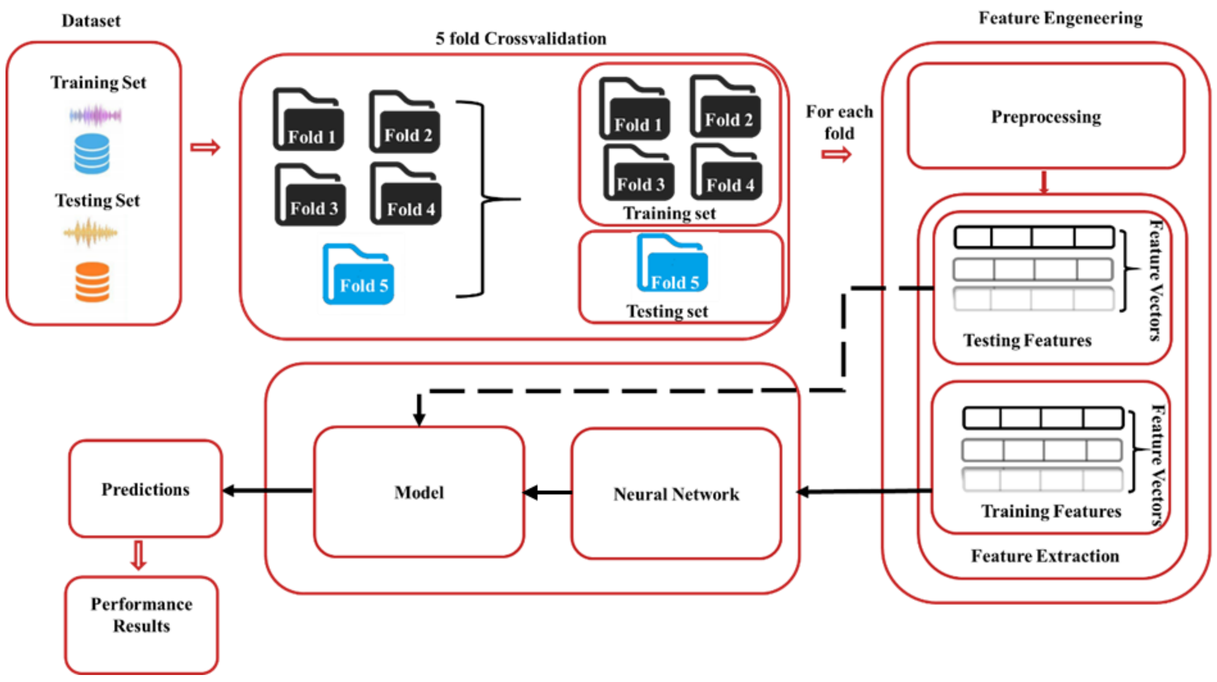


Figure 3.3: General schemas that highlight the primary steps for conducting the first case study.

3.4.1 Cross validation

We have performed **a stratified 5-cross validation** to split the training set into non-overlapping training and validation sets. We have used the training set for learning the models, whereas the validation set was employed for model selection. We have **shuffled** our data to generate different combinations in order to ensure that across each fold, there was a similar number of manually verified examples and the events are approximately equally represented. Cross validation leads to a considerable computation time increment, but it is a very common procedure used during model selection stage. We test the best model, (i.e. the result of model selection), on the provided test set. We have performed **5-fold cross validation** using Scikit-learn library.

3.4.2 Feature Engineering

A Preprocessing

After visualizing our raw data (waveform) we have noticed that some of them contained long sequences of silence. In order to remove them, we have started by segmenting our input audio files into small frames and by calculating their Root Mean Square (RMS) energy. The segments that were lower than the fixed threshold was judged to be unimportant and, hence, removed. However, the rest of the segments were kept to be used as inputs to our neural networks. Figure 3.4 depicts silence removal for one sound taken from our dataset. The extracted non-silent sections are encased in a black rectangle.

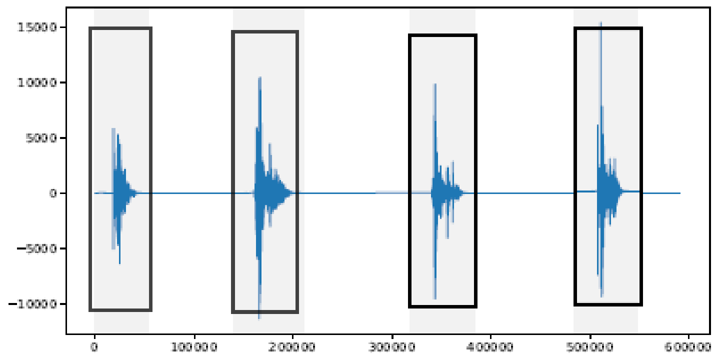


Figure 3.4: The silence removal process applied on the file "071e836c.wav".

B Feature extraction

We have implemented two configurations for the frequency domain features **Log-Mel Spectrograms**. Figure 3.5 presents the feature extraction process that we have followed in our work.

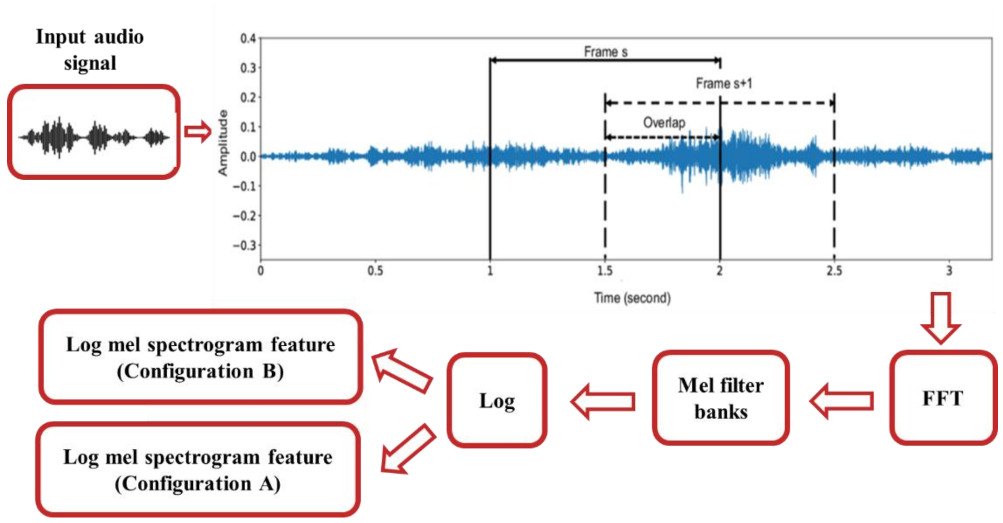

Input audio signal
Amplitude
Frame s
Frame s+1
Overlap
Time (second)
FFT
Mel filter banks
Log
Log mel spectrogram feature (Configuration B)
Log mel spectrogram feature (Configuration A)

Figure 3.5: The process of feature extraction.

Table 3.1 summarizes the parameters used for extracting the Log-Mel Spectrogram features.

Table 3.1: Log-Mel Spectrogram setup.

Parameters	Configuration A	Configuration B
Sample rate	32 000 Hz	32 000 Hz
Window size	1024	512
Hop size	512	256
Mel bands	64	64
Window function	Hamming	Hamming

Figures 3.6 and 3.7 depict the Log Mel-Spectrogram features for configuration A and B respectively of one preprocessed sound taken from our dataset.

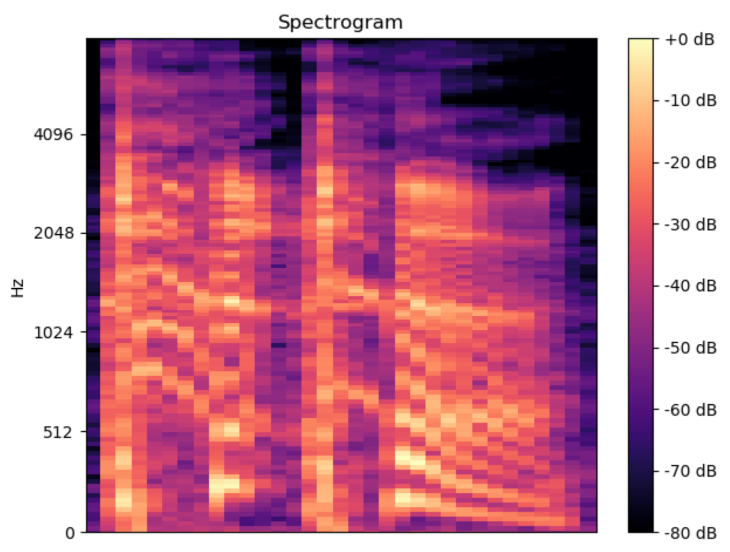


Figure 3.6: The Log-Mel Spectrogram (Configuration A) of “fff81f55_0.wav”.

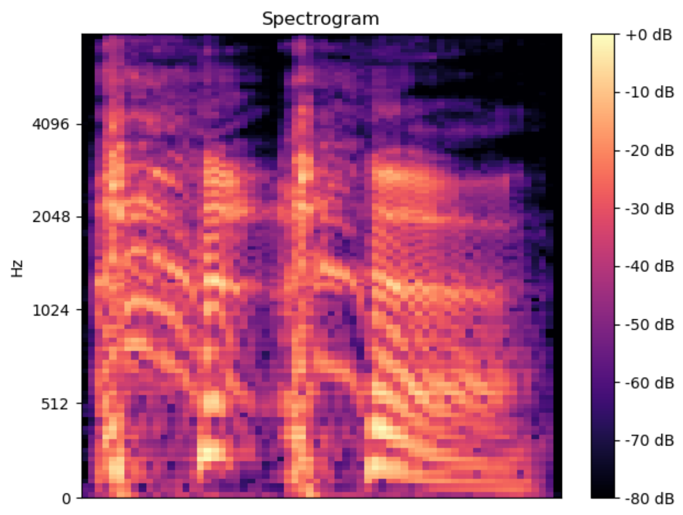


Figure 3.7: The Log-Mel Spectrogram (Configuration B) of "fff81f55_0.wav".

To address the problem of variable length sequence (or variable length inputs), which requires for our data to be transformed such that each sequence has the same length. We have decided to split each feature vector into chunks of a fixed size (128 × 64 where the first axis is the temporal dimension). This corresponds to 2 seconds chunks and 1 second chunks for configurations A and B, respectively. When the length of the feature vector was greater than the chunk size, an additional chunk was added to include the remainder of the audio file; whereas, when it was lower, the feature vector was padded.

3.4.3 Data Augmentation

Due to the limited size of our dataset and to improve the classification accuracy further, we explore the use of **mixup** data augmentation method with an alpha value set to 1.0.

3.4.4 Classification approaches

A two-stage classification method which consists of a training and a testing stage for Audio Tagging is illustrated in Figure 3.8. It is performed based on audio segments with a single class annotated throughout. The 41 annotations are encoded into target outputs which are used in the training stage with audio signals. In this case the classes are mutually exclusive, this condition is included into the neural network architecture by using an output layer with softmax activation function, which normalizes the output frame-level class probabilities to sum up to one. These probabilities are used to get the overall classification output by summing up class-wise, the frame-level class presence probabilities. Finally, the label with the highest combined probability is assigned. In our case, the predictions for chunks from the original audio files were merged using **geometric mean** to produce clip level predictions. This process *was repeated for* *the best four epochs selected according to (**Map@3**)* metric resulting into four prediction probabilities for each audio clip. Then, these probabilities were merged using **arithmetic mean** to produce the final prediction for each audio clip.

Arithmetic and Geometric mean

Calculating the average of a variable or a list of numbers is a common operation in machine learning. The average (mean) is a single number that represents the most common value for a list of numbers. More technically, it is the value that has the highest probability from the probability distribution that describes all possible values that a variable may have.

Arithmetic mean

The arithmetic mean ($\bar{x}$) is calculated as the sum of the data values divided by the total number of values, referred to as N.

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \,. \tag{3.1}$$

The arithmetic mean can be calculated using the mean NumPy function.

Geometric mean

The geometric mean of a series of positive numbers $x_1, x_2 \ldots, \ldots, \ldots, x_n$ is defined as the n^{th} root of its product:

$$\left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}} = \sqrt[n]{x_1 x_2 \cdots x_n}\,. \tag{3.2}$$

The geometric mean can be calculated using the gmean SciPy function.

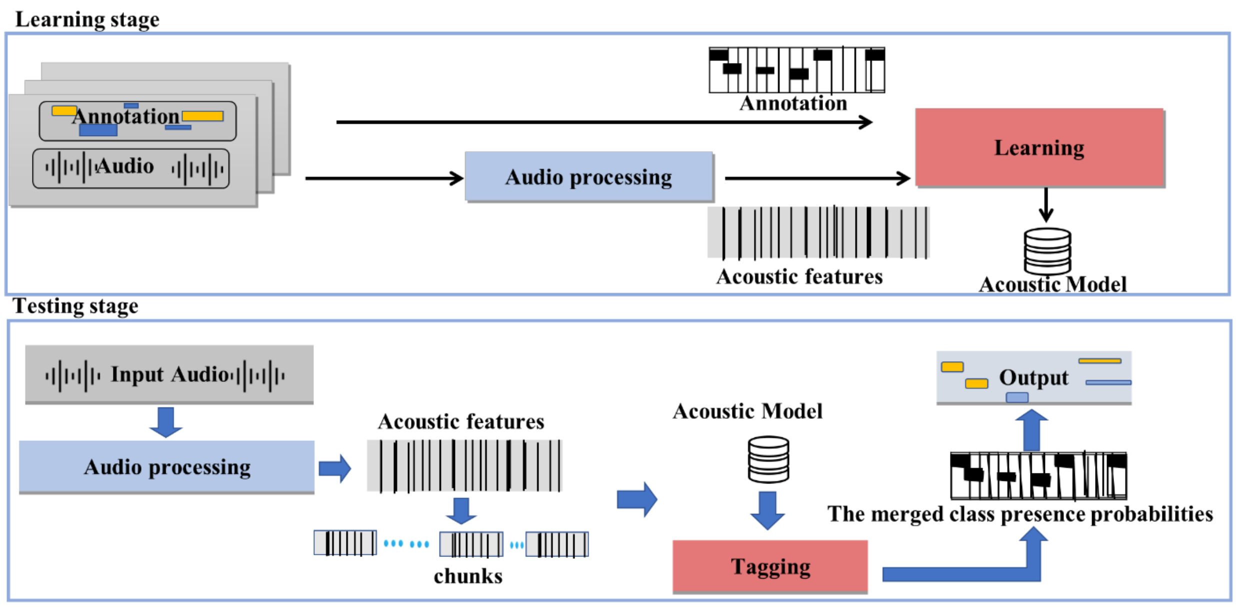


Figure 3.8: The training and testing phase of an Audio Tagging system.

A. Neural network architectures

We have built four neural network architectures VGG13, CRNN, GCNN and GCRNN. The first one is a standard CNN that was inspired by the VGG13 network proposed in [106]. Each convolutional block consists of two convolutional layers followed by a max pooling layer that halves each spatial dimension. After each convolution, which uses the rectifier (ReLU) activation function, batch normalization is applied as a form of regularization. After the convolutional blocks, each feature map is averaged to a scalar value. Finally, a softmax layer is used to generate the predictions.

The CRNN architecture is an extension of VGG13. Instead of averaging across both spatial dimensions after the convolutions, only the frequency dimension is averaged initially. A bidirectional recurrent layer is then applied to output a feature vector for each time step. Finally, these feature vectors are averaged. By using a recurrent layer, the temporal dynamics of the input can be learned.

The two remaining architectures are GCNN and GCRNN. GCNN is a variant of VGG13; whereas, GCRNN is a variant of CRNN. *The difference is that each convolutional layer is replaced with a gated convolutional layer*. Note that these architectures were inspired from papers [79, 80, 81].

Table 3.2 describes the neural network architectures used in the first case study; convolutional blocks parameters are encapsulated by square brackets. The first two parameters in each line are the kernel size and the number of filters. ‘BN’ refers to batch normalization. ‘GLU’ and ‘Bi-GRU’ refer to **Gated Linear Units** and **Bidirectional Gated Recurrent Units**, respectively.

Table 3.2: Description of the neural network architectures.

Feature size	VGG13	CRNN	GCNN	GCRNN
128 × 64		**Log-Mel Spectrogram**		
64 × 32	$\begin{bmatrix}3\times3,64, & \text{BN, ReLU}\\3\times3,64, & \text{BN, ReLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,64, & \text{BN, ReLU}\\3\times3,64, & \text{BN, ReLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,64, & \text{BN, GLU}\\3\times3,64, & \text{BN, GLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,64, & \text{BN, GLU}\\3\times3,64, & \text{BN, GLU}\end{bmatrix}$
		2x2 Max Pooling		
32 × 16	$\begin{bmatrix}3\times3,128, & \text{BN, ReLU}\\3\times3,128, & \text{BN, ReLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,128, & \text{BN, ReLU}\\3\times3,128, & \text{BN, ReLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,128, & \text{BN, GLU}\\3\times3,128, & \text{BN, GLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,128, & \text{BN, GLU}\\3\times3,128, & \text{BN, GLU}\end{bmatrix}$
		2x2 Max Pooling		
16 × 8	$\begin{bmatrix}3\times3,256, & \text{BN, ReLU}\\3\times3,256, & \text{BN, ReLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,256, & \text{BN, ReLU}\\3\times3,256, & \text{BN, ReLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,256, & \text{BN, GLU}\\3\times3,256, & \text{BN, GLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,256, & \text{BN, GLU}\\3\times3,256, & \text{BN, GLU}\end{bmatrix}$
		2x2 Max Pooling		
8 × 4	$\begin{bmatrix}3\times3,512, & \text{BN, ReLU}\\3\times3,512, & \text{BN, ReLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,512, & \text{BN, ReLU}\\3\times3,512, & \text{BN, ReLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,512, & \text{BN, GLU}\\3\times3,512, & \text{BN, GLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,512, & \text{BN, GLU}\\3\times3,512, & \text{BN, GLU}\end{bmatrix}$
		2x2 Max Pooling		
4 × 2	$\begin{bmatrix}3\times3,512, & \text{BN, ReLU}\\3\times3,512, & \text{BN, ReLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,512, & \text{BN, ReLU}\\3\times3,512, & \text{BN, ReLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,512, & \text{BN, GLU}\\3\times3,512, & \text{BN, GLU}\end{bmatrix}$	$\begin{bmatrix}3\times3,512, & \text{BN, GLU}\\3\times3,512, & \text{BN, GLU}\end{bmatrix}$
		Bi-GRU, 512, ReLU		**Bi-GRU, 512, ReLU**
		Global Average Pooling		
		Softmax (41 Classes)		

B. Ensemble learning

In order to build an ensemble model, we have followed Stacking paradigm. Recall that the Stacking model is composed of two levels. Level 1 consists of deep learning models trained using two Log-Mel Spectrogram features. Level 2 is a shallow-architecture classifier using the meta-features obtained from level 1. These meta-features are obtained by running the previous classifiers (Section A) for each out-of-fold training data to predict the probabilities for each sample in the validating set by using the whole training dataset. For each classifier, the probabilities for 41 classes are be used as the meta-features, which are concatenated to generate the new training dataset, and used as the input for level 2.

For the ensemble learning in level 2, we employ the linear regression algorithm. Inspired by [107], we weigh each training sample. The sample weight of a manually verified sample is set to 1.0, while the weight of a non-manually verified sample is set as a constant value 0.65. In this way, manually verified samples are preferred. Figure 3.9 shows the conceptual architecture of the stacking ensemble used for the second case study.

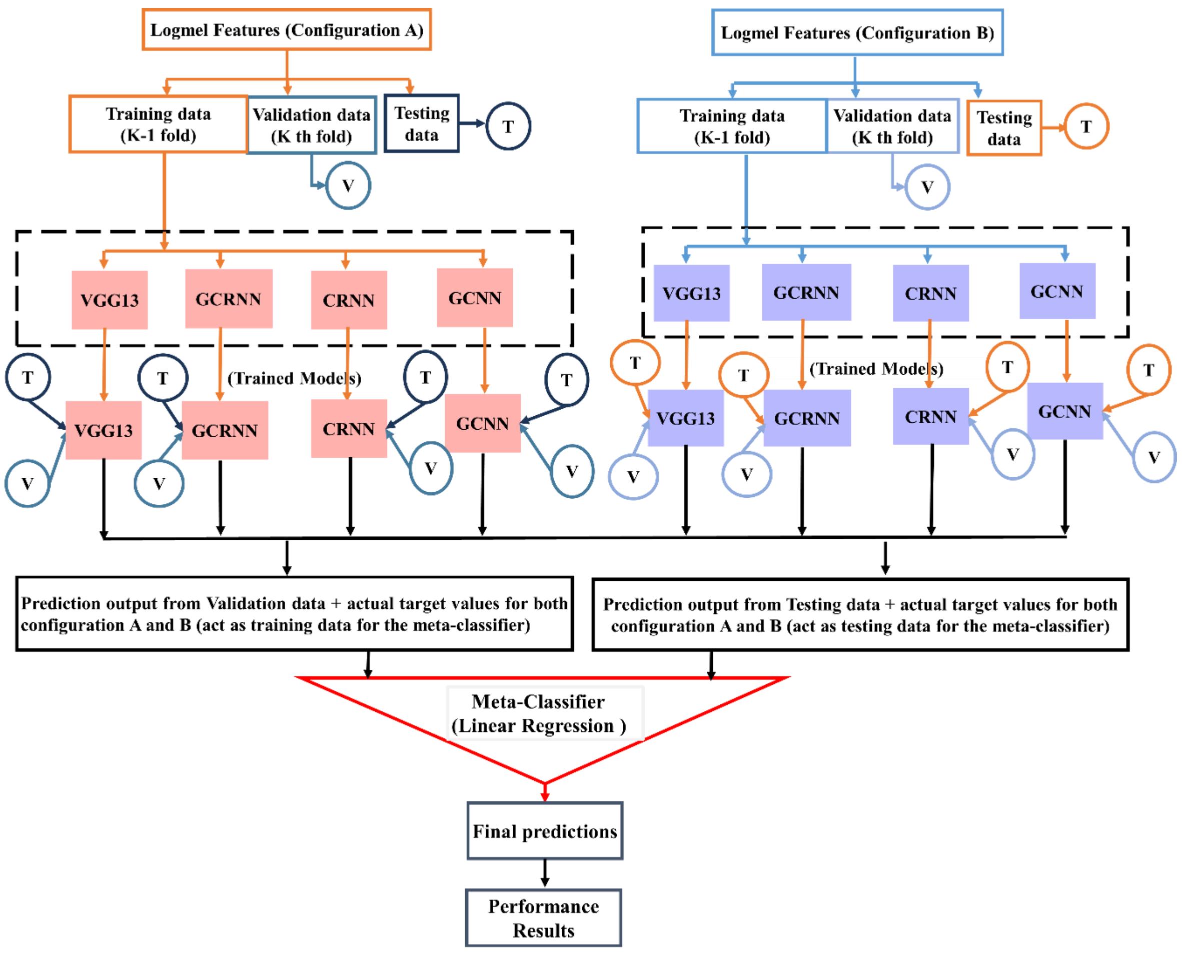

Logmel Features (Configuration A)
Training data (K-1 fold)
Validation data (K th fold)
Testing data
T
V
VGG13
GCRNN
CRNN
GCNN
(Trained Models)
Logmel Features (Configuration B)
Training data (K-1 fold)
Validation data (K th fold)
Testing data
Prediction output from Validation data + actual target values for both configuration A and B (act as training data for the meta-classifier)
Prediction output from Testing data + actual target values for both configuration A and B (act as testing data for the meta-classifier)
Meta-Classifier (Linear Regression)
Final predictions
Performance Results

Figure 3.9: Exhibits a general schema that highlight the primary steps for conducting the second case study.

3.4.5 Evaluation Procedures

The main goal of our work is to compare the performances of several Audio Tagging systems, while varying the learning paradigms and their parameters. We have tested the performance of each system using the following evaluation procedure. After generating the out of fold predictions for both the training and testing set using 5-fold cross validation. Only predictions that correspond to manually verified samples from each fold were used to perform class-wise evaluation using the mean average precision **(MAP@3)**, **F1-score** and averaging the former evaluation metrics using **macro\micro averaging**. We also used **accuracy** to evaluate the overall performance of the two main case studies. Most importantly, we have based our discussions and conclusions on strong statistical tests.

3.5 Conclusion

In this chapter, we have described the setup used to conduct our experimental enquiries, starting from cross validation to classification step. We have presented two general schemes (i.e. 4 individual learners and an ensemble learning approach “Stacking”) that highlight the key steps for carrying out our first and second set of experiments. In the following chapter, we will present the results of these experiments and analyze them in order to derive guidelines for building audio tagging systems based on numerous statistical comparisons.

CHAPTER 4: DESIGN AND ANALYSIS OF AUDIO TAGGING: RESULTS AND DISCUSSION

4.1 Introduction

This chapter discusses the experimental results that we have obtained during our experiments. In Sections 4.2 and 4.3 we analyze and discuss the results of the first and second set of our experiments. In Section 4.2.3 we talk about the training time of our experimental studies.

4.2 First Set of Experiment: The individual models

We have conducted extensive experimental comparison among Audio Tagging methods using various deep learning architectures based on data with annotations of varying reliability. Specifically, we have thoroughly examined Log-Mel Spectrogram features using two different configurations. Most importantly, we have investigated four deep learning architectures VGG13, GCNN, CRNN and GCRNN giving eight Audio Tagging systems in total. Table 4.1 describes these models. For additional information on these architectures and their parameters, please refer to Sections 3.4.4. For evaluation, we have used the F1-score along with MAP@3 metrics. Furthermore, we have based our discussions and conclusions on various statistical tests.

Table 4.1: Summary of the first set of experiment models.

Abbreviation	Classification Model	Feature Set
VGG13$_A$	Convolutional Neural Network	Log-Mel Spectrogram of Configuration A
VGG13$_B$		Log-Mel Spectrogram of Configuration B
CRNN$_A$	Convolutional Recurrent Neural Network	Log-Mel Spectrogram of Configuration A
CRNN$_B$		Log-Mel Spectrogram of Configuration B
GCNN$_A$	Gated Convolutional Neural Network	Log-Mel Spectrogram of Configuration A
GCNN$_B$		Log-Mel Spectrogram of Configuration B
GCRNN$_A$	Gated Convolutional Recurrent Neural Network	Log-Mel Spectrogram of Configuration A
GCRNN$_B$		Log-Mel Spectrogram of Configuration B

Table 4.2 gives the F1-score of the eight individual models. The first column represents the tag, whereas, the rest of the columns designate the systems that are tested in our experiment. The last two rows specify the macro and micro average of each system over all classes, respectively. Table 4.3 shows the average MAP@3 over all events.

Table 4.2: F1-score (%) results of the eight individual models.

Tag	CRNN$_A$	CRNN$_B$	VGG13$_A$	VGG13$_B$	GCRNN$_A$	GCRNN$_B$	GCNN$_A$	GCNN$_B$
Acoustic guitar	85.13	85.06	85.23	85.59	85.74	84.29	**86.83**	85.32
Applause	96.69	95.82	95.56	**97.28**	95.27	96.69	96.99	96.72
Bark	93.83	93.21	93.25	85.43	93.36	90.35	**94.77**	88.53
Bass drum	96.56	**96.93**	96.20	94.17	96.57	94.90	96.60	95.60
Burping	98.14	98.16	96.36	99.06	97.24	97.52	96.93	**99.08**
Bus	**86.05**	78.55	84.26	77.78	84.85	74.32	84.69	81.28
Cello	92.57	91.09	**92.60**	90.72	90.55	91.86	92.21	89.53
Chime	80.36	71.42	79.71	71.86	**81.77**	74.08	79.64	71.39
Clarinet	93.35	93.97	**95.42**	93.11	91.14	92.12	92.89	94.46
keyboard	82.85	87.20	85.86	89.79	83.09	86.78	88.54	**89.58**
Cough	88.75	**93.27**	89.63	90.88	90.20	92.49	89.52	92.15
Cowbell	92.04	93.44	92.90	91.64	**94.44**	92.09	92.68	92.66
Double bass	93.34	92.23	92.45	92.15	92.56	**93.63**	92.92	92.08
Drawer	80.01	80.38	82.19	79.26	85.27	82.04	**83.93**	77.78
Electric piano	92.66	93.64	93.07	92.60	**96.27**	93.22	94.05	93.94
Fart	86.39	85.15	86.09	85.73	86.75	86.58	86.83	**87.77**
Finger snapping	**95.77**	91.49	94.06	92.72	95.54	90.79	93.22	94.33
Fireworks	68.33	61.07	66.22	63.44	**69.21**	61.69	65.46	65.69
Flute	96.56	96.08	95.85	96.52	94.98	96.59	94.59	**98.00**
Glockenspiel	81.75	69.87	**82.94**	70.12	83.45	68.20	79.13	67.13
Gong	85.46	87.04	85.25	88.11	86.29	87.68	87.67	**89.06**
Gunshot	**85.60**	81.94	84.70	81.73	82.99	84.11	84.05	82.89
Harmonica	90.20	**91.96**	91.41	91.68	88.41	89.71	88.67	89.75
Hi-hat	89.92	91.32	88.94	89.72	89.40	91.55	87.85	**91.78**
Keys jangling	**85.55**	79.00	79.50	76.26	82.00	80.63	77.54	77.25
Knock	87.99	88.41	85.63	**90.37**	86.43	89.52	87.44	87.10
Laughter	88.39	**91.71**	87.84	88.36	87.89	91.17	87.99	89.77
Meow	**93.41**	91.99	91.38	89.62	92.75	91.66	90.91	88.98
Microwave	**88.17**	87.82	85.22	86.07	84.32	84.70	84.07	85.71
Oboe	97.89	96.23	**98.11**	96.04	97.38	96.68	96.28	97.19
Saxophone	95.43	96.06	96.18	96.14	94.81	95.59	94.97	**96.33**
Scissors	67.62	65.01	67.59	69.06	65.94	66.07	63.00	**69.08**
Shatter	89.67	87.89	87.34	80.60	**91.85**	83.10	85.42	81.53
Snare drum	**92.50**	90.86	89.84	88.85	90.67	92.31	91.45	88.53
Squeak	40.10	41.44	45.30	38.39	**46.60**	42.07	44.19	40.00
Tambourine	**91.28**	89.96	90.23	90.96	90.21	90.12	90.65	89.19
Tearing	**70.75**	67.73	70.09	70.44	70.40	66.85	69.98	69.84
Telephone	76.42	79.94	79.20	78.90	78.25	80.65	79.38	**81.70**
Trumpet	92.34	92.21	92.90	91.88	91.68	**93.79**	91.75	91.23
Violin	95.90	95.74	96.15	95.71	94.99	95.98	**96.27**	94.89
Writing	82.19	85.00	81.78	83.22	82.59	85.34	78.93	**85.58**
Macro Average	**87.02**	86.03	86.69	85.41	86.93	85.84	86.36	85.86
Micro Average	**88.53**	87.89	88.45	87.40	88.30	87.82	88.13	87.78

Table 4.3: Overall MAP@3 results (%) of the eight individual models.

	$\mathrm{CRNN_A}$	$\mathrm{CRNN_B}$	$\mathrm{VGG13_A}$	$\mathrm{VGG13_B}$	$\mathrm{GCRNN_A}$	$\mathrm{GCRNN_B}$	$\mathrm{GCNN_A}$	GCNN_B
Macro Average	**90.73**	90.51	90.40	89.75	90.64	90.27	90.38	90.22
Micro Average	**92.83**	91.66	91.67	91.01	91.64	91.50	91.55	91.44

The previous results indicate that $\mathrm{CRNN_A}$ achieves the best scores, whereas, $\mathrm{VGG13_B}$ produces the lowest ones. Most importantly, incorporating the recurrent layer (i.e. GRU), has demonstrated a positive impact for assigning sound tags. This improvement is due to combining the strengths of both CNN and RNN, which is well-known for better modeling long temporal sequences [21]. However, our initial analysis does not reveal considerable differences. In addition, according to numerous papers on Statistical Machine Learning, when the results on different categories of data are not comparable, their averages are meaningless [108]. To cope with this shortcoming, appropriate statistical tests should be conducted thoroughly [57]. To this end, we have statistically compared the performances of these techniques using 3 tests: Friedman test, Friedman Aligned test and Quade test. Under the null hypothesis, we have assumed that all systems are equivalents and the observed differences are merely due to chance. Table 4.4 summarizes the obtained statistics.

Table 4.4: Summary of the test statistics.

Audio Tagging System	Friedman Ranking	Friedman Aligned Ranking	Quade Ranking
$\mathrm{CRNN_A}$	**3.70**	**134.16**	**3.56**
$\mathrm{CRNN_B}$	4.58	169.24	4.58
$\mathrm{VGG13_A}$	4.41	149.95	4.38
$\mathrm{VGG13_B}$	5.15	199.61	5.41
$\mathrm{GCRNN_A}$	4.41	155.34	3.81
$\mathrm{GCRNN_B}$	4.52	173.77	4.73
$\mathrm{GCNN_A}$	4.56	160.90	4.68
$\mathrm{GCNN_B}$	4.66	173.02	4.84
Test Statistic	7.63	37.50	3.45
Degrees of Freedom	7	7	7 and 280
p-value	0.36	$\mathbf{3.8\times10^{-6}}$	$\mathbf{1.5\times10^{-3}}$

The results shown in the above table indicate that Friedman Aligned and Quade tests reject the null hypothesis with a very high level of significance ($\text{p-value}_{\mathrm{FA}} = 3.8\times10^{-6}$and $\text{p-value}_{\mathrm{Q}}=1.5\times10^{-3}$), which confirms the existence of at least one pair of systems with significantly different performances. However, the Friedman test fails to reject this hypothesis. This behavior is expected since *this latter test considers that all tags are equal in terms of importance, while the Friedman Aligned and Quade tests take into account the fact that some* *events are more difficult than others, and compute the ranks of each technique across all class labels* [57].

The above results also indicate that Configuration A of the feature extraction technique shows slightly better performance than Configuration B. Furthermore, the MAP@3 scores given in table 4.3 confirms this claim. In order to further investigate this observation, we have followed up these tests with multiple Wilcoxon signed-ranks tests, provided in the following subsection.

4.2.1 Case Study A: impact of feature extraction

This section is devoted to investigating the influence of the parameters used for extracting Log-Mel features on the performance of the deep learning models. To this end, we have carried out pairwise comparisons between CRNN$_A$ and CRNN$_B$, VGG13$_A$ and VGG13$_B$, GCRNN$_A$ and GCRNN$_B$, GCNN$_A$ and GCNN$_B$. Due to its robustness, we have considered using the Wilcoxon signed-ranks test. A summary of this test statistics is shown in Table 4.5. We report in each entry of this table the number of Win/Tie/Loss, on which there is a statistically significant win/loss of the systems trained using Configuration A over Configuration B features. An entry is bold if the number of wins/losses is significant using the Wilcoxon signed-ranks test.

Table 4.5: Summary of the Wilcoxon signed-ranks statistics.

	Deep neural networks architectures			
	VGG13	**CRNN**	**GCNN**	**GCRNN**
W/T/L	**25/0/16**	24/0/17	23/0/18	19/0/22
p-value	**0.05**	0.18	0.63	0.46
Decision	**VGG13$_A$ wins over VGG13$_B$**	Could not reject the null hypothesis	Could not reject the null hypothesis	Could not reject the null hypothesis

We observe in Table 4.5 that, overall, Configuration A wins in most cases. Most importantly, the results indicate that VGG13$_A$ is significantly better than VGG13$_B$with p-value ≤ 0.05. However, data are not sufficient to reach the same conclusion regarding CRNN, GCRNN, and GCNN as depicted in Table 4.5.

Figures 4.2 and 4.3 show the confusion matrices of CRNN$_A$ and CRNN$_B$, respectively, computed on the test set. The matrices indicate appreciable diagonals, meaning that many classes are correctly classified. However, we observe that some classes are easier to classify while the others are not; for instance, in case of Squeak, Scissors and Fireworks, all Audio Tagging systems exhibit low generalization ability (Table 4.2). We believe this behavior occurs due to two main reasons: (1) Some of the mostly misclassified classes are difficult to distinguish even for human beings; for example, Fireworks are sometimes predicted as Gunshots or Tearing sounds (line 18 of CRNN_A confusion matrix). (2) Some of these events are rare and were mostly not manually verified; for instance, Scissors and Telephone events are rare; 78.90% of Squeak sound data are not manually verified. In addition, we observe that for some events such as "Computer Keyboard" and "Knock", systems that were trained using Configuration B features outperform those trained using Configuration A features, as indicated in Table 4.2. Figure 4.1 shows two representations of a sound file of the event "Computer Keyboard".

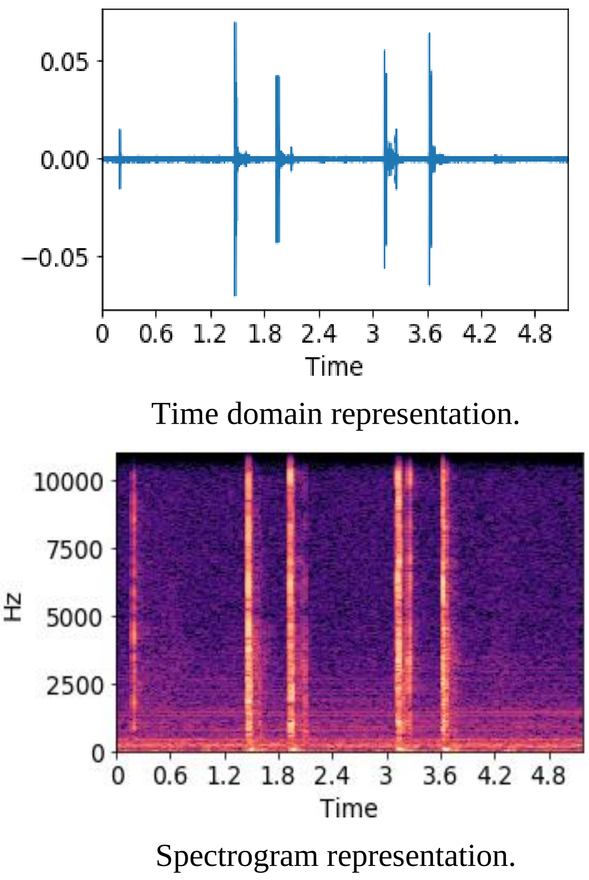


Figure 4.1: **Two representations of a "Computer Keyboard" sound file.**

The event "Computer Keyboard" belongs to the category of impulsive signals. It is worth underscoring that systems built using smaller windows work significantly better on impulsive signals [109]. Recall that Configuration B uses a smaller window for extracting features, whereas, Configuration A uses a larger one, please refer to Section 3.4.2. This fact justifies previous observation concerning the outperformance of Configuration B over Configuration A -based systems in case of impulsive events.

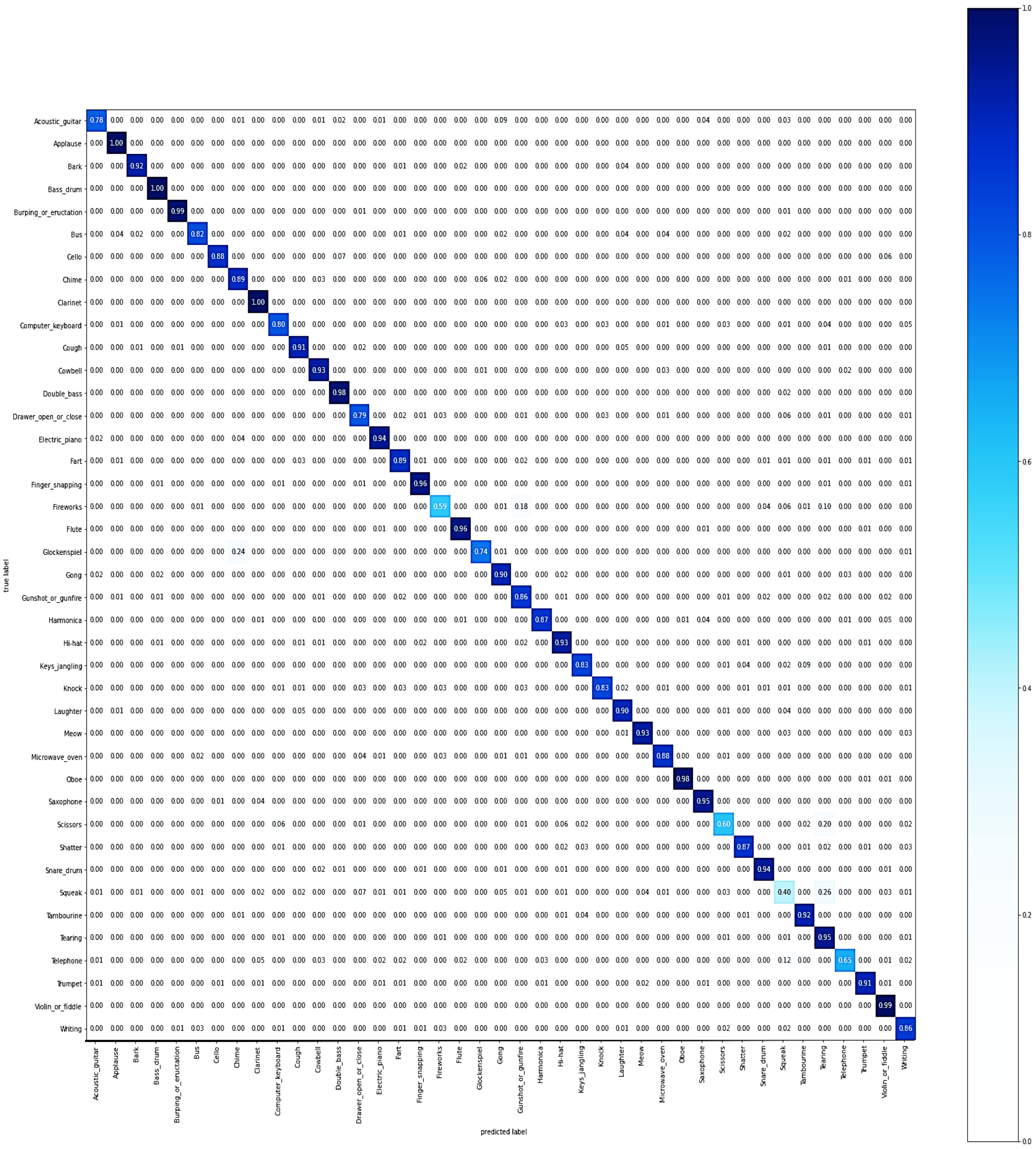


Figure 4.2: CRNN$_A$ **confusion matrix.**

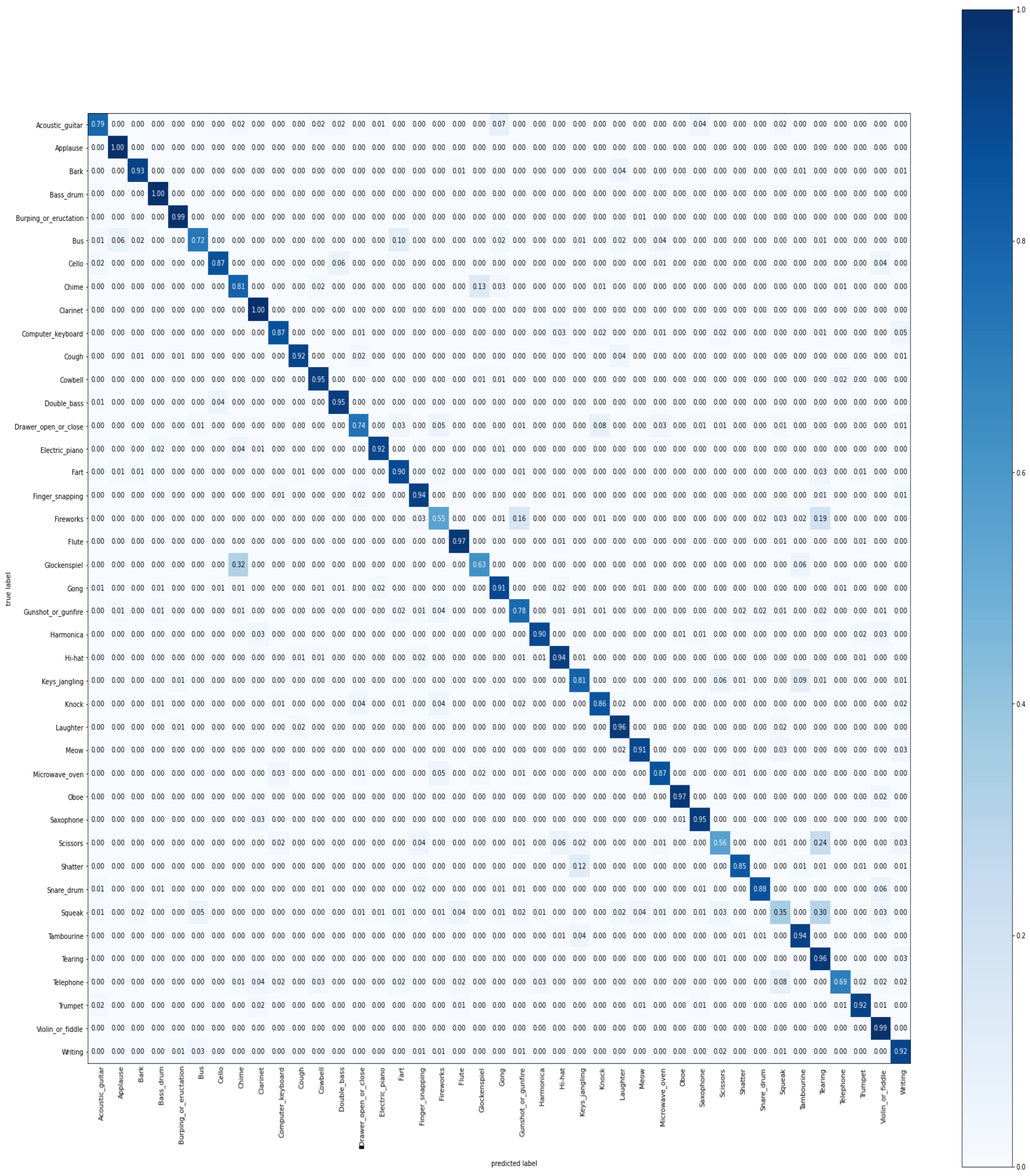


Figure 4.3: CRNN$_B$ confusion matrix.

4.2.2 Case Study B: Impact of the gating mechanism

We aim at studying the impact of the gating mechanism on the performance of our Audio Tagging systems. To this end, we have carried out pairwise comparisons between GCNN$_A$ and VGG13$_A$, GCNN$_B$ and VGG13$_B$, GCRNN$_A$ and CRNN$_A$, GCRNN$_B$ and CRNN$_B$. Similarly, to the previous experiment, we have considered using the Wilcoxon signed-ranks test. A summary of this test statistics is shown in Table 4.6. Similarly, we report in each entry of this table the number of win/Tie/loss of the models which use the gating mechanism over the non-gating-based systems. An entry is bold if the number of wins/losses is significant using the Wilcoxon test.

Table 4.6: Summary of Wilcoxon.

	GCNN$_A$ against VGG13$_A$	**GCNN$_B$against VGG13$_B$**	**GCRNN$_A$ against CRNN$_A$**	**GCRNN$_B$ against CRNN$_B$**
W/T/L	18/0/23	**24/0/17**	16/0/25	22/0/19
p-value	0.29	**0.06**	0.29	0.84
Decision	Could not reject the null hypothesis	**GCNN$_B$ wins over VGG13$_B$**	Could not reject the null hypothesis	Could not reject the null hypothesis

The results shown in Table 4.6 indicate that GCNN$_B$ exhibits significantly better performance compared to VGG13$_B$with $\text{p-value} \leq 0.06$. However, introducing the gating mechanism does not demonstrate any improvement on the remaining models. More specifically, the gating-based systems lose in most cases, which is not expected since these approaches are complex but generally very effective and accurate [3] [15]. Many reasons may cause this behavior. It can be related to the hyperparameters used for training such as the batch size, the learning rate, the number of epochs, etc. In addition, the introduction of the mixup step can make the learning more challenging, as reported by many studies [110] [111]. To further investigate this issue, we have trained GCNN$_A$ for another extra 20 epochs, without varying the other hyperparameters. We report in Figure 4.4 both the training and validation losses during the learning process, and in Figure 4.5 the changes in the average accuracy.

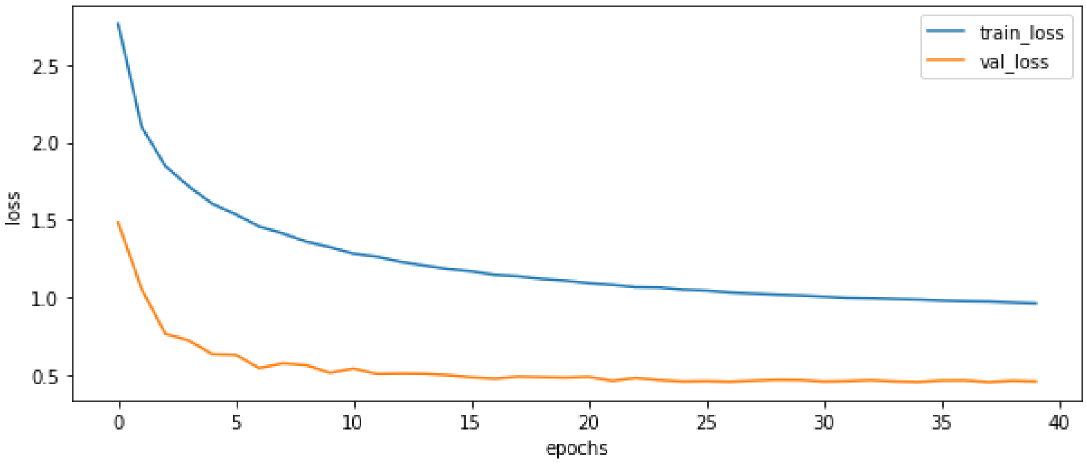


Figure 4.4: Training and validation loss during the learning process of $\mathrm{GCNN_A}$.

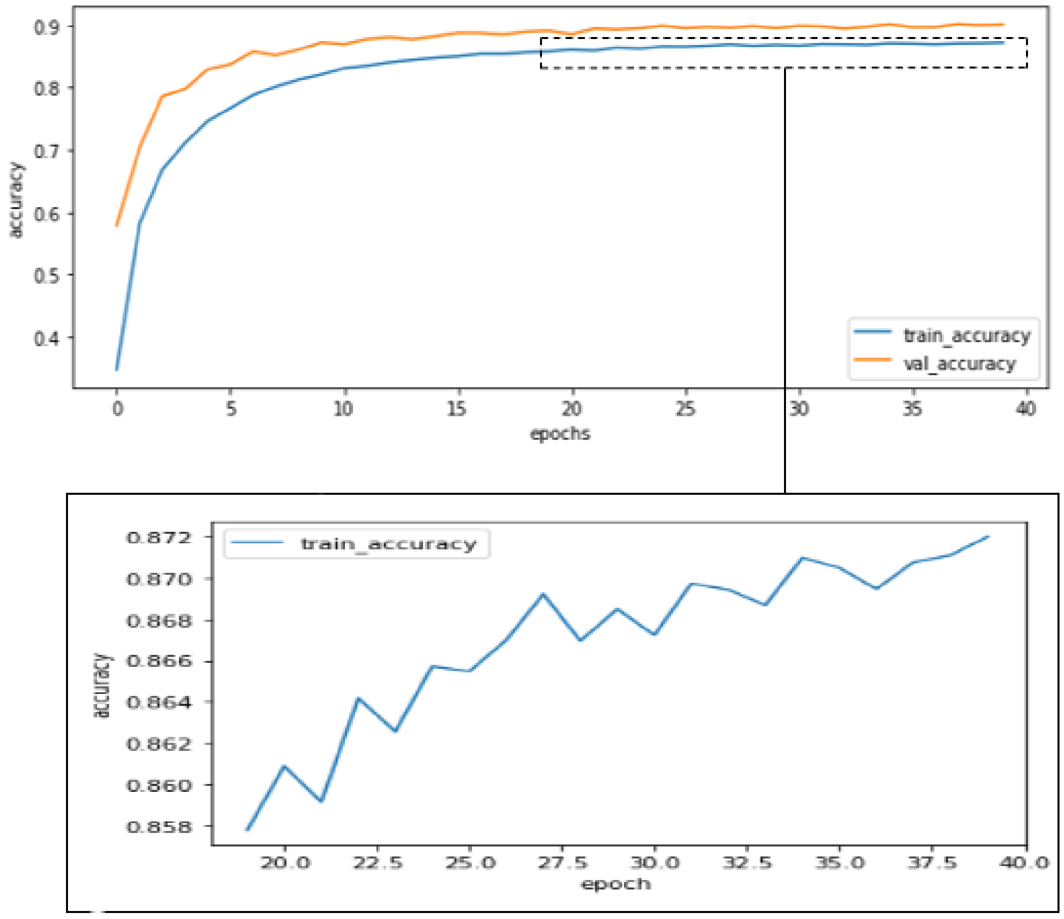


Figure 4.5: Training and validation loss during the learning process of $\mathrm{GCNN_A}$.

The analysis of the results illustrated in Figures 4.4 and 4.5 is summarized as follows: Figure 4.4 indicates that the training loss is higher than the validation loss, contrary to intuition. The main reason for this behavior is that the loss function for validation does not use either regularization nor augmentation, whereas the training process uses more data, resulting in higher average loss values. Similar results have been reported in [112].

As depicted in the zoom of the accuracy plot (Figure 4.5), the performance exhibits an improvement as the number of epochs increases. We observe a rise of 2% between epoch 20 and epoch 40. This latter finding coincides with our initial intuition regarding the parameters setting for the gating models. We can conclude that gating-based systems need to be trained for longer runs.

CONFUSION MATRIX

Figures 4.6 and 4.7 show the confusion matrices of GCRNN_A and GCRNN_B, respectively, computed on the test set. The obtained results (Table 4.2 and Figures 4.6, 4.7) reveal that, for some of the audio classes such as “Applause” “Bass drum” and “Burping”, the gating-based and non-gating-based systems perform similarly. Interestingly, for some rare events such as “Fireworks”, “Scissors” and “Telephone”, the generalization ability of the gating-based systems is higher. Therefore, introducing the gating mechanism can improve the overall performance in case of rare or non-manually verified events. Similar results have been reported in [3] [21].

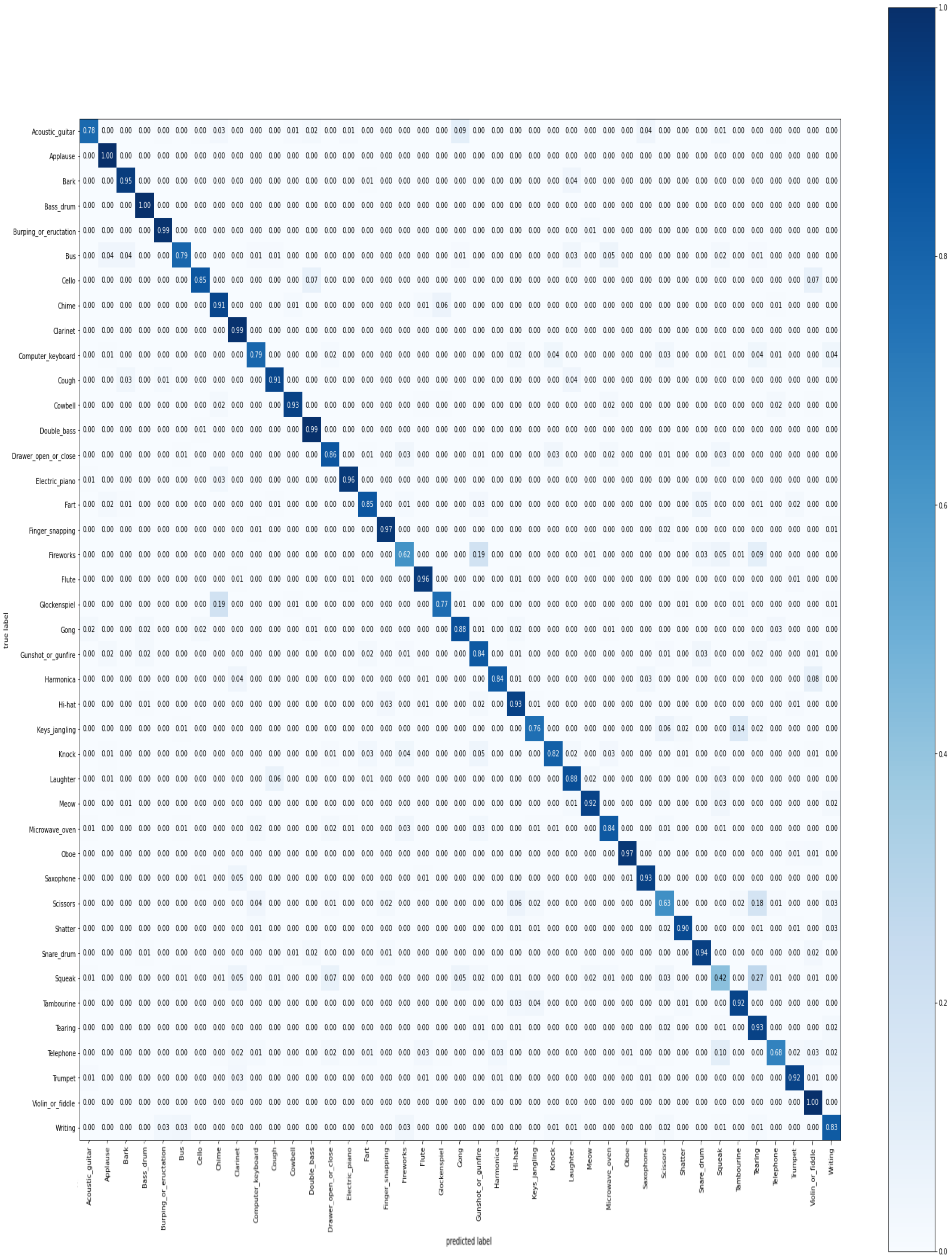
4.2.3 Systems complexity

Table 4.7 gives the trainable parameters and training time of the 4 neural network architectures.

Table 4.7: Number of trainable parameters and training time for each system.

	VGG13	GCNN	CRNN	GCRNN
Trainable parameters	**9 430 761**	18 834 601	12 600 553	22 004 393
Training time	**20h38**	33h	25h	37h50

VGG13 models yield the lowest training time since they have the least number of parameters. The second-best result is attributed to CRNN systems, whereas, the gating-based systems achieve the worst results in terms of the training time and the architecture complexity. Although CRNN does not provide the best training time, it succeeds at capturing most events present in our dataset, requiring bearable training time. It is worth underscoring that these results represent the time required for building our models, while the prediction time is instantaneous for all systems.



	Acoustic_guitar	Applause	Bark	Bass_drum	Burping_or_eructation	Bus	Cello	Chime	Clarinet	Computer_keyboard	Cough	Cowbell	Double_bass	Drawer_open_or_close	Electric_piano	Fart	Finger_snapping	Fireworks	Flute	Glockenspiel
Acoustic_guitar	0.78	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.01	0.02	0.00	0.01	0.00	0.00	0.00	0.00	0.00
Applause	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Bark	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
Bass_drum	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Burping_or_eructation	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Bus	0.00	0.04	0.04	0.00	0.00	0.79	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Cello	0.00	0.00	0.00	0.00	0.00	0.00	0.85	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Chime	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.91	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.06
Clarinet	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Computer_keyboard	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.79	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00
Cough	0.00	0.00	0.03	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.91	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Cowbell	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.93	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Double_bass	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Drawer_open_or_close	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.86	0.00	0.01	0.00	0.03	0.00	0.00
Electric_piano	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.96	0.00	0.00	0.00	0.00	0.00
Fart	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.85	0.00	0.01	0.00	0.00
Finger_snapping	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.00
Fireworks	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.62	0.00	0.00
Flute	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.96	0.00
Glockenspiel	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.19	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.77
Gong	0.02	0.00	0.00	0.02	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Gunshot_or_gunfire	0.00	0.02	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.01	0.00	0.00
Harmonica	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
Hi-hat	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.01	0.00
Keys_jangling	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Knock	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.03	0.00	0.04	0.00	0.00
Laughter	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
Meow	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Microwave_oven	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.02	0.01	0.00	0.00	0.03	0.00	0.00
Oboe	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Saxophone	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
Scissors	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.01	0.00	0.00	0.02	0.00	0.00	0.00
Shatter	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Snare_drum	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.00	0.00	0.00	0.01	0.00	0.00	0.00
Squeak	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.05	0.00	0.01	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00
Tambourine	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Tearing	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Telephone	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.01	0.00	0.00	0.00	0.02	0.00	0.01	0.00	0.00	0.03	0.00
Trumpet	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
Violin_or_fiddle	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Writing	0.00	0.00	0.00	0.00	0.03	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00

	Gong	Gunshot_or_gunfire	Harmonica	Hi-hat	Keys_jangling	Knock	Laughter	Meow	Microwave_oven	Oboe	Saxophone	Scissors	Shatter	Snare_drum	Squeak	Tambourine	Tearing	Telephone	Trumpet	Violin_or_fiddle	Writing
Acoustic_guitar	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
Applause	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Bark	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Bass_drum	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Burping_or_eructation	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Bus	0.01	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.01	0.00	0.00	0.00	0.00
Cello	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.00
Chime	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
Clarinet	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Computer_keyboard	0.00	0.00	0.00	0.02	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.01	0.00	0.04	0.01	0.00	0.00	0.04
Cough	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Cowbell	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00
Double_bass	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Drawer_open_or_close	0.00	0.01	0.00	0.00	0.00	0.03	0.00	0.00	0.02	0.00	0.00	0.01	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00
Electric_piano	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Fart	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.01	0.00	0.02	0.00	0.00
Finger_snapping	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
Fireworks	0.00	0.19	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.03	0.05	0.01	0.09	0.00	0.00	0.00	0.00
Flute	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
Glockenspiel	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01
Gong	0.88	0.01	0.00	0.02	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00
Gunshot_or_gunfire	0.00	0.84	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.03	0.00	0.00	0.02	0.00	0.00	0.01	0.00
Harmonica	0.00	0.00	0.84	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00
Hi-hat	0.00	0.02	0.00	0.93	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
Keys_jangling	0.00	0.00	0.00	0.00	0.76	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.02	0.00	0.00	0.14	0.02	0.00	0.00	0.00	0.00
Knock	0.00	0.05	0.00	0.00	0.00	0.82	0.02	0.00	0.03	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
Laughter	0.00	0.00	0.00	0.00	0.00	0.00	0.88	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00
Meow	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.92	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.02
Microwave_oven	0.00	0.03	0.00	0.00	0.01	0.01	0.00	0.00	0.84	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
Oboe	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00
Saxophone	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.93	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Scissors	0.00	0.00	0.00	0.06	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.63	0.00	0.00	0.00	0.02	0.18	0.01	0.00	0.00	0.03
Shatter	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.90	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.03
Snare_drum	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.94	0.00	0.00	0.00	0.00	0.00	0.02	0.00
Squeak	0.05	0.02	0.00	0.01	0.00	0.00	0.00	0.02	0.01	0.00	0.00	0.03	0.00	0.00	0.42	0.00	0.27	0.01	0.00	0.01	0.00
Tambourine	0.00	0.00	0.00	0.03	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.92	0.00	0.00	0.00	0.00	0.00
Tearing	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.01	0.00	0.93	0.00	0.00	0.00	0.02
Telephone	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.68	0.02	0.03	0.02
Trumpet	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.92	0.01	0.00
Violin_or_fiddle	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
Writing	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.83

Figure 4.6: GCRNN$_A$ **confusion matrix.**

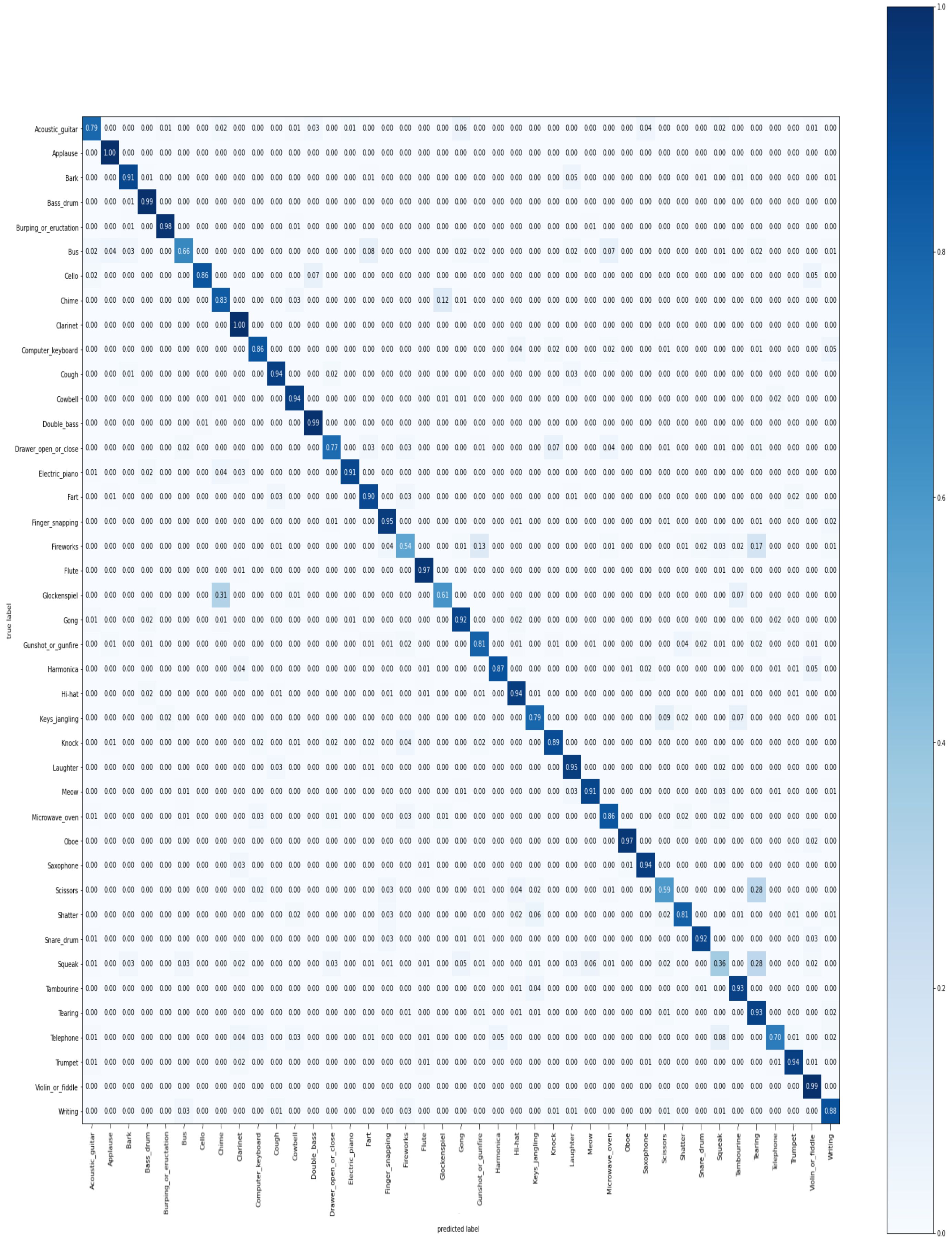


Figure 4.1: GCRNN_B confusion matrix.

4.2.4 Summary of the first of experiment

From the above results we can derive several lessons:

- Some audio events are easier to classify while others are not.
- The introduction of the gating mechanism can improve the overall performance in case of rare or non-manually verified events. In addition, such mechanism works better when the model is trained for longer runs, i.e. higher number of epochs.
- The systems that were trained using a smaller window for extracting features can detect better impulsive events.

However, it is hard to make a firm generalization on the acoustic characteristics of these events that can explain the above observations. In addition, some systems capture sound events that are impulsive, rare, or even non-manually verified, with various confidences. Specifically, the individual models provide sufficiently diverse predictions of the events present in the dataset. Therefore, a proper combination of these models would improve substantially the generalization ability. Ensemble learning combines the strengths of each model by merging their predictions [49]. Numerous studies have demonstrated that amalgamating several learners could improve the generalization ability [45] [113]. An ensemble made of our eight systems would learn: (1) impulsive events; (2) events that are rare or non-manually verified. In the next set of experiments, we further investigate the use of a well-known ensemble learning technique “Stacking”.

4.3 Second Set of Experiment: Ensemble of models

In this section, we have investigated the effect of combining the predictions of the previously trained models. In order to achieve this, we have employed a popular technique called stacking. The resulting system is made of our eight models. Table 4.8 summarizes the obtained performance results. The last row specifies the averaged rank obtained by the Friedman test of each method. Note that we also include the performance of each individual model in order to highlight the improvement provided by Stacking these learners.

Table 4.8: F1-score (%) results of the eight individual models vs STACKED$_8$ model.

Tag	CRNN$_A$	CRNN$_B$	VGG13$_A$	VGG13$_B$	GCRNN$_A$	GCRNN$_B$	GCNN$_A$	GCNN$_B$	STAKED$_8$
Acoustic guitar	85.13	85.06	85.23	85.59	85.74	84.29	86.83	85.32	**87 .50**
Applause	96.69	95.82	95.56	97.28	95.27	96.69	96.99	96.72	**98 .46**
Bark	93.83	93.21	93.25	85.43	93.36	90.35	94.77	88.53	**98 .18**
Bass drum	96.56	96.93	96.20	94.17	96.57	94.90	96.60	95.60	**98 .25**
Burping	98.14	98.16	96.36	99.06	97.24	97.52	96.93	99.08	**100 .0**
Bus	86.05	78.55	84.26	77.78	84.85	74.32	84.69	81.28	**89 .36**
Cello	92.57	91.09	92.60	90.72	90.55	91.86	92.21	89.53	**96 .15**
Chime	**80.36**	71.42	79.71	71.86	81.77	74.08	79.64	71.39	76 .06
Clarinet	93.35	93.97	95.42	93.11	91.14	92.12	92.89	94.46	**100 .0**
keyboard	82.85	87.20	85.86	**89.79**	83.09	86.78	88.54	89.58	88 .46
Cough	88.75	**93.27**	89.63	90.88	90.20	92.49	89.52	92.15	91 .80
Cowbell	92.04	93.44	92.90	91.64	**94.44**	92.09	92.68	92.66	93 .98
Double bass	93.34	92.23	92.45	92.15	92.56	93.63	92.92	92.08	**96 .39**
Drawer	80.01	80.38	82.19	79.26	**85.27**	82.04	83.93	77.78	82 .76
Electric piano	92.66	93.64	93.07	92.60	**96.27**	93.22	94.05	93.94	93 .75
Fart	86.39	85.15	86.09	85.73	86.75	86.58	86.83	87.77	**93 .33**
Finger snapping	95.77	91.49	94.06	92.72	95.54	90.79	93.22	94.33	**96 .97**
Fireworks	**68.33**	61.07	66.22	63.44	69.21	61.69	65.46	65.69	66 .67
Flute	96.56	96.08	95.85	96.52	94.98	96.59	94.59	98.00	**98 .15**
Glockenspiel	81.75	69.87	82.94	70.12	**83.45**	68.20	79.13	67.13	72 .34
Gong	85.46	87.04	85.25	88.11	86.29	87.68	87.67	89.06	**90 .91**
Gunshot	85.60	81.94	84.70	81.73	82.99	84.11	84.05	82.89	**88 .06**
Harmonica	90.20	91.96	91.41	91.68	88.41	89.71	88.67	89.75	**93 .75**
Hi-hat	89.92	91.32	88.94	89.72	89.40	91.55	87.85	91.78	**93 .51**
Keys jangling	**85.55**	79.00	79.50	76.26	82.00	80.63	77.54	77.25	80 .00
Knock	87.99	88.41	85.63	**90.37**	86.43	89.52	87.44	87.10	87 .67
Laughter	88.39	**91.71**	87.84	88.36	87.89	91.17	87.99	89.77	89 .47
Meow	**93.41**	91.99	91.38	89.62	92.75	91.66	90.91	88.98	91 .53
Microwave	**88.17**	87.82	85.22	86.07	84.32	84.70	84.07	85.71	87 .10
Oboe	**97.89**	96.23	98.11	96.04	97.38	96.68	96.28	97.19	97 .56
Saxophone	95.43	96.06	96.18	96.14	94.81	95.59	94.97	96.33	**99 .10**
Scissors	67.62	65.01	67.59	69.06	65.94	66.07	63.00	69.08	**77 .55**
Shatter	89.67	87.89	87.34	80.60	91.85	83.10	85.42	81.53	**90 .91**
Snare drum	92.50	90.86	89.84	88.85	90.67	92.31	91.45	88.53	**97 .14**
Squeak	40.10	41.44	45.30	38.39	46.60	42.07	44.19	40.00	**50 .00**
Tambourine	91.28	89.96	90.23	90.96	90.21	90.12	90.65	89.19	**93 .83**
Tearing	70.75	67.73	70.09	70.44	70.40	66.85	69.98	69.84	**72 .73**
Telephone	76.42	79.94	79.20	78.90	78.25	80.65	79.38	81.70	**82 .98**
Trumpet	92.34	92.21	92.90	91.88	91.68	93.79	91.75	91.23	**94 .74**
Violin	95.90	95.74	96.15	95.71	94.99	95.98	96.27	94.89	**97 .30**
Writing	82.19	85.00	81.78	83.22	82.59	85.34	78.93	**85.58**	84 .75
Macro Average	87.02	86.03	86.69	85.41	86.93	85.84	86.36	85.86	**89 .25**
Micro Average	88.53	87.89	88.45	87.40	88.30	87.82	88.13	87.78	**90 .88**

The results given in Table 4.8 indicate that **STACKED$_8$** outperforms the other methods in most cases. In order to confirm the significance of the observed differences, we have compared the performances of these techniques using the average ranks over the 41 events. Following Demasar's recommendations [48], we have first conducted a Friedman test to statistically compare the performance of these systems, assuming that all systems perform similarly. this test rejects this hypothesis with $X_F^2 = 58.92 > X^2(8) = 47.97$ for $\alpha = 1.0$ 1.0×10^{-7} (X_F^2 is distributed according to the X^2distribution with $9 - 1 = 8$ degrees of freedom), and therefore confirms the existence of at least one pair of techniques with significantly different performances.

Second, because we are only interested in comparing STACKED$_8$with the other alternatives, we proceeded with a Bonferroni Dunn test while considering STACKED$_8$ as the control system. Figure 4.8 shows the results of the Bonferroni-Dunn test at a 0.1% significance level with the critical value $q_{0.001} = 3.83$ and the critical difference $\text{CD} = 2.32$. On the horizontal axis, we represent the average ranks of each method (given in Table 4.8), and we mark using a thick line the interval of one CD to the left and to the right of the average rank of STACKED$_8$Any system with a rank outside this area is significantly different from the control system.

The analysis of Bonferroni-Dunn test results illustrated by Figure 4.8 indicates that STACKED$_8$ has the lowest rank and all the other techniques fall outside the marked interval. Therefore, we can conclude that STACKED$_8$ significantly outperforms the individual models, which is consistent with our initial observations.

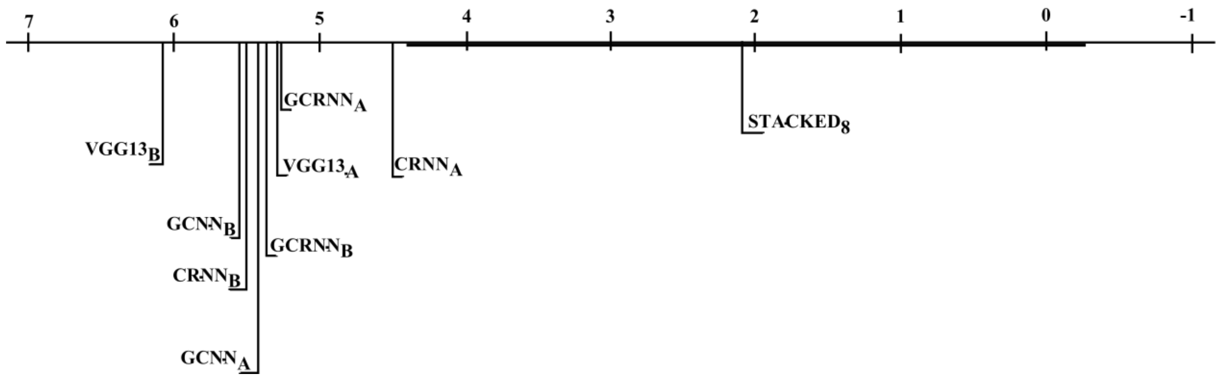


Figure 4.8: Comparison of the Stacked$_8$ model with the other stacked models with the Bonferroni Dunn test.

4.3.1 Case Study A: STACKED$_8$ system

In order to get a better insight on the effect of Stacking on each kind of the sound events, we depict in Figure 4.9 the confusion matrix of STACKED$_8$ estimated on the test set.

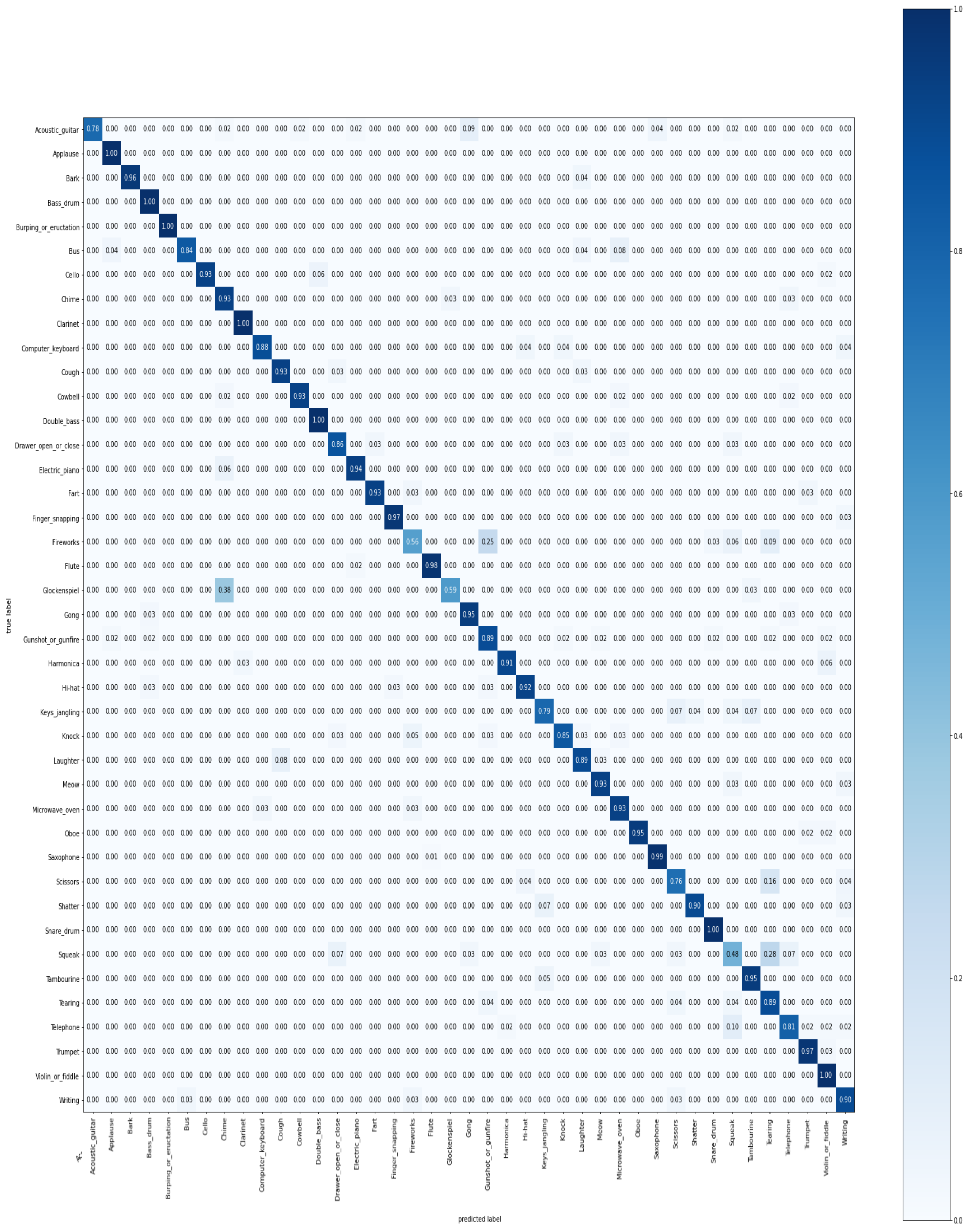


Figure 4.1: STACKED$_8$ confusion matrix.

Based on our comparative analysis of the confusion matrices of STACKED$_8$(Figure 4.8), along with CRNN$_A$ (Figure 4.2), CRNN$_B$ (Figure 4.3), we can derive the following findings:

1. In case of many rare events such as “Scissors” and “Bus”, the stacking considerably improves the performance of the individual learners. For instance, the F1-score of scissors has increased by at least 8.5% (please refer to Table 4.8).
2. In the case of the non-manually verified data such as “Snare drum” (80% of the samples are non-manually verified), the stacking model boosts the score of the best model by 5%. Moreover, the number of false negatives has decreased drastically. For instance, we observed in Figure 4.2 and Figure 4.9. “Snare_drum” has been misclassified with “Violin”,” Hi_hat”, “Gong” and “Cowbell. Stacking has shown remarkable decrease in the number of false negatives.
3. The stacked ensemble provides better tagging scores of the impulsive events than CRNN$_A$ and CRNN$_B$. Specifically, the classification rates of some impulsive sounds like “Gunshots”, “Computer Keyboard” and “Finger snapping” have known a remarkable rise, please refer to Figures 4.3, 4.4 and Table 4.8.

Based on these insights, we conclude that stacking provides an appropriate combination of systems: (1) trained on Configuration A and B features; (2) built using numerous deep neural network architectures; which elevates the generalization ability of the individual models.

4.3.2 Case Study B: Impact of the size of the stacked model

This section is devoted to investigating how the size of the stacked ensemble influences the performance. We have carried out the following experiment. We have trained 4 stacked models; each model is composed of 4 base learners. A summary of these stacked models is given in Table 4.9. We have measured the F1-score of these systems on the test set. The results are provided in Table 4.10. We also report the F1-score of the STACKED$_8$ model as the control system.

Table 4.9: The base learners of the 4 stacked models.

	Base models
Config$_A$	VGG13$_A$, CRNN$_A$, GCNN$_A$, GCRNN$_A$
Config$_B$	VGG13$_B$, CRNN$_B$, GCNN$_B$, GCRNN$_B$
Mix$_{ab}$	VGG13$_A$, CRNN$_A$, VGG13$_B$, CRNN$_B$
Mix$_G$	GCNN$_A$, GCRNN$_A$, GCNN$_B$, GCRNN$_B$

Table 4.10: F1-score (%) results of all the stacked models.

Class	Config$_A$	Config$_B$	Mix$_{ab}$	Mix$_G$	STACKED$_8$
Acoustic guitar	86.42	86.75	87.50	**88.89**	87 .50
Applause	96.97	**98.46**	96.97	96.97	**98 .46**
Bark	96.43	**98.18**	**98.18**	96.43	**98 .18**
Bass drum	96.55	**98.25**	94.92	**98.25**	**98 .25**
Burping	98.46	98.41	**100.00**	98.46	**100 .00**
Bus	**89.36**	82.61	**89.36**	**89.36**	**89 .36**
Cello	93.20	92.31	95.24	95.15	**96 .15**
Chime	**81.82**	69.57	78.26	76.06	76 .06
Clarinet	99.12	97.39	98.25	99.12	**100 .00**
keyboard	88.00	**92.31**	**92.31**	90.20	88 .46
Cough	90.32	**94.92**	91.80	91.80	91 .80
Cowbell	**96.30**	93.98	95.12	**96.30**	93 .98
Double bass	93.98	93.83	95.12	95.24	**96 .39**
Drawer	**89.66**	76.36	82.76	89.29	82 .76
Electric_piano	**95.38**	**95.38**	**95.38**	93.75	93 .75
Fart	88.14	91.80	93.10	**93.33**	**93 .33**
Finger_snapping	**96.97**	**96.97**	**96.97**	**96.97**	**96 .97**
Fireworks	**71.43**	60.38	67.86	65.45	66 .67
Flute	**98.18**	**98.18**	**98.15**	97.25	**98 .15**
Glockenspiel	**84.62**	65.31	75.00	75.00	72 .34
Gong	88.31	**90.91**	**90.91**	**90.91**	**90 .91**
Gunshot	84.38	87.02	**88.37**	87.02	88 .06
Harmonica	93.75	**93.94**	93.75	92.31	93 .75
Hi-hat	**93.67**	92.11	93.51	**93.67**	93 .51
Keys_jangling	**81.48**	77.78	**81.48**	80.77	80 .00
Knock	88.00	87.67	**90.41**	88.00	87 .67
Laughter	89.47	**92.31**	**92.31**	88.00	89 .47
Meow	91.53	89.66	**93.10**	91.53	91 .53
Microwave	**91.53**	87.10	88.52	90.00	87 .10
Oboe	97.56	**97.62**	97.56	97.56	97 .56
Saxophone	97.74	97.72	97.74	98.64	**99 .10**
Scissors	75.00	74.51	**77.55**	74.51	**77 .55**
Shatter	**92.86**	89.29	**92.86**	**92.86**	90 .91
Snare_drum	95.77	94.12	**97.14**	**97.14**	**97 .14**
Squeak	50.00	44.07	**52.63**	51.72	50 .00
Tambourine	92.50	**93.83**	**93.83**	92.50	**93 .83**
Tearing	69.70	73.24	**76.47**	71.64	72 .73
Telephone	81.72	82.22	83.87	**84.44**	82 .98
Trumpet	94.74	**96.00**	94.74	94.74	94 .74
Violin	97.30	96.83	**97.74**	97.30	97 .30
Writing	81.97	**88.14**	86.67	83.33	84 .75
Macro Average	89.27	87.99	**89.84**	89.31	89 .25
Micro Average	90.63	89.69	**91.25**	90.81	90 .88

In order to study these results and reveal significant differences, we have carried out statistical tests. We have first conducted the Friedman test, while assuming that the observed differences are due to random behavior. This test rejects our hypothesis with $\alpha = 0.02$, which indicates an existence of at least one pairwise significant difference. For further analysis of these results, we have compared these scores in a pairwise manner based on the Wilcoxon test in Table 4.11. The first row of each entry specifies the number of Win/Tie/Loss of the technique in the column over the technique in the row; whereas, the second row shows the p-values for the Wilcoxon test. If the entry is bold, this means that the number of wins/losses over 41 is statistically significant using the Wilcoxon test.

Table 4.11: Pairwise comparisons of F1-score results based on Wilcoxon signed-ranks test.

		Config$_A$	**Config$_B$**	**Mix$_G$**	**Mix$_{ab}$**
STACKED$_8$	W/T/L	14/0/27	**15/0/26**	15/1/25	**21/1/19**
	p-value	0.24	**0.02**	0.65	**0.08**
Config$_A$	W/T/L		17/3/21	15/15/11	**21/10/10**
	p-value		0.41	0.32	**0.05**
Config$_B$	W/T/L			**23/5/13**	**26/7/8**
	p-value			**0.03**	**0.0006**
Mix$_G$	W/T/L				**20/10/11**
	p-value				**0.04**

We can sum-up the analysis of the previous results as follows:

(1) Overall, Mix$_{ab}$ yields a remarkable performance. Most importantly, it significantly outperforms the STACKED$_8$ model by a p-value= 0.08. A possible explanation of this behavior might be related to the correlation among the individual members of the STACKED$_8$ model. It is widely acknowledged that an ensemble made of correlated, or non-diverse, members leads to lower generalization power [114]. We believe that the STACKED$_8$ ensemble is composed of highly correlated members, which justifies the obtained results. A further investigation and experimentation is required to solidify our conclusion. These results suggest an appealing future work direction.

(2) Config$_B$exhibits very poor performance, which is expected since, according to our previous findings (case study A from the first set of experiments), Configuration B-based systems have delivered better results on impulsive events, but have exhibited an overall weaker performance than the other counterparts. Note that our dataset is

not domain specific. It is composed of events that some of them are impulsive, whereas, some are not. Therefore, configuration B may not generalize well which justifies the reported weak performance. This fact justifies the reported low performance, which is consistent with our previous findings.

(3) The Gating-based systems are significantly worse than the non-gating-based systems, which is consistent with our previous findings. As highlighted in case study B (first set of experiments), we have trained the gating-based models for 20 epochs only. Our experimental investigation Case study B (first set of experiments) has shown that such systems should been trained for longer runs.

Based on these observations, we can conclude that the size of the ensemble can significantly influence the power of Stacking. Specifically, our analysis indicates that a smaller ensemble can yield better results. Note that several experimental and theoretical studies have shown that large ensembles do not always guarantee better predictive performance [115] [116] [117] This fact coincides with our previous conclusions. Furthermore, stacking the gating-based systems causes a drop in the predictive scores, which is expected since our individual models have not been well trained, please refer to Case study A (first set of experiments) for additional justification.

4.4 Conclusion and summary of experimental findings

From these experiments, we can derive 4 lessons:

1. Integrating gated recurrent units within CNN can induce better systems for Audio Tagging.
2. Introducing (i) the gating mechanism and (ii) the feature extraction configurations yield models which produce diverse and complementary predictions.
3. Stacking demonstrates a remarkable improvement of the individual learners' performances. Most importantly, it provides a proper fusion of their predictions, which leads to better handling of events, including rare and impulsive cases.
4. Combining a larger number of models can entail a deterioration in the overall performance. An ensemble made of VGG13$_A$, CRNN$_A$, VGG13$_B$and CRNN$_B$yields significantly better scores than STACKED$_8$.

CONCLUSION

The primary goal of this thesis was to conduct an **empirical analysis and comparisons among Single-tag Audio Tagging systems** that are able to recognize an increased number of sound events of very diverse nature. To this end, we have carried out **two sets of experiments** on FSDKaggle2018 dataset. Our main contribution is three folds: (1) We describe the major steps involved in the design and analysis of several Audio Tagging systems. (2) We present extensive experimental comparisons founded on strong statistical tests. (3) We test the impact of stacking 8 deep learning models on the overall performance. A detailed description of our work is provided below. First, we have investigated four deep learning architectures (VGG13, GCNN, CRNN and GCRNN). In addition, we have trained these models on Log-Mel Spectrogram features using two different configurations for extraction. From this experimental study, we can derive the following conclusions:

1. The integration of gated recurrent units within CNN (i.e. CRNN) can induce better systems for Audio Tagging.
2. The gating mechanism works better when the model is trained for longer runs, i.e. higher number of epochs.
3. The mechanism for extracting the feature sets plays an important role for designing Audio Tagging systems. Models trained on features extracted using a smaller window are better at capturing impulsive events, but show an overall weak performance. In addition, the confusion matrices indicate that the mechanism for extracting features induce models which produce diverse yet complementary predictions.

Second, motivated by the fact that the individual models produce diverse predictions, we have considered ensembling the predictions of all the trained models through stacking. Based on our analysis, we can conclude that:

1. Stacking demonstrates a remarkable improvement of the individual learners' performances. Most importantly, it provides a proper fusion of their diverse predictions, which better captures events, including rare and impulsive cases.
2. Combining a larger number of models can entail a deterioration in the overall performance. An ensemble made of VGG13_A, CRNN_A, VGG13_B and CRNN_B yields significantly better scores than $\text{STACKED}_{\mathbf{8}}$.

Limits and Future work

In the previous section, we have summarized our main contributions. However, it is of paramount importance to specify the limitations of our work and highlight potential future work directions. The success of the ensembling method relies mainly on the fusion of the individual learners predictions. However, in our experiments we have only considered the stacking mechanism. An appealing work direction would be to thoroughly investigate other fusion methods such as Grading [118], Adaptive fusion and co-operative training [119], Mathematical programming [49], and even ensemble pruning techniques [120]. Similarly, the hyperparameters used for training the deep learning models considerably affect the generalization ability. For instance, in the first set of experiments, we have found that training the gating-based models for longer runs improves the performance. Due to the lack of a dedicated computational platform, we have trained our models for 20 epochs only. A natural extension of this work would be to investigate tuning several hyperparameters like: the number of epochs, the batch size, the learning rate, and exploring other data augmentation techniques. The majority of the available sound datasets are made of a large amount of labeled and unlabeled data. To cope with this matter, it would be interesting to investigate semi-supervised approaches such as pseudo-labeling [121], which offers an elegant mathematical model that serves this purpose very well.

During this project, we have encountered many struggles. The training of the learning models took a very long time due to the lack of dedicated computational platforms. In addition, when performing model selection, storing the trained classifiers caused a considerable increase in the usage of memory space.

This field of research is interesting as it contributes directly to the development of smart cities. We have acquired knowledge and many skills throughout the past 8 months, such as: fundamentals of Machine Learning and key steps for conducting proper Machine Learning experiments. We have also learned the analysis of the experimental findings based on statistical tests. Moreover, we have mastered Python and have discovered “Google Collaboratory” platform that we will continue using for future machine learning projects.

APPENDIX A: DEMONSTRATION

A.1 Introduction

In this section, we present the web application we developed for our Audio tagging systems. First, in Section A.1 we present the tools that we have used for development. Next, in Section A.2 we describe all the possible interaction with the application.

A.2 Development tools

Hardware device/ Environmental setup

In order to create this website, we have used the following resources:
The application was developed on a desktop running on Windows 10 operating system, with Intel Core i4 and 4 Gb RAM.

Table A.1: Development tools.

Tools	Description /functionality
FLASK	Flask is a micro-framework designed to create a web application in a short time. We have used flask for the backend development of this application. It only implements the core functionality giving developers the flexibility to add the feature as required during the implementation.
HTML	Hypertext Markup Language is the standard markup language for documents designed to be displayed in a web browser.
CSS	Cascading Style Sheet is a style language that separates the style of a web document from its content. It is used to customize the layout and control the appearance of web pages written by markup languages.
BOOTSTRAP	Bootstrap is a popular front-end development framework that includes HTML, CSS and JS components. We have used bootstrap to facilitate the design of our webpages.
PYTHON	Python is an object-oriented open source programming language. We used python as the main programming language for this project.
JQUERY	Jquery is a lightweight JavaScript library used for enabling the interactivity of the web pages.
TENSORFLOW	TensorFlow is an open-source software library for high performance numerical computation. It is a Machine Learning tool mainly designed to process neural network models.
KERAS	Keras is a high-level neural network API, Keras is written in pure Python and based on Tensorflow.

A.3 Developed system

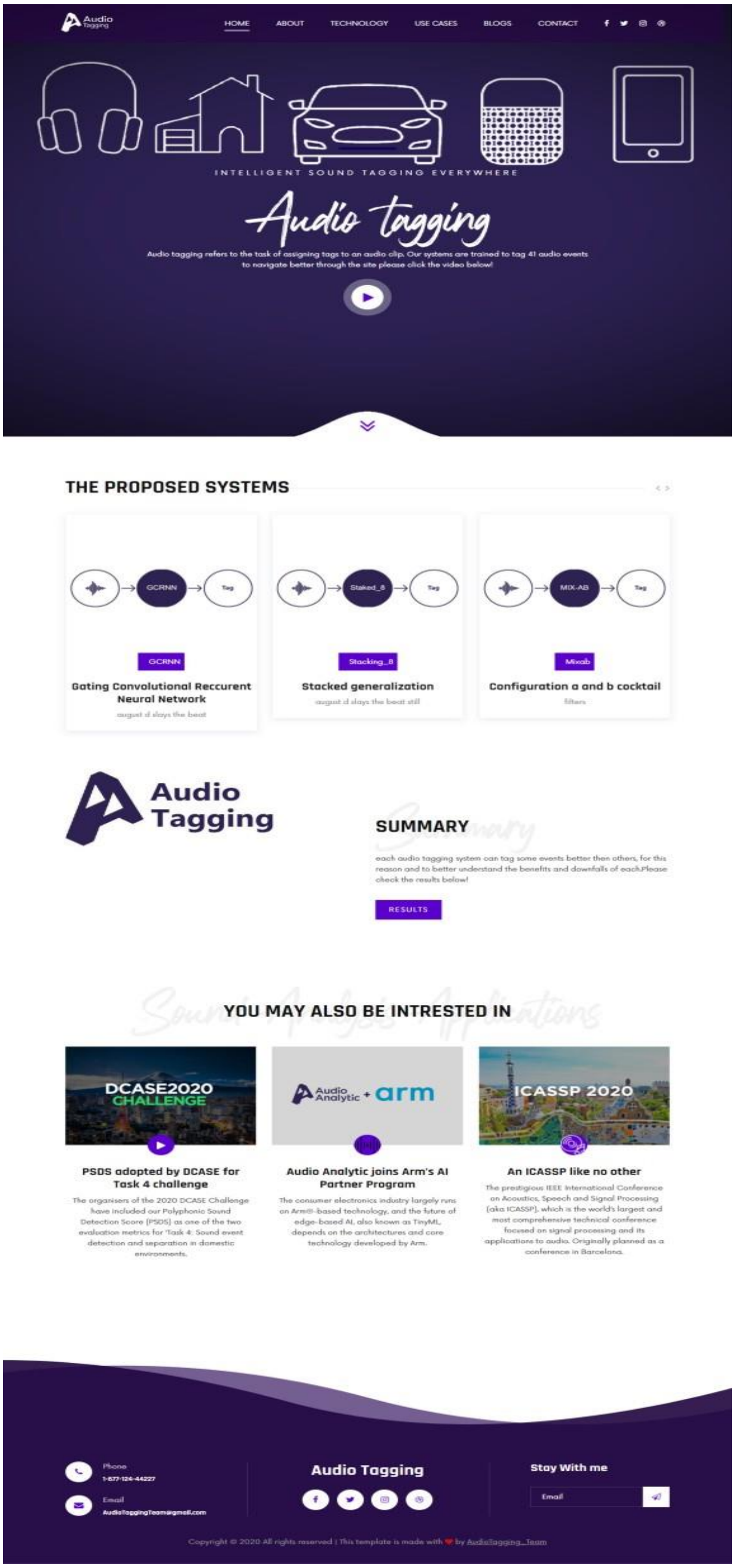
In the pipeline of designing Audio Tagging systems, creating the model is the hardest, but it is not the end. In order to benefit from the created models, we deployed the pretrained models online to be accessed by internet users using Flask micro web framework, Python, HTML (HyperText Markup Language), CSS (Cascading Style Sheet), and JavaScript. Simple web pages are built where the user can access the application and upload an audio clip to the server. Based on the deployed models, the audio clip is classified and its class label is returned back to the user with additional information on the process. The following sections demonstrate several parts of the application:

Home

The home page (Figure A.1) is the main page where the user can navigate to different parts of the application. it consists of three main parts:

- **Header** which is on the top of the page. It consists of five tabs (About, Technology, Use Cases, Blogs and Contact page) and a logo. When the user clicks on one of these tabs the corresponding page is displayed.
- **The work area** which is located below the navigation bar is where the primary interaction with the app occurs. It includes a list of our systems (VGG13$_A$, VGG13$_B$, CRNN$_A$, CRNN$_B$, GCNN$_A$, GCNN$_B$, GCRNN$_A$, GCRNN$_B$ and the five stacked models**).** In order to access the aforementioned systems hover and click on their corresponding card.
- **Footer** located below the work area. Here, we can find “Contact” links in the bottom center of the footer (email, phone etc.) as well as our social media accounts.

The rest of the web pages follow the same template as the home page depicted in the Figure below.



INTELLIGENT SOUND TAGGING EVERYWHERE

Audio tagging

Audio tagging refers to the task of assigning tags to an audio clip. Our systems are trained to tag 41 audio events to navigate better through the site please click the video below!

THE PROPOSED SYSTEMS

GCRNN
Gating Convolutional Reccurent Neural Network
august d slays the beat

Stacking_8
Stacked generalization
august d slays the beat still

Mixab
Configuration a and b cocktail
filters

Audio Tagging

SUMMARY

each audio tagging system can tag some events better then others, for this reason and to better understand the benefits and downfalls of each.Please check the results below!

RESULTS

YOU MAY ALSO BE INTRESTED IN

PSDS adopted by DCASE for Task 4 challenge

The organisers of the 2020 DCASE Challenge have included our Polyphonic Sound Detection Score (PSDS) as one of the two evaluation metrics for 'Task 4: Sound event detection and separation in domestic environments.

Audio Analytic joins Arm's AI Partner Program

The consumer electronics industry largely runs on Arm®-based technology, and the future of edge-based AI, also known as TinyML, depends on the architectures and core technology developed by Arm.

An ICASSP like no other

The prestigious IEEE International Conference on Acoustics, Speech and Signal Processing (aka ICASSP), which is the world's largest and most comprehensive technical conference focused on signal processing and its applications to audio. Originally planned as a conference in Barcelona.

Phone
1-877-124-44227

Email
AudioTaggingTeam@gmail.com

Audio Tagging

Stay With me

Email

Copyright © 2020 All rights reserved | This template is made with ♥ by AudioTagging_Team

Figure A.1: The home page.

The System web Page

When the user chooses an audio tagging system, the page shown in Figure A.2 pops up on the screen. The Audio tagging module consists of an audio player which contains the general information of the audio file (current time/end time, name of the audio file and the pause /play button) that helps users control the audio player. Note that we have dedicated a separate web page for each individual model (VGG13$_A$, VGG13$_B$, CRNN$_A$, CRNN$_B$, GCNN$_A$, GCNN$_B$, GCRNN$_A$, GCRNN$_B$). Note that the Stacked models (STACKED$_8$, Config$_A$, Config$_B$, Mix$_G$ and Mix$_{ab}$) are given in a separate page (Figure A.2) in order to observe the differences in between the stacked models and their base learner. The process of prediction is as follows:

1. First, the user chooses the audio file to upload from the internal storage (computer) to the web server by clicking on the upload button. Note that our web application only deals with (.wav) audio files and any other extension is not supported. Figure A.3 depicts the upload process.

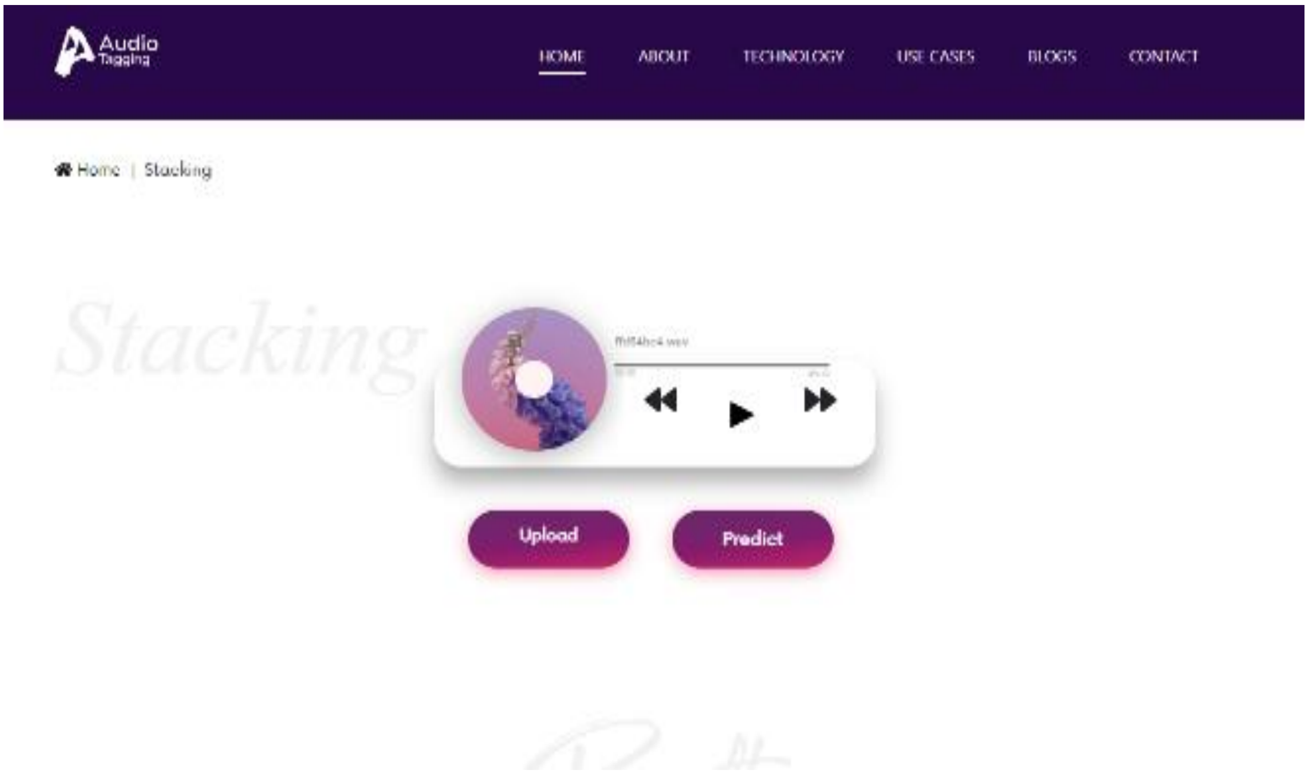
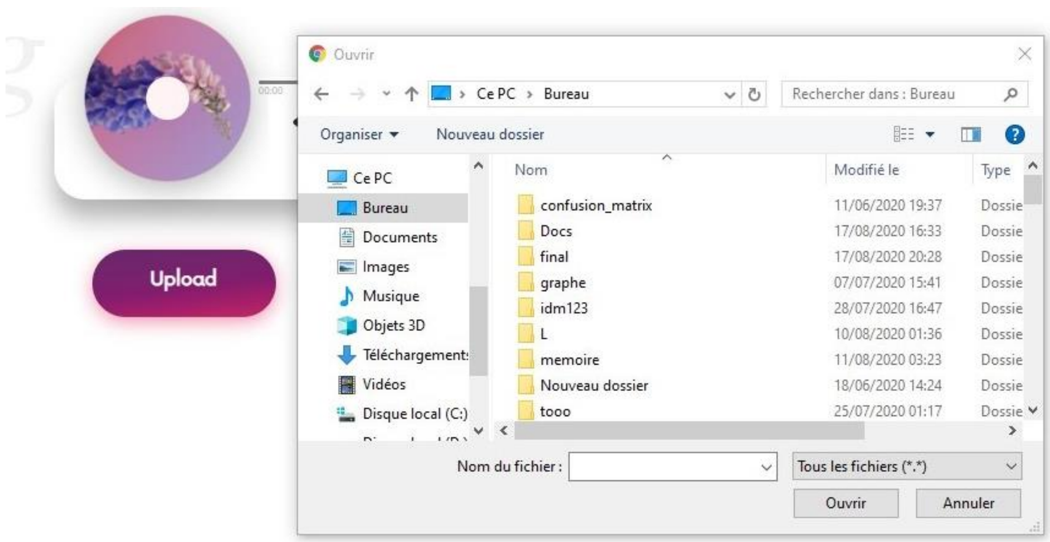


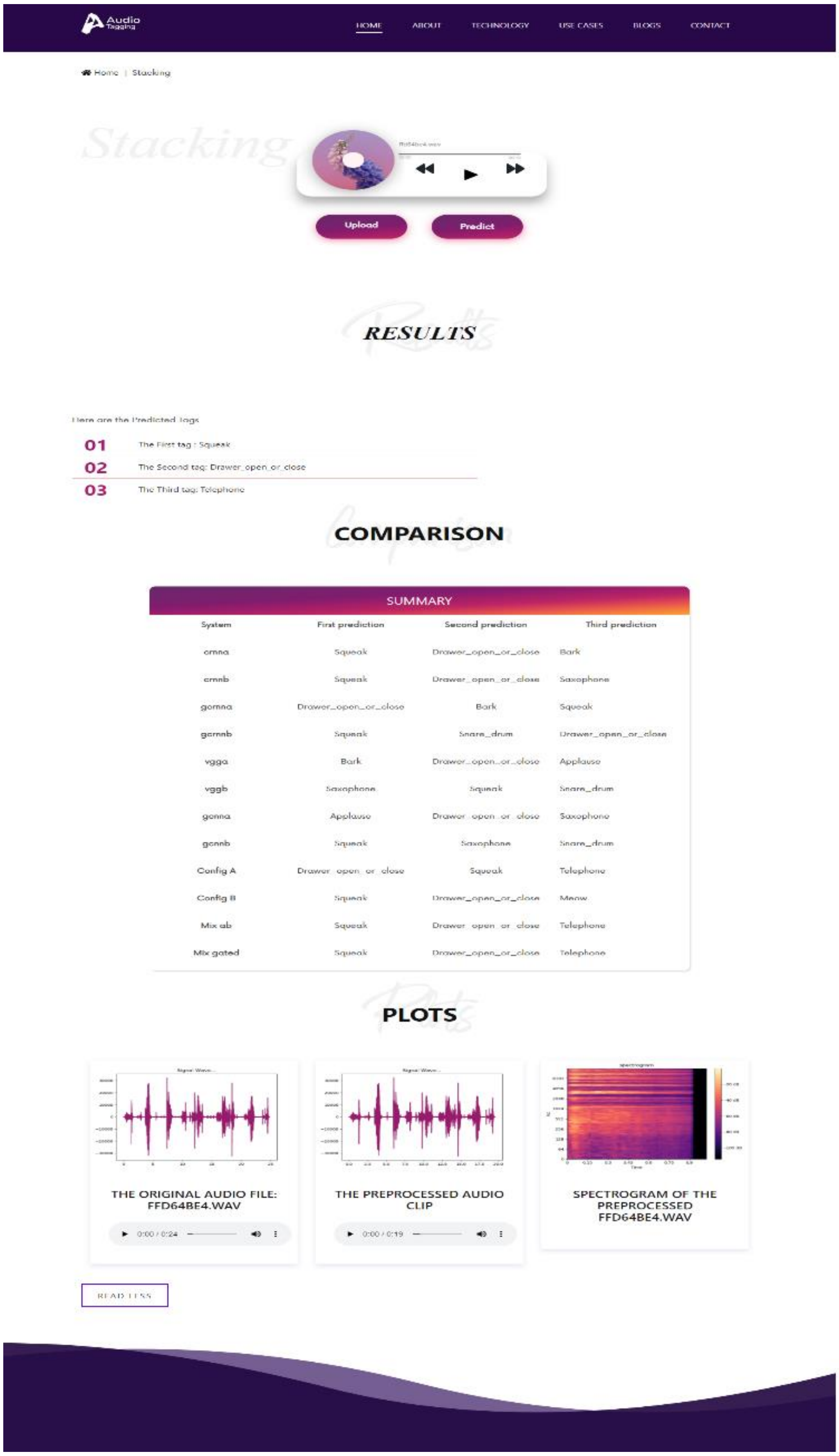
Figure A.2: The system page before any prediction process.



Nom	Modifié le	Type
confusion_matrix	11/06/2020 19:37	Dossie
Docs	17/08/2020 16:33	Dossie
final	17/08/2020 20:28	Dossie
graphe	07/07/2020 15:41	Dossie
idm123	28/07/2020 16:47	Dossie
L	10/08/2020 01:36	Dossie
memoire	11/08/2020 03:23	Dossie
Nouveau dossier	18/06/2020 14:24	Dossie
tooo	25/07/2020 01:17	Dossie

Figure A.3: The upload process.

2. Second, once the audio file is selected, the user can click on the predict button (below the audio player). This process goes through the following steps: first the Feature Engineering process (pre-processing, acoustic feature extraction), followed by the classification process. When the prediction process is done the results are shown at the bottom of the web page.
3. Third, the result section for the individual models contains a list of the 3 best predictions tags ordered from the most probable to the least. Note that true tags are displayed in green. Additional information are provided when the user chooses the" More details" options. This latter includes details on the feature engineering step. Moreover, the user can play the audio file and its preprocessed version, view time representation of the audio files and the Spectrogram resulted from the feature extraction step. As for the stacked models page, the result section contains an additional table displaying the predictions for the four 4 stacked models and their base models as shown in Figure A.5.



RESULTS

Here are the Predicted Tags

01	The First tag : Squeak
02	The Second tag: Drawer_open_or_close
03	The Third tag: Telephone

COMPARISON

SUMMARY

System	First prediction	Second prediction	Third prediction
omna	Squeak	Drawer_open_or_close	Bark
omnb	Squeak	Drawer_open_or_close	Saxophone
gomna	Drawer_open_or_close	Bark	Squeak
gomnb	Squeak	Snare_drum	Drawer_open_or_close
vgga	Bark	Drawer_open_or_close	Applause
vggb	Saxophone	Squeak	Snare_drum
gonna	Applause	Drawer_open_or_close	Saxophone
gonnb	Squeak	Saxophone	Snare_drum
Config A	Drawer_open_or_close	Squeak	Telephone
Config B	Squeak	Drawer_open_or_close	Meow
Mix ab	Squeak	Drawer_open_or_close	Telephone
Mix gated	Squeak	Drawer_open_or_close	Telephone

PLOTS

THE ORIGINAL AUDIO FILE: FFD64BE4.WAV

0:00 / 0:24

THE PREPROCESSED AUDIO CLIP

0:00 / 0:19

SPECTROGRAM OF THE PREPROCESSED FFD64BE4.WAV

READ LESS

Figure A.4: The results page.

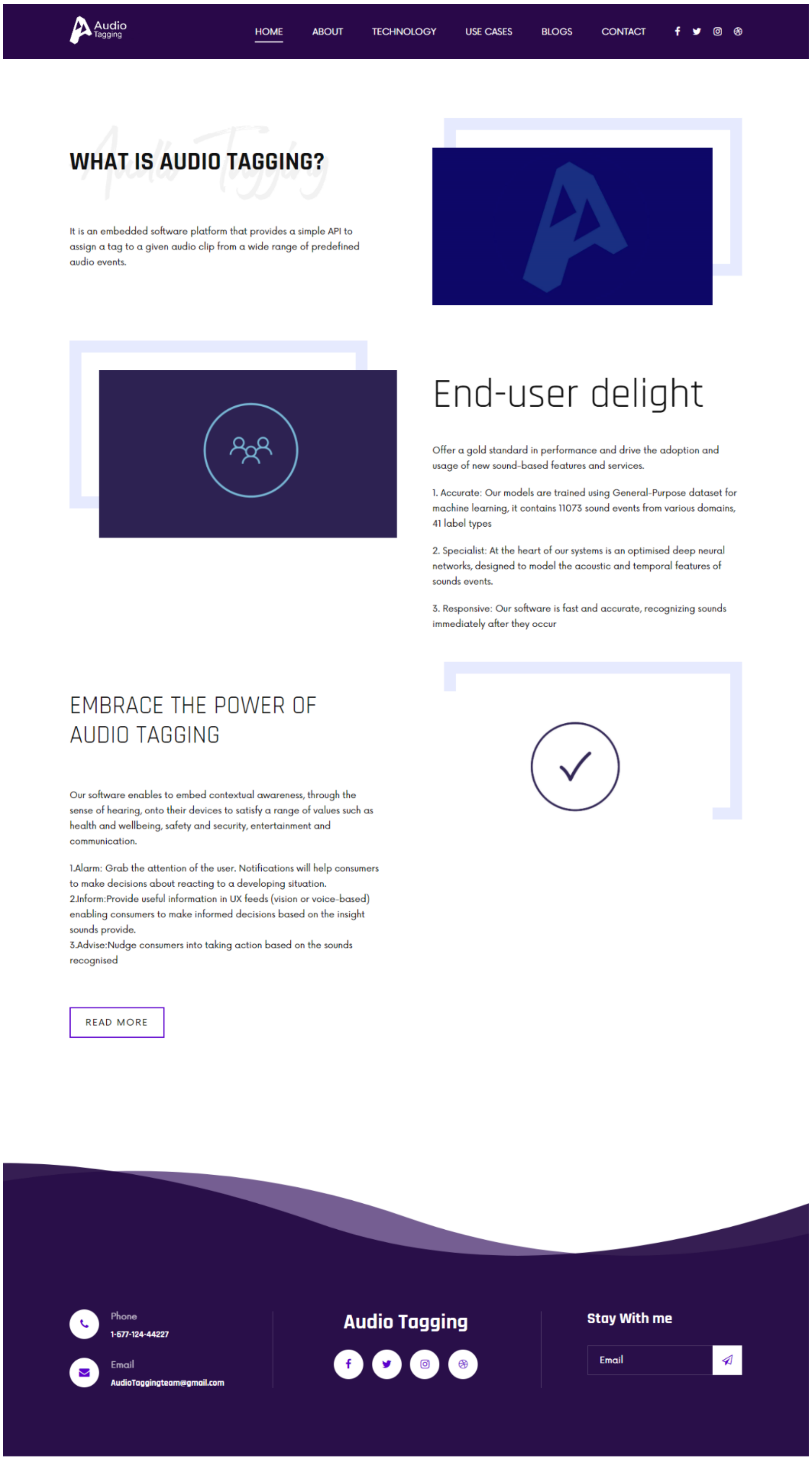
Additional web pages

The about page (Figure A.5) contains the general description of audio tagging, an overview of the applications characteristics and advantages.

Use cases page (Figure A.6) shows a brief description of a wide range of applications related to Audio Tagging.

Blogs (Figure A.7) includes some interesting blogs and articles related to audio tagging and many other sound analysis tasks.

Technology (Figure A.8) provides a brief description on the main concepts used to build a system capable of recognizing wide variety real-world sounds.



WHAT IS AUDIO TAGGING?

It is an embedded software platform that provides a simple API to assign a tag to a given audio clip from a wide range of predefined audio events.

End-user delight

Offer a gold standard in performance and drive the adoption and usage of new sound-based features and services.

1. Accurate: Our models are trained using General-Purpose dataset for machine learning, it contains 11073 sound events from various domains, 41 label types

2. Specialist: At the heart of our systems is an optimised deep neural networks, designed to model the acoustic and temporal features of sounds events.

3. Responsive: Our software is fast and accurate, recognizing sounds immediately after they occur

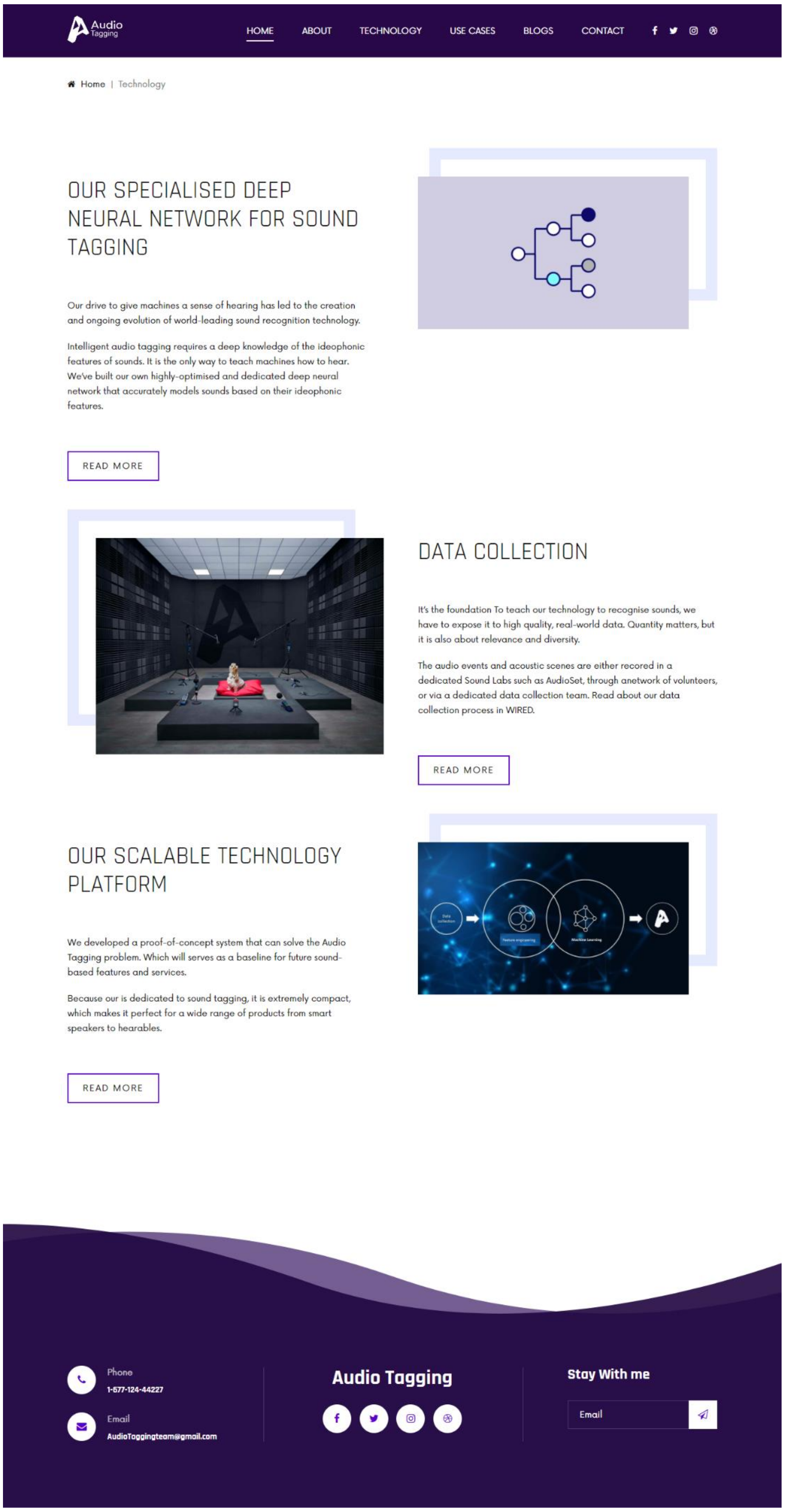
EMBRACE THE POWER OF AUDIO TAGGING

Our software enables to embed contextual awareness, through the sense of hearing, onto their devices to satisfy a range of values such as health and wellbeing, safety and security, entertainment and communication.

1.Alarm: Grab the attention of the user. Notifications will help consumers to make decisions about reacting to a developing situation.
2.Inform:Provide useful information in UX feeds (vision or voice-based) enabling consumers to make informed decisions based on the insight sounds provide.
3.Advise:Nudge consumers into taking action based on the sounds recognised

Phone
1-677-124-44227
Email
AudioTaggingteam@gmail.com

Figure A.5: The about page.



OUR SPECIALISED DEEP NEURAL NETWORK FOR SOUND TAGGING

Our drive to give machines a sense of hearing has led to the creation and ongoing evolution of world-leading sound recognition technology.

Intelligent audio tagging requires a deep knowledge of the ideophonic features of sounds. It is the only way to teach machines how to hear. We've built our own highly-optimised and dedicated deep neural network that accurately models sounds based on their ideophonic features.

DATA COLLECTION

It's the foundation To teach our technology to recognise sounds, we have to expose it to high quality, real-world data. Quantity matters, but it is also about relevance and diversity.

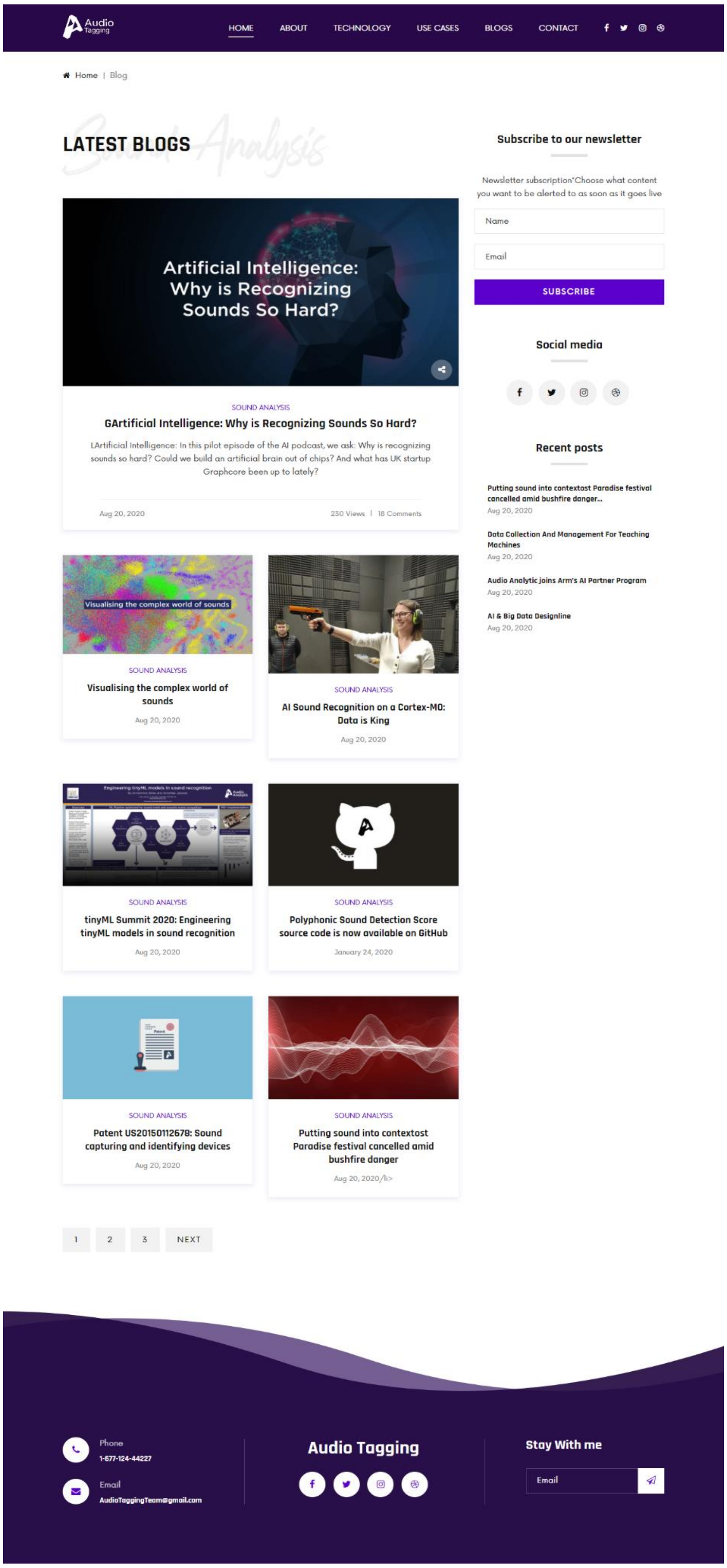
The audio events and acoustic scenes are either recored in a dedicated Sound Labs such as AudioSet, through anetwork of volunteers, or via a dedicated data collection team. Read about our data collection process in WIRED.

OUR SCALABLE TECHNOLOGY PLATFORM

We developed a proof-of-concept system that can solve the Audio Tagging problem. Which will serves as a baseline for future sound-based features and services.

Because our is dedicated to sound tagging, it is extremely compact, which makes it perfect for a wide range of products from smart speakers to hearables.

Figure A.6: The use case page.


Audio Tagging
HOME
ABOUT
TECHNOLOGY
USE CASES
BLOGS
CONTACT
Home | Blog

LATEST BLOGS

Sound Analysis
Artificial Intelligence: Why is Recognizing Sounds So Hard?

SOUND ANALYSIS

GArtificial Intelligence: Why is Recognizing Sounds So Hard?

LArtificial Intelligence: In this pilot episode of the AI podcast, we ask: Why is recognizing sounds so hard? Could we build an artificial brain out of chips? And what has UK startup Graphcore been up to lately?

Aug 20, 2020 230 Views | 18 Comments

Visualising the complex world of sounds

SOUND ANALYSIS

Visualising the complex world of sounds

Aug 20, 2020

SOUND ANALYSIS

AI Sound Recognition on a Cortex-M0: Data is King

Aug 20, 2020

SOUND ANALYSIS

tinyML Summit 2020: Engineering tinyML models in sound recognition

Aug 20, 2020

SOUND ANALYSIS

Polyphonic Sound Detection Score source code is now available on GitHub

January 24, 2020

SOUND ANALYSIS

Patent US20150112678: Sound capturing and identifying devices

Aug 20, 2020

SOUND ANALYSIS

Putting sound into contextost Paradise festival cancelled amid bushfire danger

Aug 20, 2020/li>

Subscribe to our newsletter

Newsletter subscription'Choose what content you want to be alerted to as soon as it goes live

Social media

Recent posts

Putting sound into contextost Paradise festival cancelled amid bushfire danger...
Aug 20, 2020

Data Collection And Management For Teaching Machines
Aug 20, 2020

Audio Analytic joins Arm's AI Partner Program
Aug 20, 2020

AI & Big Data Designline
Aug 20, 2020

Phone
1-677-124-44227

Email
AudioTaggingTeam@gmail.com

Audio Tagging

Stay With me

Figure A.1 : The blogs page.

REFERENCES

[1] T. Virtanen, M. D. Plumbley, and D. Ellis, “Computational analysis of sound scenes and events,” in *Computational Analysis of Sound Scenes and Events*, 2017, pp. 16–41.

[2] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, “Semantic annotation and retrieval of music and sound effects,” *IEEE Trans. Audio, Speech Lang. Process.*, 2008, doi: 10.1109/TASL.2007.913750.

[3] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, “Large-Scale Weakly Supervised Audio Classification Using Gated Convolutional Neural Network,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2018-April, pp. 121–125, 2018, doi: 10.1109/ICASSP.2018.8461975.

[4] I.-Y. Jeong and H. Lim, “Audio tagging system using densely connected convolutional networks,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, 2018.

[5] S. Dimitrov, J. Britz, B. Brandherm, and J. Frey, “Analyzing sounds of home environment for device recognition,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 2014, doi: 10.1007/978-3-319-14112-1_1.

[6] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, “A survey of audio-based music classification and annotation,” *IEEE Trans. Multimed.*, 2011, doi: 10.1109/TMM.2010.2098858.

[7] S. Chu, S. Narayanan, C. C. Jay Kuo, and M. J. Matarić, “Where am I? Scene recognition for mobile robots using audio features,” *2006 IEEE Int. Conf. Multimed. Expo, ICME 2006 - Proc.*, vol. 2006, pp. 885–888, 2006, doi: 10.1109/ICME.2006.262661.

[8] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, “Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, 2017, doi: 10.1109/TASLP.2017.2690575.

[9] V. Morfi and Dan Stowell, “Deep learning for audio event detection and tagging on low-resource datasets,” *Appl. Sci.*, 2018, doi: 10.3390/app8081397.

[10] S. Sigtia, A. M. Stark, S. Krstulović, and M. D. Plumbley, “Automatic Environmental Sound Recognition: Performance Versus Computational Cost,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, 2016, doi: 10.1109/TASLP.2016.2592698.

[11] B. Uzkent, B. D. Barkana, and H. Cevikalp, “Non-speech environmental sound classification using SVMs with a new set of features,” *Int. J. Innov. Comput. Inf. Control*, 2012.

[12] J. Pons, O. Nieto, M. Prockup, E. Schmidt, A. Ehmann, and X. Serra, “End-to-end learning for music audio tagging at scale,” *Proc. 19th Int. Soc. Music Inf. Retr. Conf. ISMIR 2018*, no. Nips, pp. 637–644, 2018.

[13] Q. Kong, Y. Xu, W. Wang, and M. D. Plumbley, “A joint detection-classification model for audio tagging of weakly labelled data,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2017, doi: 10.1109/ICASSP.2017.7952234.

[14] E. Cakır and T. H. and T. Virtanen, “DOMESTIC AUDIO TAGGING WITH CONVOLUTIONAL NEURAL NETWORKS,” *2016 Int. Work. Detect. Classif. Acoust. Scenes Events*, 2016, doi: 10.1109/IWAENC.2016.7602891.

[15] W. Li, S. Li, X. Shao, and Z. Li, *Proceedings of the 6th Conference on Sound and Music Technology*. 2019.

[16] E. Fonseca *et al.*, “General-purpose Tagging of Freesound Audio with AudioSet Labels: Task Description, Dataset, and Baseline,” no. 688382, 2018.

[17] E. Fonseca *et al.*, “Freesound datasets: A platform for the creation of open audio datasets,” *Proc. 18th Int. Soc. Music Inf. Retr. Conf. ISMIR 2017*, pp. 486–493, 2017.

[18] A. Vaiserman and O. Lushchak, *Geroscience*. 2019.

[19] V. K. I. DIMITRIS G. MANOLAKIS, *Applied digital signal processing:theory and practice*. Cambridge

University Press, 2011.

[20] E. Babaee, N. B. Anuar, A. W. Abdul Wahab, S. Shamshirband, and A. T. Chronopoulos, "An Overview of Audio Event Detection Methods from Feature Extraction to Classification," *Appl. Artif. Intell.*, vol. 31, no. 9–10, pp. 661–714, 2017, doi: 10.1080/08839514.2018.1430469.

[21] E. Cakir, *Deep Neural Networks for Sound Event Detection*, vol. 12. 2019.

[22] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic sound event detection using multi label deep neural networks," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2015-Septe, no. July, 2015, doi: 10.1109/IJCNN.2015.7280624.

[23] R. Lee, Ed., *Software Engineering Research, Management and Applications*, vol. 578. Cham: Springer International Publishing, 2015.

[24] S. S. Stevens, J. Volkmann, and E. B. Newman, "A Scale for the Measurement of the Psychological Magnitude Pitch," *J. Acoust. Soc. Am.*, 1937, doi: 10.1121/1.1915893.

[25] S. S. Stevens and J. Volkmann, "The Relation of Pitch to Frequency: A Revised Scale," *Am. J. Psychol.*, 1940, doi: 10.2307/1417526.

[26] S. B. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*. 1980, doi: 10.1109/TASSP.1980.1163420.

[27] R. Serizel *et al.*, "Acoustic Features for Environmental Sound Analysis To cite this version : HAL Id : hal-01575619 Chapter 4 : Acoustic Features for Environmental Sound Analysis .," 2017.

[28] T. Virtanen, M. D. Plumbley, and D. Ellis, *Computational Analysis of Sound Scenes and Events*. .

[29] K. S. Rao and S. G. Koolagudi, *Linear Prediction Analysis of Speech*. 2013.

[30] B. Ayoub, K. Jamal, and Z. Arsalane, "Gammatone frequency cepstral coefficients for speaker identification over VoIP networks," *2016 Int. Conf. Inf. Technol. Organ. Dev. IT4OD 2016*, 2016, doi: 10.1109/IT4OD.2016.7479293.

[31] M. Todisco, H. Delgado, and N. Evans, "A new feature for automatic speaker verification anti-spoofing: Constant Q cepstral coefficients," *Odyssey 2016 Speak. Lang. Recognit. Work.*, pp. 283–290, 2016, doi: 10.21437/Odyssey.2016-41.

[32] H. Phan, L. Hertel, M. Maass, P. Koch, and A. Mertins, "CaR-FOREST: Joint Classification-Regression Decision Forests for Overlapping Audio Event Detection," no. 1, pp. 1–5, 2016.

[33] J. F. Gemmeke *et al.*, "AUDIO SET : AN ONTOLOGY AND HUMAN-LABELED DATASET FOR AUDIO EVENTS," 2016.

[34] P. Foster, S. Sigtia, S. Krstulovic, J. Barker, and M. D. Plumbley, "CHIME-HOME : A DATASET FOR SOUND SOURCE RECOGNITION IN A DOMESTIC ENVIRONMENT School of Electronic Engineering and Computer Science , Queen Mary University of London , UK Audio Analytic , Cambridge , UK Department of Computer Science , University of Sheff," pp. 4–8, 2015.

[35] B. L. Sturm, "An analysis of the GTZAN music genre dataset," in *MIRUM 2012 - Proceedings of the 2nd International ACM Workshop on Music Information Retrieval with User-Centered and Multimodal Strategies, Co-located with ACM Multimedia 2012*, 2012, doi: 10.1145/2390848.2390851.

[36] K. J. Piczak, "ESC: Dataset for environmental sound classification," in *MM 2015 - Proceedings of the 2015 ACM Multimedia Conference*, 2015, doi: 10.1145/2733373.2806390.

[37] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *European Signal Processing Conference*, 2016, doi: 10.1109/EUSIPCO.2016.7760424.

[38] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*, 2014, doi: 10.1145/2647868.2655045.

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

[40] L. Barker, "Pattern Classification," *Technometrics*, 2002, doi: 10.1198/tech.2002.s659.

[41] M. Zhang, Z. Zhou, and S. Member, "Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization," vol. 18, no. 10, pp. 1338–1351, 2006.

[42] S. T. Examiner, T. Virtanen, H. H. Examiner, and E. Engineering, “Emre Cakir Multilabel Sound Event Classification with Neural Networks,” no. September, 2014.

[43] Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,” *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997, doi: 10.1006/jcss.1997.1504.

[44] M. Kuhn and K. Johnson, *Applied predictive modeling*. 2013.

[45] A. Scenes, “ENSEMBLE OF CONVOLUTIONAL NEURAL NETWORKS FOR GENERAL PURPOSE School of Electrical Engineering Signals and Systems Department Belgrade , Serbia,” 2018.

[46] I.-Y. Jeong, H. Lim, and C. Ai, “AUDIO TAGGING SYSTEM FOR DCASE 2018: FOCUSING ON LABEL NOISE, DATA AUGMENTATION AND ITS EFFICIENT LEARNING Technical Report,” pp. 1–4, 2018.

[47] M. Dorfer and G. Widmer, “Training General-Purpose Audio Tagging Networks With Noisy Labels and Iterative Self-Verification,” *Dcase*, no. November, 2018.

[48] J. Demˇ, “Statistical Comparisons of Classifiers over Multiple Data Sets,” vol. 7, pp. 1–30, 2006.

[49] L. Rokach, “Taxonomy for characterizing ensemble methods in classification tasks : A review and annotated bibliography,” *Comput. Stat. Data Anal.*, vol. 53, no. 12, pp. 4046–4072, 2009, doi: 10.1016/j.csda.2009.07.017.

[50] S. Arlot and A. Celisse, “A survey of cross-validation procedures for model selection,” *Stat. Surv.*, vol. 4, pp. 40–79, 2010, doi: 10.1214/09-SS054.

[51] S. Edition, *An Introduction to Machine Learning | SpringerLink*. .

[52] M. Peitler, “Acoustic Event Detection of General Sounds,” 2016.

[53] P. Sciences and A. C. Pocock, “FEATURE SELECTION JOINT LIKELIHOOD,” 2012.

[54] Q. Chen, M. Zhang, and B. Xue, “Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression,” *IEEE Trans. Evol. Comput.*, vol. 21, no. 5, pp. 792–806, 2017, doi: 10.1109/TEVC.2017.2683489.

[55] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, “A survey of the recent architectures of deep convolutional neural networks,” *Artif. Intell. Rev.*, pp. 1–68, 2020, doi: 10.1007/s10462-020-09825-6.

[56] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, “Language Modeling with Gated Convolutional Networks,” 2017.

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

[58] K. Choi and M. Sandler, “Automatic tagging using deep convolutional neural networks.”

[59] A. Senior, “Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling Has.”

[60] J. Wang, R. Zhao, D. Wang, R. Yan, K. Mao, and F. Shen, “Machine health monitoring using local feature-based gated recurrent unit networks,” *IEEE Trans. Ind. Electron.*, vol. 65, no. 2, pp. 1539–1548, 2017, doi: 10.1109/TIE.2017.2733438.

[61] J. Hu, “Squeeze-and-Excitation Networks,” pp. 7132–7141.

[62] C. Koch and E. Niebur, “A Model of Saliency-based Visual Attention for Rapid Scene Analysis,” no. January 2013, 1998, doi: 10.1109/34.730558.

[63] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, “Attention and Localization based on a Deep Convolutional Recurrent Model for Weakly Supervised Audio Tagging.”

[64] “SCALABLE STACKING AND LEARNING FOR BUILDING DEEP ARCHITECTURES,” pp. 2133–2136, 2012.

[65] S. Adavanne, P. Pertila, and T. Virtanen, “Sound event detection using spatial features and convolutional recurrent neural network,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, no. March, pp. 771–775, 2017, doi: 10.1109/ICASSP.2017.7952260.

[66] T. Heittola, A. Eronen, and T. Virtanen, “Acoustic event detection in real-life recordings,” no. July, 2014.

[67] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, “Detection and Classification of Acoustic Scenes and Events,” *IEEE Trans. Multimed.*, vol. 17, no. 10, pp. 1733–1746, 2015, doi: 10.1109/TMM.2015.2428998.

[68] Y. Xu, Q. Huang, W. Wang, P. J. B. Jackson, and M. D. Plumbley, “Fully DNN-based Multi-label regression for audio tagging,” no. September, 2016.

[69] K. Li, J. Daniels, C. Liu, P. Herrero, and P. Georgiou, “Convolutional Recurrent Neural Networks for Glucose Prediction,” *IEEE J. Biomed. Heal. Informatics*, vol. 24, no. 2, pp. 603–613, 2020, doi: 10.1109/JBHI.2019.2908488.

[70] M. Monteleone, “NooJ local grammars and formal semantics: Past participles vs. adjectives in Italian,” *Commun. Comput. Inf. Sci.*, vol. 607, no. 8, pp. 83–95, 2016, doi: 10.1007/978-3-319-42471-2_8.

[71] G. Parascandolo, “RECURRENT NEURAL NETWORKS,” no. August, 2015.

[72] F. Ortega-Zamorano, J. M. Jerez, D. U. Munoz, R. M. Luque-Baena, and L. Franco, “Efficient Implementation of the Backpropagation Algorithm in FPGAs and Microcontrollers,” *IEEE Trans. Neural Networks Learn. Syst.*, 2016, doi: 10.1109/TNNLS.2015.2460991.

[73] “Deep Learning.” .

[74] R. Pascanu, D. Tour, T. Mikolov, and D. Tour, “On the difficulty of training recurrent neural networks,” no. 2.

[75] E. T. Virtanen and A. Diment, “Convolutional Neural Networks for Acoustic,” no. May, 2016.

[76] S. Ioffe and C. Szegedy, “Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift.”

[77] M. Lin, Q. Chen, and S. Yan, “Network In Network,” pp. 1–10.

[78] A. Sehgal and N. Kehtarnavaz, “A Convolutional Neural Network Smartphone App for Real-Time Voice Activity Detection,” *IEEE Access*, 2018, doi: 10.1109/ACCESS.2018.2800728.

[79] H. C. Vemula, “Multiple Drone Detection and Acoustic Scene Classification with Deep Learning,” 2018.

[80] J. Feng and S. Lu, “Performance Analysis of Various Activation Functions in Artificial Neural Networks,” *J. Phys. Conf. Ser.*, vol. 1237, no. 2, 2019, doi: 10.1088/1742-6596/1237/2/022030.

[81] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” *Insights Imaging*, vol. 9, no. 4, pp. 611–629, 2018, doi: 10.1007/s13244-018-0639-9.

[82] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[83] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, “Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection,” *2016 Int. Conf. Platf. Technol. Serv. PlatCon 2016 - Proc.*, 2016, doi: 10.1109/PlatCon.2016.7456805.

[84] T. Zia, A. Abbas, U. Habib, and M. S. Khan, “Learning deep hierarchical and temporal recurrent neural networks with residual learning,” *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 4, pp. 873–882, 2020, doi: 10.1007/s13042-020-01063-0.

[85] P. J. Werbos, “Backpropagation Through Time: What It Does and How to Do It,” *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990, doi: 10.1109/5.58337.

[86] I. Vasilev, D. Slater, G. Spacagna, P. Roelants, and V. Zocca, *Python Deep Learning: Exploring Deep Learning Techniques and Neural Network Architectures with PyTorch, Keras and TensorFlow*. 2018.

[87] F. A. Gers, J. Schmidhuber, F. Cummins, F. A. Gers, and F. Cummins, “(Edinburgh , Scotland), 1 Introduction 2 Standard LSTM The basic unit in the hidden layer of an,” vol. 2, pp. 850–855, 1999.

[88] M. Schuster and K. K. Paliwal, “Bidirectional Recurrent Neural Networks,” vol. 45, no. 11, pp. 2673–2681, 1997.

[89] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, “No Title,” pp. 2–6.

[90] J. Salamon and J. P. Bello, “Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification,” *IEEE Signal Process. Lett.*, vol. 24, no. 3, pp. 279–283, 2017, doi: 10.1109/LSP.2017.2657381.

[91] A. Scenes, “MULTI-LABEL AUDIO TAGGING SYSTEM FOR FREESOUND 2019 : FOCUSING ON NETWORK ARCHITECTURES , LABEL NOISY AND LOSS FUNCTIONS Technical Report Xiaofeng Hong , Gang Liu Beijing University of Posts and Telecommunications,” 2019.

[92] N. Zealand, “Stacked Generalization : when does it work ? Stacked Generalization : w h e n does it w o r k ?,” no. December, 2014.

[93] T. V. EmreÇakır, Sharath Adavanne, Giambattista Parascandolo, Konstantinos Drossos, “Convolutional Recurrent Neural Networks for Bird Audio Detection,” p. 5, doi: 10.23919/EUSIPCO.2017.8081508.

[94] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, “End-to-end attention-based large vocabulary speech recognition,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2016, doi: 10.1109/ICASSP.2016.7472618.

[95] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” in *Advances in Neural Information Processing Systems*, 2015.

[96] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, “Recurrent models of visual attention,” in *Advances in Neural Information Processing Systems*, 2014.

[97] T. Lidy and A. Schindler, “CQT-based Convolutional Neural Networks for Audio Scene Classification,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 2016.

[98] A. Mesaros *et al.*, “Detection and Classification of Acoustic Scenes and Events 2017,” *Dcase 2017*, no. November, 2017.

[99] M. Verleysen, “Classification in the Presence of Label Noise : a Survey,” no. May, 2014, doi: 10.1109/TNNLS.2013.2292894.

[100] F. Font, G. Roma, and X. Serra, “Freesound Technical Demo,” pp. 411–412.

[101] M. Lutz, *Learning Python: Powerful Object-Oriented Programming*. 2009.

[102] H. P. Langtangen *et al.*, *A Primer on Scientific Programming with Python Fifth Edition*. 2016.

[103] B. McFee *et al.*, “librosa: Audio and Music Signal Analysis in Python,” *Proc. 14th Python Sci. Conf.*, no. Scipy, pp. 18–24, 2015, doi: 10.25080/majora-7b98e3ed-003.

[104] S. Van der Walt and M. Aivazis, “The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering,” *Comput. Sci. Eng.*, 2011.

[105] W. McKinney, “pandas: a Foundational Python Library for Data Analysis and Statistics,” *Python High Perform. Sci. Comput.*, 2011.

[106] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[107] K. Xu *et al.*, “General audio tagging with ensembling convolutional neural networks and statistical features,” *J. Acoust. Soc. Am.*, vol. 145, no. 6, pp. EL521–EL527, 2019, doi: 10.1121/1.5111059.

[108] M. Sugiyama, *Introduction to Statistical Machine Learning*. 2015.

[109] F. Ykhlef, S. A. Hamada, F. Ykhlef, A. Derbal, and D. Bouchaffra, “Real-time detection of impulsive sounds for audio surveillance systems,” *CEUR Workshop Proc.*, vol. 2351, pp. 1–10, 2019.

[110] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “MixUp: Beyond empirical risk minimization,” in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

[111] D. Liang, F. Yang, T. Zhang, and P. Yang, “Understanding mixup training methods,” *IEEE Access*, 2018, doi: 10.1109/ACCESS.2018.2872698.

[112] C. G. Cordero, S. Hauke, M. Muhlhauser, and M. Fischer, “Analyzing flow-based anomaly intrusion detection using Replicator Neural Networks,” *2016 14th Annu. Conf. Privacy, Secur. Trust. PST 2016*, pp. 317–324, 2016, doi: 10.1109/PST.2016.7906980.

[113] J. Read, B. Pfahringer, and G. Holmes, “Multi-label classification using ensembles of pruned sets,” *Proc. - IEEE Int. Conf. Data Mining, ICDM*, pp. 995–1000, 2008, doi: 10.1109/ICDM.2008.74.

[114] Y. Bian and H. Chen, “When does Diversity Help Generalization in Classification Ensembles?,” pp. 1–12, 2019.

[115] J. Zhou, H. Peng, and C. Y. Suen, “Data-driven decomposition for multi-class classification,” vol. 41, pp. 67–76, 2008, doi: 10.1016/j.patcog.2007.05.020.

[116] J. Zhou and C. Y. Suen, “Unconstrained Numeral Pair Recognition Using Enhanced Error Correcting Output Coding : A Holistic Approach,” 2005.

[117] C. Qian, Y. Yu, and Z. Zhou, “Pareto Ensemble Pruning ∗,” pp. 2935–2941, 2012.

[118] A. K. Seewald and J. Fürnkranz, “An evaluation of grading classifiers,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2001, doi: 10.1007/3-540-44816-0_12.

[119] N. M. Wanas, R. A. Dara, and M. S. Kamel, “Adaptive fusion and co-operative training for classifier ensembles,” *Pattern Recognit.*, 2006, doi: 10.1016/j.patcog.2006.02.003.

[120] G. Tsoumakas, I. Partalas, and I. Vlahavas, “An ensemble pruning primer,” in *Studies in Computational Intelligence*, 2009, doi: 10.1007/978-3-642-03999-7_1.

[121] E. Arazo, D. Ortego, P. Albert, N. E. O’Connor, and K. McGuinness, “Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning,” 2019.