Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Curriculum in Ingegneria Elettronica, Elettrotecnica e delle Telecomunicazioni

Deep Learning for Sound Event Detection and Classification

Ph.D. Dissertation of:
Fabio Vesperini

Advisor:
Prof. Stefano Squartini

Curriculum Supervisor:
Prof. Francesco Piazza

XVII edition - new series

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Curriculum in Ingegneria Elettronica, Elettrotecnica e delle Telecomunicazioni

Deep Learning for Sound Event Detection and Classification

Ph.D. Dissertation of:
Fabio Vesperini

Advisor:
Prof. Stefano Squartini

Curriculum Supervisor:
Prof. Francesco Piazza

XVII edition - new series

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Facoltà di Ingegneria
Via Brecce Bianche – 60131 Ancona (AN), Italy

alla mia famiglia

Acknowledgments

Foremost, I would like to express my sincere gratitude to my supervisor since the time of the Bachelor thesis, Prof. Stefano Squartini, for his support of my studies and research, for his patience, motivation, and immense knowledge. His guidance has helped me in a personal growth process and I will always be grateful to him for everything.

Together with my advisor, I would like to thank the rest of my colleagues of the A3Lab: Emanuele, for his unfailing experience, Marco, who has been like an older brother, Roberto and Leonardo, for the sharing of ideas and opinions on all kinds of topics, Paul, for his mood without compromises and also Stefania, Daniele, Livio, Stefano, Marco G., Christian, Michele and Alessandro. I will have definitely a lovely memory of these years, and the merit is yours too.

A special thanks also goes to Diego, a friend, a roommate and a colleague of stimulating discussions, sleepless nights and of old and new adventures. To him and Chiarella I wish all the best.

I also want to thank the people that were close to me, beside my work, but that were just as indispensable. In particular my childhood friend and acquired brother Michele (Gino), my bandmates from the "Plebos", the "VHS" and the "BlueNight", and my music masters since all times. You really rock, guys!

And finally, last but by no means least, a really big thank to my whole family, my parents Filippo and Anna and my super brothers Marco and Christian. You are and will be my source of encouragement and my spiritual support throughout my life.

Ancona, Ottobre 2018

Fabio Vesperini

Abstract

The recent progress on acoustic signal processing and machine learning techniques have enabled the development of innovative technologies for automatic analysis of sound events. In particular, nowadays one of the hottest approach to this problem lays on the exploitation of Deep Learning techniques. As further proof, in several occasion neural architectures originally designed for other multimedia domains have been successfully proposed to process the audio signal. Indeed, although these technologies have been faced for a long time by statistical modelling algorithms such as Gaussian Mixture Models, Hidden Markov Models or Support Vector Machines, the new breakthrough of machine learning for audio processing has lead to encouraging results into the addressed tasks. Hence, this thesis reports an up-to-date state of the art and proposes several reliable DNN-based methods for Sound Event Detection (SED) and Sound Event Classification (SEC), with an overview of the Deep Neural Network (DNN) architectures used on purpose and of the evaluation procedures and metrics used in this research field.

According to the recent trend, which shows an extensive employment of Convolutional Neural Networks (CNNs) for both SED and SEC tasks, this work reports also rather new approaches based on the Siamese DNN architecture or the novel Capsule computational units. Most of the reported systems have been designed in the occasion of international challenges. This allowed the access to public datasets, and to compare systems proposed by the most competitive research teams on a common basis.

The case studies reported in this dissertation refer to applications in a variety of scenarios, ranging from unobtrusive health monitoring, audio-based surveillance, bio-acoustic monitoring and classification of the road surface conditions. These tasks face numerous challenges, particularly related to their application in real-life environments. Among these issues there are unbalancing of datasets, different acquisition setups, acoustic disturbance (i.e., background noise, reverberation and cross-talk) and polyphony. In particular, since multiple events are very likely to overlap in real life audio, two algorithms for polyphonic SED are reported in this thesis. A polyphonic SED algorithm can be considered as system which is able to perform contemporary detection - determining onset and offset time of the sound events - and classification - assigning a label to each of the events occurring in the audio stream.

Contents

1 Introduction **1**
1.1 Sound Analysis Tasks . . . 2
1.1.1 Applications . . . 3
1.2 The Deep Learning Approach . . . 4
1.3 State of the Art . . . 5
1.4 Main Issues . . . 7
1.5 Case studies . . . 8

2 Background **9**
2.1 The Artificial Neural Networks . . . 9
2.1.1 The Human Nervous System . . . 10
2.2 Historical Background . . . 12
2.2.1 Fundamentals of the Artificial Neural Networks . . . 13
2.3 Deep Neural Network Architectures for Analysis of Sound Events 16
2.3.1 Multi Layer Perceptron (MLP) . . . 16
2.3.2 Convolutional Neural Networks (CNN) . . . 17
2.3.3 Siamese Neural Networks . . . 19
2.3.4 Generative Adversarial Networks (GAN) . . . 20
2.3.5 Recurrent Neural Networks (RNN) . . . 21
2.3.6 Capsule Neural Networks (CapsNet) . . . 24
2.4 Optimization Algorithms . . . 26
2.4.1 Stochastic Gradient Descent (SGD) . . . 28
2.5 Generalization Techniques . . . 28
2.5.1 Dropout . . . 28
2.5.2 Batch Normalization . . . 29

3 Datasets and Evaluation **31**
3.1 Datasets . . . 31
3.1.1 Dataset Acquisition . . . 32
3.1.2 Dataset Labelling . . . 33
3.1.3 Acoustic Features . . . 33
3.1.4 Data Augmentation . . . 35
3.2 Evaluation Setup . . . 37
3.3 Evaluation Metrics . . . 38
3.3.1 Metrics Computation . . . 38

3.3.2 Detection Metrics 40
3.3.3 Final Remarks 41

4 Sound Event Classification **43**
4.1 Snore Sounds Excitation Localization 44
4.1.1 Proposed Approach 46
4.1.2 Comparative Approaches 51
4.1.3 Experiments 52
4.1.4 Results 55
4.1.5 Conclusion and Outlook 58
4.2 Road Surface Roughness Classification 59
4.2.1 Proposed Method 61
4.2.2 Dataset 63
4.2.3 Experiments 66
4.2.4 Conclusions 68
4.3 Bird Audio Detection 70
4.3.1 Related Works 70
4.3.2 Proposed Method 71
4.3.3 Experimental Setup 72
4.3.4 Results 75
4.3.5 Conclusion and Outlook 76

5 Sound Event Detection **77**
5.1 Overnight Snore Sounds Detection 78
5.1.1 Proposed Approach 78
5.1.2 A3 Snore Dataset 80
5.1.3 Data Augmentation Techniques 81
5.1.4 Experimental Setup 83
5.1.5 Results 83
5.1.6 Conclusion 84
5.2 Rare Sound Event Detection 86
5.2.1 Related Works 86
5.2.2 Proposed Method 87
5.2.3 Experimental Setup 90
5.2.4 Results 93
5.2.5 Conclusion 94
5.3 CNN with 3-D Kernels for VAD in a Multiroom Environment . 95
5.3.1 Proposed Approach 96
5.3.2 DIRHA Dataset 98
5.3.3 Experiments 98
5.3.4 Results 99
5.3.5 Conclusion 101

6 Polyphonic Sound Event Detection **103**
6.0.1 State-of-the-art Overview . . . 103
6.1 Sound Event Detection in Real Life Audio - DCASE 2016 . . . 104
6.1.1 Proposed method . . . 104
6.1.2 Experiments . . . 107
6.1.3 Results . . . 109
6.1.4 Conclusion . . . 111
6.2 Polyphonic SED by using CapsNets . . . 113
6.2.1 Related Works . . . 113
6.2.2 Proposed Method . . . 114
6.2.3 Experimental Setup . . . 117
6.2.4 Comparative Algorithms . . . 118
6.2.5 Neural Network configuration . . . 119
6.2.6 Results . . . 121
6.2.7 TUT-SED 2016 . . . 121
6.2.8 TUT-SED 2017 . . . 124
6.2.9 TUT-Rare SED 2017 . . . 125
6.2.10 Alternative Dynamic Routing for SED . . . 127
6.2.11 Conclusion . . . 127

7 Other contributions **129**
7.1 Acoustic Novelty Detection with Adversarial Autoencoders . . . 129
7.1.1 Details . . . 130
7.2 Siamese Nets for Human-Fall Detection . . . 132
7.2.1 Proposed Approach . . . 132
7.2.2 Results . . . 133
7.3 Generative Raw Audio Synthesis . . . 134
7.3.1 Algorithm Selection . . . 134
7.3.2 Input Material and Dataset Creation . . . 136
7.3.3 Results . . . 137

8 Conclusions **139**
8.1 Future Perspective . . . 140
8.2 A Zoom Out . . . 141

List of Publications **143**

List of Figures

1.1 System input and output 2
1.2 Basic structure 4

2.1 Human Brain 10
2.2 Neuron Model 11
2.3 Artificial Neuron Model 13
2.4 The *tanh* 14
2.5 The *ReLU* 15
2.6 MLP 16
2.7 CNN 18
2.8 Convolution and Pooling 18
2.9 Siamese DNN 20
2.10 Generative Adversarial Networks 21
2.11 LSTM 22
2.12 GRU 24
2.13 Example of Gradient Descent 27

3.1 Dummy head 33
3.2 Cross-Validation Splitting 37
3.3 P-R Curve 41

4.1 System input and output for SEC 43
4.2 VOTE Locations 44
4.3 VOTE Localization Algorithm 47
4.4 The Morlet Wavelet Filterbanks 49
4.5 SCAT Extraction Tree 49
4.6 Spectrograms of VOTE Sounds 54
4.7 VOTE Classification - Devel set results 56
4.8 VOTE Classification - Test set results 57
4.9 Tyre-Road Noises 59
4.10 Third-Octave Band Noise Spectra 60
4.11 Road Surface Roughness Classification 61
4.12 Siamese Architecture 62
4.13 Car Equipment for Dataset Recording 64
4.14 Microphones Positioning 65

List of Figures

4.15 HEAD Acoustics SQuadriga II 65
4.16 Road Noise Spectrograms 66
4.17 Road Samples 66
4.18 Siamese Nets Output 68
4.19 CapsNet for Bird Audio Detection 73

5.1 System input and output for SED 77
5.2 Snoring Detection with CRNNs 78
5.3 Recording Room 81
5.4 Snoring Detection - Results 84
5.5 Proposed Approach for Rare Sound Event Detection 88
5.6 Multiroom VAD with 3-D CNNs 96
5.7 CNNs with 3D kernels 97
5.8 DIRHA Apartment 98
5.9 Multiroom VAD - Results 101

6.1 System input and output for Poly-SED 104
6.2 Sound Event Detection - DCASE 2016 105
6.3 Polyphonic SED with CapsNets 116
6.4 Polyphonic SED - Network outputs I 123
6.5 Polyphonic SED - Network outputs II 125

7.1 Acoustic Novelty Detection 129
7.2 Acoustic Novelty Detection with GANs 131
7.3 Siamese Nets for Human-Fall Detection 133
7.4 Siamese Nets for Human-Fall Detection - Results 133
7.5 SampleRNN - Raw Audio Generation 138

List of Tables

4.1 The Munich-Passau Snore Sound Corpus 53
4.2 VOTE Classification - Experiments 54
4.3 VOTE Classification - Results 55
4.4 VOTE Classification - Comparative Results 58
4.5 Road Surface Roughness Classification - Experiments 67
4.6 Road Surface Roughness Classification - Results 67
4.7 Bird Audio Detection - DCASE 2018 dataset 74
4.8 CapsNet for Bird Audio Detection - Experiments 75
4.9 CapsNet for Bird Audio Detection - Results 76

5.1 A3-SNORE dataset 81
5.2 Snoring Detection - Experiments 83
5.3 Rare Sound Event Detection - ED Stage 1 Experiments 91
5.4 Rare Sound Event Detection - Results 92
5.5 Rare Sound Event Detection - ED Stage 1 Best models 93
5.6 Rare Sound Event Detection - Comparative Results 94
5.7 Multiroom VAD - Network Layouts 99

6.1 Classes of DCASE 2016 task 3 dataset 108
6.2 Sound Event Detection - DCASE 2016 - Experiments 110
6.3 Sound Event Detection - DCASE 2016 - Results 111
6.4 Sound Event Detection - DCASE 2016 - Best Results 111
6.5 Sound Event Detection - DCASE 2016 - Comparative Results . 112
6.6 Polyphonic SED with CapsNets - Random Search 120
6.7 Polyphonic SED with CapsNets - Best models I 122
6.8 Polyphonic SED with CapsNets - Results 122
6.9 Polyphonic SED with CapsNets - Best models II 126
6.10 Polyphonic SED with CapsNets - Results II 126
6.11 Polyphonic SED with CapsNets - Alternative routing 127

Chapter 1

Introduction

Human cognition relies on the ability to sense, process, and understand the surrounding environment and its sounds. Although the skill of listening and understanding their origin is so natural for living beings, it still results in a very challenging task for computers. In recent years several novel methods have been proposed to analyze this information automatically, and several new applications have emerged [1]. However, the creation of "machine listening" algorithms that can mimic this cognitive feature by means of artificial systems remains a very challenging task.

Systems for automatic acoustic event recognition have the aim to mimic this cognitive feature. Basically, these algorithms are designed to analyze a continuous audio signal in order to extract a description of the sound events occurring in the stream. This description is commonly expressed as a label that marks the start, the ending, and the nature of the occurred sound (*e.g.,* children crying, cutlery, glass jingling).

Thanks to works like Bregman's "Auditory Scene Analysis: The Perceptual Organization of Sound" [2], we can trace back the birth of this research topic to 1994, when the field of computational auditory scene analysis (CASA) was introduced in order to model humans' sound perception. Following this work, many other contributions were written aiming to describe how artificial systems can be designed in order to perceive sounds similarly to as humans do; most of these works will be later collected in Divenyi's book [3] in 2004.

Sound events often occur in unstructured environments in real-life. Factors such as background noise and overlapping sources are commonly present in the environments. Moreover, there can be multiple sound sources that produce sound events belonging to the same class, e.g., a dog bark sound event can be produced from several breeds of dogs with different acoustic characteristics. These represent some of the main challenges which the systems described in this thesis have to face in order to prove their effectiveness in real-life situations.

1.1 Sound Analysis Tasks

The type of information to be extracted with acoustic event analysis algorithms depends on the application. In particular, we can classify the tasks explored in this dissertation into two high-level categories: sound event *detection* and sound event *classification*. In Sound Event *Detection* (SED), or acoustic event detection, the goal is to detect the onset and offset times for a variety of sound events captured in an audio recording. In Sound Event Classification (SEC), the goal is to categorize an audio recording into one of a set of predefined categories by associating a textual descriptor. The different input and output respectively for classification and detection systems are illustrated in Figure 1.1.

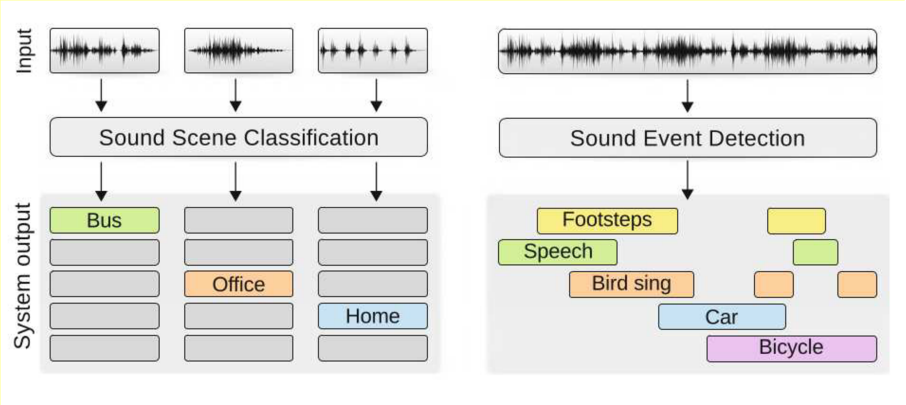


Figure 1.1: System input and output for the two main analysis systems: sound scene classification and sound event detection. Pic. courtesy of [1].

Labels extracted with a sound recognition system allow to achieve a better insight of the considered acoustic scenario. Usually, they are used as mid-level representation useful for other CASA research areas. In [4, 5], for example, authors make use of SED for designing audio context recognition systems, while in [6] and [7] SED is exploited for automatic tagging and audio segmentation respectively. Moreover, both SED and SEC found many direct applications in a variety of scenarios, some examples being context-based indexing and retrieval in multimedia databases [8], unobtrusive health monitoring [9], and audio-based surveillance [10, 11, 12]. When more than one event can be active (and should be detected) at a time, therefore foreseeing the overlapping of two or more of these labels, we can refer to it as polyphonic-SED. This problem is addressable as a "mixture problem" and it is usually not trivial to solve mainly due to the superimposition of different event energies in the audio spectra.

1.1.1 Applications

Acoustic surveillance can be considered one of the most interesting subjects in technological research. Surveillance can be seen as control of public safety or as the supervision of private environments where people may live alone. The increasing level of public security over the past decades has motivated the installation of sensors such as cameras or microphones in public places (stores, subway, airports, etc.), while is possible to effectively consider personal multimedia devices (smartphones, tablets, etc.) as virtual assistant which is able to monitor the user and eventually intervene in case of necessity without having the need for a physical interface (i.e., keyboard) anymore. Thus, the need of unsupervised situation assessment stimulated the signal processing community towards experimenting with several automated frameworks, due to their potential in several engineering applications.

In these contexts, sound or sound sensing can be advantageous with respect to other modalities of multimedia processing, due to the short duration of certain events (i.e., a human fall, a gunshot or a glass breaking) or the personal privacy motivate the exploitation of the audio information rather than, e.g., the image processing. Reports suggests that 90% of physical aggression is preceded by verbal aggression [13]. In public spaces, or in specific environments like prisons and detention facilities, a tool able to recognize vocal hostility and enable security guards to intervene can prevent further escalation and save lives and public goods.

Hence, being the research on automatic-assisted home environments an active area for study in the recent years, a particular attention has been paid to the processing of audio signals [14, 15, 16]. Typically, to increase the quality of the audio signal and improve the performance of the successive audio analysis stages in complex systems, pre-processing algorithms are employed [17, 18, 19]. To confirm this, several works appeared recently in the literature that address speech interaction in multi-room scenarios. For example, in [20] the authors developed a multi-room spoken command recognizer, while in [21, 22, 23] voice activity detectors able to concurrently identify the location in time and the room of origin of speech segments have been proposed.

In addition, audio processing is often less computationally demanding compared to other multimedia domains, thus embedded devices can be easily equipped with microphones and sufficient computational capacity to locally process the signal captured. These could be smart home devices for home automation purposes or sensors for wildlife and biodiversity monitoring (i.e., bird calls detection [24]).

Some of these applications have already become commercial products that are able recognize certain specific sound categories in realistic environments and improve home security [25] or companies with as much impactful missions,

such as preserve the Rainforest from illegal deforestation [26].

1.2 The Deep Learning Approach for Sound Event Detection and Classification

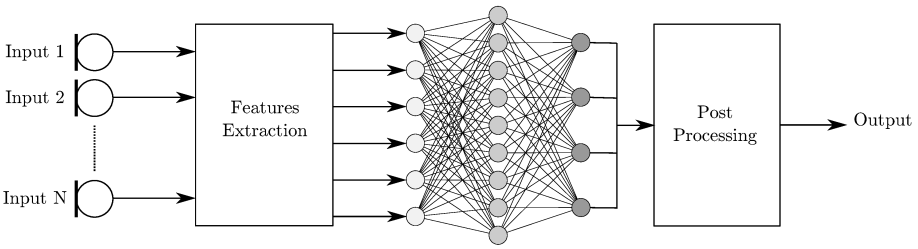


Figure 1.2: Basic structure of an audio analysis system.

The typical computational sound scene or event analysis system based on machine learning is depicted in Figure 1.2.

As the first stage, all the systems take as input one or more audio signals, either in real-time, captured by a microphone, or offline, from an audio recording. In this dissertation, we always assume discrete-time signals, obtained by using analog-to-digital converters.

The *Feature Extraction* block consists of different processing stages and outputs acoustic features, as the actual analysis of audio is rarely based on the raw audio signal, but rather on the compact signal representation with features. The purpose of the feature extraction is to obtain information sufficient for detecting or classifying the target sounds, making the subsequent modeling stage computationally cheaper and also easier to achieve with limited amount of development material. Very often the feature extraction procedure is also preceded by the down-mixing the audio signal into a single (mono) channel and re-sampling it into fixed sampling frequency. Although every application could require a specific set of features able to highlight the discriminating particularities of each data sample, the most common representations used for audio signals are non-linear representation for magnitudes (power spectra and logarithm) and nonlinear frequency scaling (requency warping according to the mel scale). More details of the acoustic features extraction process for each examined case-study will be provided in further chapters.

The *Deep Learning*-based model takes the acoustic features as input and it is trained to produce an output which will assign a class label depending on the application. Almost all the system presented in this thesis are based on the supervised machine learning approach, where the system is trained using labeled examples of sounds from each of target sound type. At the development stage,

the obtained acoustic features are used together with reference annotations of the audio training examples, to learn models for the sound classes of interest. Annotations contain information about the presence of target sound classes in the training data, and are used as a reference information to automatically learn a mapping between acoustic features and class labels. The mapping is represented by acoustic models. The learning process consists in updating the parameters or *weights* of the neural network, searching for the optimal model that minimize a certain cost-function. At the usage stage, the learned acoustic models are used to do recognition (detection or classification), which predicts labels for the input audio. The recognition stage may also involve temporal models and post-processing of labels.

After a prediction is obtained through the trained acoustic model, the *Post Processing* stage translates this signal into the effective activity information for each class. Very often this relies on a simple *thresholding* operation or on the selection of the most probable class.

1.3 State of the Art

As aforementioned, research on automatic a classification of real-world sounds grew in the middle 1990s. One of the earliest systems [27] provided similarity-based access to databases of isolated sound effects by representing each clip by a fixed-size feature vector comprising perceptual features such as loudness, pitch, and brightness. Similarly, techniques used to recognize the human voice exploited algorithms based on the thresholding of typical characteristics of the acoustic wave, such energy, pitch and zero-crossing rate [28].

Later, more complex computational acoustic event analysis has been approached with statistical modelling methods, including Hidden Markov Models (HMM) [29], Gaussian Mixture Models (GMM) [5] or techniques Non-negative Matrix Factorization (NMF) [30] and support vector machines (SVM) [31].

In the recent era of the "Deep Learning", different neural network architectures have been successfully used for sound event detection and classification tasks, including feed-forward neural networks (FNN) [32], deep belief networks [33], convolutional neural networks (CNNs) [34] and Recurrent Neural Networks (RNNs) [35]. In addition, these architectures laid the foundation for end-to-end systems [36, 37], in which the feature representation of the audio input is automatically learnt from the raw audio signal waveforms. An interesting comparison between computational costs of different systems is carried out in [38] highlighting that deep neural networks (DNNs) are able to achieve top performance at the cost of being the most computationally expensive approach. A brilliant example of such performance is given in [39], where different DNNs are trained on a big video dataset and then used for different scopes, among

which also SED. For a wider overview of the most recent and powerful SED techniques the reader can refer to the comprehensive analysis carried out by Sharan *et al.* in [40]. In [41] an LSTM based Voice Activity Detector (VAD) using RASTA-PLP features outperforms three different VAD algorithms applied to speech recognition of Hollywood-movies audio. Another well-fitting example is given in [42], where auto-encoders based on architectures comprehending MLP, RNN and bidirectional-RNN are trained on three datasets recorded in real life environments in order to detect abnormal events or hazardous situations exploiting only the information carried by the acoustic signal. Specifically, an autoencoder is a neural network trained to set the target values equal to its input. The experimental results show that autoencoders based on deep RNNs outperform the probabilistic approaches over the three databases.

The use of deep learning models has been motivated by the increased availability of datasets and computational resources and resulted in significant performance improvements, outperforming in most of the cases the human accuracy [43]. The methods based on CNNs and RNNs have established the new state-of-the-art performance on the SED, thanks to the capabilities to learn the non-linear relationship between time-frequency features of the audio signal and a target vector representing sound events. In [44], the authors show how "local" patterns can be learned by a CNN and can be exploited to improve the performance of detection and classification of non-speech acoustic events occurring in conversation scenes, in particular compared to a FNN-based system which processes multiple resolution spectrograms in parallel.

This success is a result of close academic-industrial collaboration, which started from the speech or speaker recognition task and extended to the analysis of non-speech, music and sound scenes and events. The combination of the CNN structure with recurrent units has increased the detection performance by taking advantage of the characteristics of each architecture. This is the case of convolutional recurrent neural networks (CRNNs) [45], which provided state-of-the-art performance especially in the case of polyphonic SED. CRNNs consolidate the CNN property of local shift invariance with the capability to model short and long term temporal dependencies provided by the RNN layers. This architecture has been also employed in almost all of the most performing algorithms proposed in the last editions of research challenges such as the IEEE Audio and Acoustic Signal Processing (AASP) Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE) [46]. In detail, both the first two classified algorithms for the SED task at the DCASE-2017 make use of mel spectrogram coefficients as spectral representation of the audio signal which is processed by a CNN with 1D filters in the case of the first ranked [47] or by a 2D CNN with frequency pooling in the case of the second classified [48]. The architectures are, then, combined with recurrent layers to

process the features obtained by the convolutional blocks. In [49] the authors propose a hierarchical structure based on CNNs and DNNs trained with multi-task loss functions. Specifically, in the first stage the networks are trained for background noise rejection, using a weighted loss function to penalize the false positive errors. In the second stage the multi-task loss enables the networks to simultaneously perform the event classification task and the onset time estimation. This approach obtained the third place in the final ranking. All of the aforementioned systems largely outperform the baseline system based on a Multi Layer Perceptron architecture (MLP) and Logmel energies as features.

On the other hand, if the datasets are not sufficiently large, problems such as overfitting can be encountered with these models, which typically are composed of a considerable number of free-parameters (i.e., more than 1M).

Encouraging polyphonic SED performance have been obtained using CapsNets in preliminary experiments conducted on the Bird Audio Detection task in occasion of the DCASE 2018 challenge [50], confirmed by the results reported in [51]. The CapsNet [52] is a recently proposed architecture for image classification and it is based on the grouping of activation units into novel structures introduced in [53], named *capsules*, along with a procedure called dynamic routing. The capsule has been designed to represent a set of properties for an entity of interest, while dynamic routing is included to allow the network to implicitly learn global coherence and to identify part-whole relationships between capsules.

1.4 Main Issues

In controlled laboratory conditions where the data used to develop computational sound scene and event analysis methods matches well with the test data, it is possible to achieve relatively high accuracies in the detection and classification of sounds. However, there are several complexities in computational sound analysis and current technologies face many challenges, mainly related to the acoustics of sound scenes and events, when they are employed in realistic environments. Among these challenges we can include:

- the effect of the environment acoustics: reverberation, background noises and the channel coupling (impulse response) between the source and the recording equipment;
- the intra-class variability, i.e., high difference of the acoustic characteristics of even a single class of sounds and on the other hand the similarity of many different types of sounds to the target events [54];
- the *polyphony*, i.e. the occurrence of multiple simultaneous events. In

realistic environments there are almost always multiple sources producing sound at the same time.

In addition to these complications related to the acoustics of sound scenes and events, there are also several fundamental limitations related to the computational methods. In particular, to develop effective models based on the *deep learning* paradigm, a very large set of examples of the target (and non-target) sounds is required. In contrast to the situation in image classification, currently available datasets that can be used to train such systems are more limited in size, diversity, and number of event instances, even though recent contributions such as AudioSet [55] or the DCASE challenges and related workshops [56, 46, 57] have provided public available datasets to reduce this gap.

1.5 Case studies

In this thesis, different application of deep learning for computational audio models in real environments are analyzed. They are evaluated and compared with state-of-the-art methods on different databases, some of these resulting novel approaches. The broad and extensive experimental evaluations highlight the advantages provided by the acoustic models based on deep learning.

The addressed tasks are the following:

- Sound event *Classification*:
  - Snore sounds excitation localization;
  - Acoustic road surface roughness classification;
  - Bird audio detection;
- Sound event *Detection*:
  - Overnight snore sound detection;
  - Rare sound event detection;
  - Voice activity detection in multiroom environments;
- *Polyphonic* Sound event Detection:
  - A neural network approach for sound event detection in real life audio;
  - Polyphonic sound event detection by using CapsNets

Chapter 2

Background

The Computers are able to perform complex calculus operations in a short amount of time. However computers cannot compete with humans in dealing with: common sense, ability to recognize people, objects, sounds, comprehension of natural language, ability to learn, categorize, generalize.

Therefore, why does the human brain show to be superior w.r.t common computers for these kind of problems? Is there any chance to mimic the mechanisms characterizing the way of working of our brain in order to produce more efficient machines?

In the field of signal analysis, the aim is the characterization of such real-world signals in terms of *signal models*, which can provide the basis for a theoretical description of a signal processing system. They are potentially capable of letting us learn a great deal about the signal source, without having to have the source available.

The "Deep Learning" is a new area of machine learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence. Deep Learning is about learning multiple levels representation and abstraction that help to make sense of data such as images, sound, and text.

Therefore, in this chapter a theoretical description of the principal Deep Neural Network (DNN) architectures is given. In addition, the algorithms used for their parameter estimation are described, with a focus on the most widely model structure used in the field of the computational acoustic event analysis, with a particular focus on the supervised machine learning approach, which is the mainstream and typically the most efficient and generic approach in developing such systems.

2.1 The Artificial Neural Networks

The human brain is composed of a big set of specialized cells (*neurons*) connected among them, which memorize and process information, thus controlling the body activities they belong to as depicted in Figure 2.1. The human brain

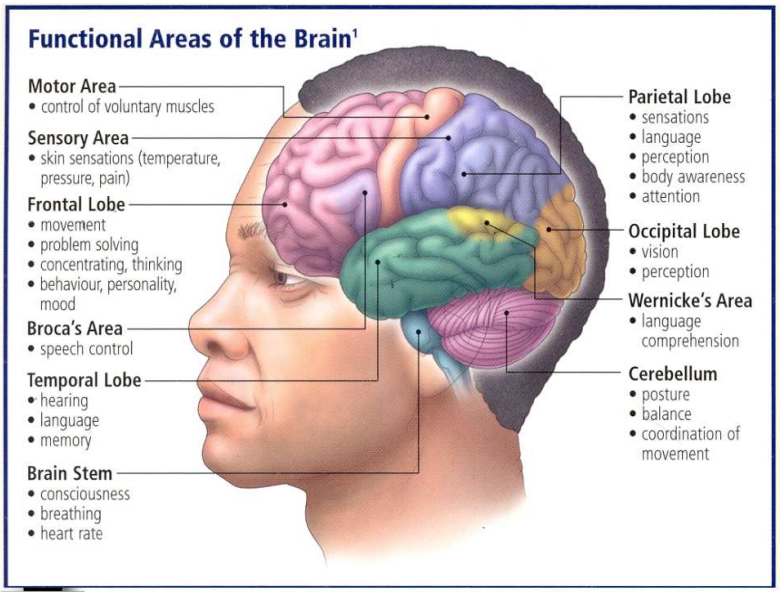


Figure 2.1: The human brain.

is probably the most remarkable result of evolution for its ability to elaborate information. The Artificial Neural Networks are mathematical models that represent the interconnection between elements defined "Artificial Neurons", mathematical constructs that somehow imitate the properties of biological neurons, going to reproduce the functioning of the human nervous system.

2.1.1 The Human Nervous System

The biological neuron is composed of three main parts: the cell body is named *Soma*, which is the calculation unit, the *Axon*, that acts as a transmission line output and the *Dendrites*, that are receptive areas and that receive input signals from other axons via the synapses. The *Synapses* are the functional units of the elementary structure, that manages the iterations between neurons. The cell body performs a weighted sum (integration) of the input signals. If the result exceeds a certain threshold value then the neuron is active and is produced the *action potential* which is transported to the axon, instead if the result does not exceed the threshold value of the neuron remains in a state of rest. The Biological neurons (Figure 2.2) are electro-chemical devices, operating at low rates (approximately in the order of milliseconds), while digital circuits operate at very high rates (nanoseconds).

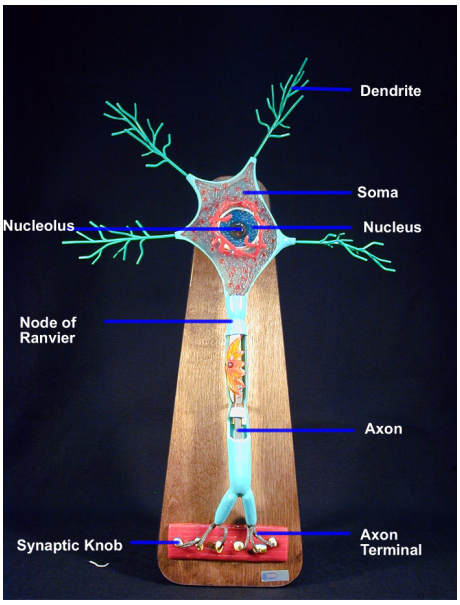


Figure 2.2: The neuron model.

The *neuron* properties can be described in:

- *local simplicity*: the neuron receives stimuli (excitation or inhibition) from dendrites and produces an impulse to the axon which is proportional to the weighted sum of the inputs;

- *global complexity*: the human brain possess $\mathcal{O}(10^{10})$ neurons, with more than 10k connections each;

- *learning*: even though the network topology is relatively fixed, the strength of connections (synaptic weights) can change when the network is exposed to external stimuli;

- *distributed control*: no centralized control, each neuron reacts only to its own stimuli;

- *tolerance to failures*: performance slowly decrease with the increase of failures.

The biological Neural Networks are able to solve very complex tasks in few time instants (like memorization, recognition, association, and so on.)

2.2 Historical Background

In 1943, McCulloch and Pitts created a computational model for neural networks based on mathematics and algorithms. They called this model threshold logic. The model paved the way for neural network research to split into two distinct approaches. One approach focused on biological processes in the brain and the other focused on the application of neural networks to artificial intelligence. In the late 1940s, the psychologist Donald Hebb created a hypothesis of learning based on the mechanism of neural plasticity that is now known as Hebbian learning. Hebbian learning is considered to be a 'typical' unsupervised learning rule and its later variants were early models for long term potentiation. These ideas started being applied to computational models in 1948 with Turing's B-type machines. In 1954, Farley and Clark first used computational machines, then called calculators, to simulate a Hebbian network at MIT. Other neural network computational machines were created by Rochester, Holland, Habit, and Duda (1956).

In 1958, Rosenblatt created the perceptron, an algorithm for pattern recognition based on a two-layer learning computer network using simple addition and subtraction. With mathematical notation, Rosenblatt also described circuitry not in the basic perceptron, such as the exclusive-or circuit, a circuit whose mathematical computation could not be processed until after the backpropagation algorithm was created by Werbos (1975). Neural network research stagnated after the publication of machine learning research by Minsky and Papert in 1969. They discovered two key issues with the computational machines that processed neural networks. The first issue was that single-layer neural networks were incapable of processing the exclusive-or circuit. The second significant issue was that computers were not sophisticated enough to effectively handle the long run time required by large neural networks. Neural network research slowed until computers achieved greater processing power. Also key in later advances was the backpropagation algorithm presented by Werbos in 1975, which effectively solved the exclusive-or problem.

The parallel distributed processing of the mid-1980s became popular under the name connectionism. The paper presented by Rumelhart and McClelland [58] provided a full exposition on the use of connectionism in computers to simulate neural processes. In 1994, Y. Bengio presented another architecture inspired by the animal visive cortex named Convolutional Neural Networks [59], which will have a huge success in the field of image processing.

Anyway, after the initial enthusiasm, the artificial neural networks lost interest, especially due the limited resources available with that time's CPUs. In the mid-2000, the computing power increased through the use of GPUs and distributed computing and neural networks were deployed on a large scale,

particularly in image and visual recognition problems. Since then, it became known as "deep learning".

2.2.1 Fundamentals of the Artificial Neural Networks

An artificial neural network (ANN), is a mathematical/informatical model calculation based on biological neural networks. This model is constituted by a group of interconnections of information consisting of artificial neurons and processes using a connectionist approach to computation. In most cases, an artificial neural network is an adaptive system that changes its structure, which is based on external or internal information that flows through the network during the learning phase. In practical terms neural networks are non-linear structures of statistical data organized as modelling tools. They can be used to simulate the complex relationships between inputs and outputs that other analytic functions fail to represent. An artificial neural network receives external signals on a layer of nodes (processing unit) input, each of which is connected with a number of internal nodes, organized in several levels. Each node processes the received signals performing a very simple task and transmits the result to subsequent nodes.

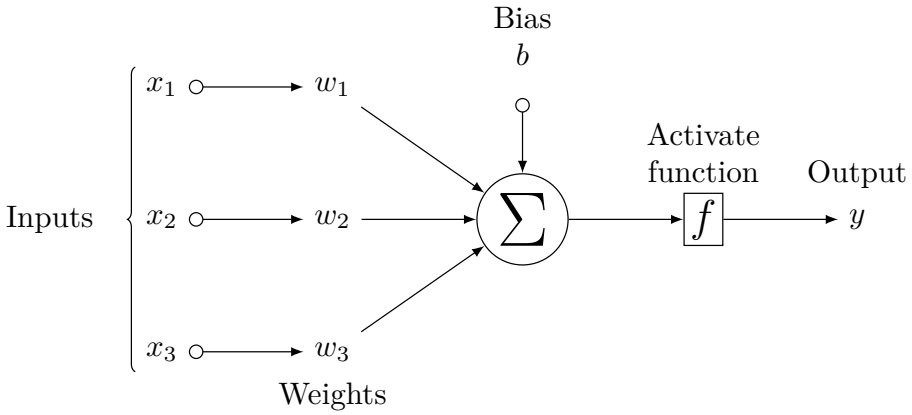


Figure 2.3: The artificial neuron model.

The artificial neuron is an information-processing unit that is fundamental to the operation of a neural network. The model of a neuron is composed of three basic elements, as shown in (Figure 2.3)

- a *set of synapses*, or connecting links, each of which is characterized by a weight or strength of its own, w_{km}; The neural model also includes an externally applied *bias*, denoted by b_k.

- an *adder* for summing the input signals, weighted by the respective synaptic strengths of the neuron; the operations described here constitute a linear combiner;
- an *activation function* for limiting the amplitude of the output of a neuron. Typically, the normalized amplitude range of the output of a neuron is written as the closed unit interval [0,1], or, alternatively, [-1,1].

The most typical non-linear function $\varphi(x)$ employed as activation functions are:

- the *sigmoid function*: it is defined as a strictly increasing function that exhibits a graceful balance between linear and nonlinear behavior; an example of the sigmoid function is the *logistic function* defined by:

$$\varphi\left(v\right) = \frac{1}{1 + exp\left(-av\right)} \tag{2.1}$$

- the *hyperbolic tangent* (*tanh*): it is simply a scaled and shifted version of the sigmoid function, defined as:

$$\varphi(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.2}$$

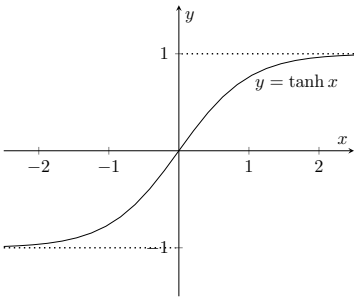


Figure 2.4: The *tanh* non-linear function.

- the *Rectifier Linear Unit* (*ReLU*):

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

- the $softmax$: it is used on the last layer of a classifier setup: the outputs of the softmax layer represent the probabilities that a sample belongs to the different classes. Indeed, the sum of all the output is equal to 1.

$$\varphi(x_k) = \frac{e^{x_k}}{\sum_{j=1}^{N} e^{x_j}} \text{ for } k = 1, \ldots, K \tag{2.4}$$

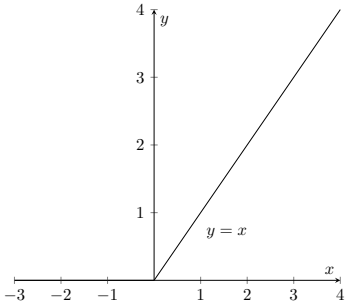


Figure 2.5: The *ReLU* non-linear function.

The link input-output, which is the transfer function of the network, is not programmed but is simply obtained by a learning process based on empirical data. At the beginning of the training procedure, the weights w_{km} are randomly initialized, and they are adjusted as learning proceeds. There are three major learning paradigms, each corresponding to a particular abstract learning task. They are supervised learning, unsupervised learning, and reinforcement learning. Usually a type of network architecture may be used in each of these tasks:

- *the Supervised Learning:* is used if is available a set of data for the training comprising typical examples of the inputs and their corresponding outputs: in this way the network can learn to infer the relation that binds them. Subsequently, the network is trained by means of an appropriate algorithm (typically, the backpropagation which is precisely a supervised learning algorithm), which uses such data for the purpose of modifying the weights and other parameters of the network to minimize the estimation error for the whole training. If the training is successful, the network learns to recognize the unknown relationship that binds the input variables to the output, and is therefore able to make predictions even where the output is not known a priori; in other words, the final objective supervised learning is the prediction of the output value for each valid value input, relying only on a limited number of examples of correspondence. To do this, the network must be finally provided with an adequate generalization ability, with reference to cases unknown to it. This will solve the problems of regression or classification;

- *The Reinforcement Learning:* The algorithms for reinforcement learning ultimately trying to determine a policy to maximize the incentives received by the agent accumulated in the course of its exploration of the problem. The reinforcement learning differs from the supervised because it has never seen pairs of input-output examples known, nor shall correct

explicit suboptimal actions. In addition, the algorithm is focused on the online use, which implies a balance between exploitation and exploration of unknown situations of current knowledge;

- *The non-Supervised Learning:* is based on training algorithms that modify the network weights only by reference to a set of data that includes just the input variables. These algorithms attempt to group the input data and therefore to identify the appropriate cluster representative of the data, typically by making use of topological methods or probabilistic. The unsupervised learning is also used to develop techniques for data compression.

2.3 Deep Neural Network Architectures for Analysis of Sound Events

The manner in which the neurons of a neural network are structured is intimately linked with the task they are designed for. Hereafter, a brief description of the neural network architectures employed in this thesis for the analysis of sound events is provided.

2.3.1 Multi Layer Perceptron (MLP)

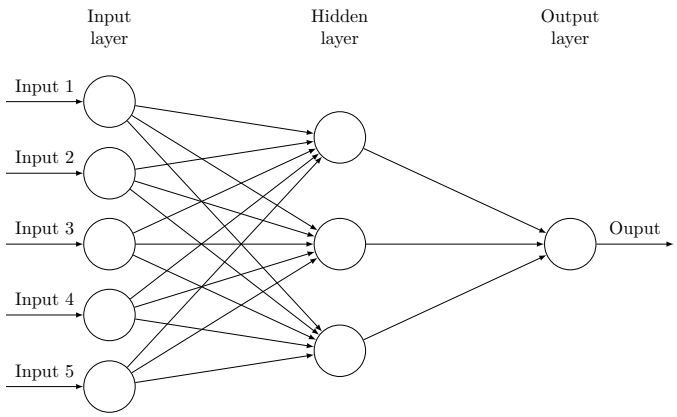


Figure 2.6: The MLP Neural Network.

The Multi Layer Perceptron (MLP) or Multilayer Feedforward Network is characterized by the presence of one or more hidden layers, whose computation nodes are correspondingly called *hidden neurons*. The MLP is one of the first

"deep" architectures being introduced in 1986 [58]. Artificial Neural Networks are often referred as deep when they have more than 1 or 2 hidden layers. Indeed, it is well known that an MLP with one or more hidden layers and a sufficient number of non-linear units (neurons) can approximate any continuous function on a compact input domain with arbitrary precision. Each node applies an activation function over the weighted sum of its inputs. The units are arranged in layers, with feed forward connections from one layer to the next. The behaviour of this architecture is parametrized by the connection weights, which are adapted during the supervised network training. In the forward pass, input examples are fed to the input layer, and the resulting output is propagated via the hidden layers towards the output layer. At the backward pass, the error signal originating at the output neurons is sent back through the layers and the network parameters (i.e., weights and biases) are tuned.

A single neuron can be formally described as:

$$g(\mathbf{u}[n]) = \varphi\left(\sum_{j=1}^{D} w_j u_j[n] + b\right), \tag{2.5}$$

where $\mathbf{u}[n] \in \mathbb{R}^{D\times 1}$, the bias b is an externally applied term and $\varphi(\cdot)$ is the non-linear activation function. Thus, the mathematical description of a one-hidden-layer MLP is a function $\mathbf{f} : \mathbb{R}^{D} \rightarrow \mathbb{R}^{D'}$, where D' is the size of the output vector, so:

$$\mathbf{f}(\mathbf{u}[n]) = \varphi\left(\mathbf{b}_2 + \mathbf{W}_2\left(\varphi\left(\mathbf{b}_1 + \mathbf{W}_2 \cdot \mathbf{u}[n]\right)\right)\right), \tag{2.6}$$

where $\mathbf{W}_i$ and $\mathbf{b}_i$ are the respective synaptic weights matrix and the bias vector of the i-th layer.

Regarding the computational complexity, considering additions and multiplications as separate operations, the total number of operations for each layer is given by

$$\text{Cost}_{\text{MLP}} = \sum_{i=1}^{P} 2Q_{i-1}Q_i + \sum_{i=1}^{P} Q_i, \tag{2.7}$$

where P is the number of layers, and Q_i denotes the number of units of layer i. The first term, considers the number of operations for a linear unit, while the second term considers the operations required by the ReLU activation function (i.e., the maximum operation).

2.3.2 Convolutional Neural Networks (CNN)

The birth of this kind of feed-forward neural network is related to image processing [60]. Indeed, neural networks as MLPs take vectors as input, while this

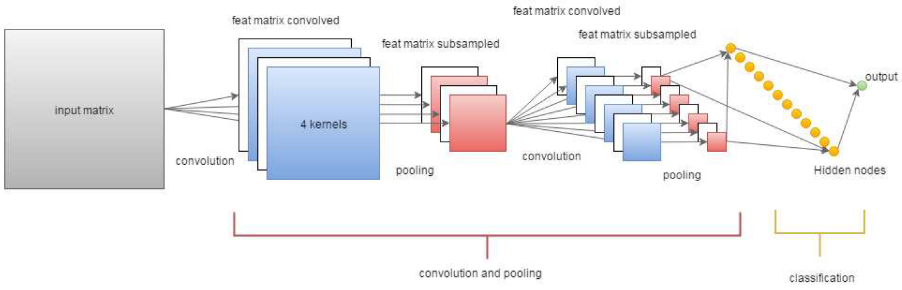


Figure 2.7: The Convolutional Neural Network.

condition is not satisfied in the image case. The problem is tackled by means of convolutional layers, whose aim is to process restricted a area of a 2-D input, similarly to what happens in the animal visual cortex. Convolution kernels process the input data matrix by dividing it in local *receptive fields*, a region of the same size of the kernel, and sliding the local receptive field across the entire input. Thus, the whole input matrix is processed by repeated application of a function across its sub-regions, obtaining so-called *feature maps*. Practically, this is implemented by a convolution of the input data with a linear filter, adding a bias term and then applying a non-linear function. The weights in each *feature map* are *shared*: all hidden neurons are aimed to detect exactly the same pattern just at different locations in the input image. The kernels are generally small compared to the input, allowing CNNs to process large inputs with few trainable parameters.

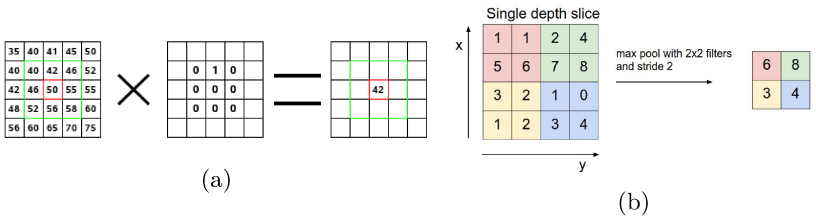


Figure 2.8: Details of *convolution* (a) and *max-pooling* (b) operations.

CNN is a feed-forward neural network [61] usually composed of three types of layers: convolutional layers, pooling layers and layers of neurons, as shown in Figure 2.7. Pooling layer just reduces the dimension of the matrix by a rule: a submatrix of the input is selected, and the output is the maximum value of this submatrix. The pooling process introduces tolerance against shifts of the input patterns. Together with convolution layer it allows the CNN to detect if a particular event occurs, regardless its deformation or its position. Finally, at the top of the network, a layer of neurons is applied. This layer does not differ

from MLP, being composed by a set of activation and being fully connected with the previous layer.

Denoting with $\mathbf{W}_m \in \mathbb{R}^{K_{1m} \times K_{2m}}$ the m-th kernel and with $\mathbf{b}_m \in \mathbb{R}^{D_1 \times D_2}$ the bias vector of a generic convolutional layer, the m-th feature map $\mathbf{h}_m \in \mathbb{R}^{D_1 \times D_2}$ is given by:

$$\mathbf{h}_m = \varphi \left(\sum_{d=1}^{D_3} \mathbf{W}_m * \mathbf{u}_d + \mathbf{b}_m \right), \tag{2.8}$$

where $*$ represent the convolution operation, and $\mathbf{u}_d \in \mathbb{R}^{D_1 \times D_2}$ is a matrix of the three-dimensional input tensor $\mathbf{u} \in \mathbb{R}^{D_1 \times D_2 \times D_3}$. The dimension of the m-th feature map $\mathbf{h}_m$ depends on the zero padding of the input tensor: here, padding is performed in order to preserve the dimension of the input, i.e., $\mathbf{h}_m \in \mathbb{R}^{D_1 \times D_2}$.

The computational complexity of the CNN is given by the sum of the computational complexity of the convolutional layers and of the fully connected layers. The complexity of convolutional layers can be obtained from (2.8) and from the number of operations of the max-pooling operator. Regarding the former, supposing square kernels, i.e., $K_{1m} = K_{1m} = K_m$, the number of operations required for calculating the feature map $\mathbf{h}_m$ is

$$\text{Cost}_{\text{Per-Feature-Map}} = 2K_m^2 D_1 D_2 D_3 + D_1 D_2 (D_3 - 1), \tag{2.9}$$

where the first term of the sum considers the number of operations required for the convolution and the sum with the bias term, and the second term the operations for the ReLU activation function. As aforementioned, the max-pooling operator calculates the maximum over a $P_1 \times P_2$ matrix. Supposing that the maximum operation is calculated pair by pair, the max-pooling operator requires the following number of operations

$$\text{Cost}_{\text{Max-Pooling}} = (P_1 P_2 - 1)(D_1 - P_1 + 1)(D_1 - P_2 + 1). \tag{2.10}$$

The total number of operations per layer can be calculated by multiplying the expressions in (2.9) and (2.10) by the number of kernels and summing the contributions. Finally, the total number of operations of the CNN can be obtained by summing the individual contributions of the convolutional layers and of the fully connected layers.

2.3.3 Siamese Neural Networks

The Siamese Neural Network is an architecture able to learn a latent representation of a given input. In particular, a SNN is composed of two twin networks with binded weights. A pair of inputs is provided to the system, one to each

twin network. Downstream, the network maps these inputs into two different representation vectors. Then, a certain type of distance between those two representations is computed. The euclidean distance is typically used.

Consider X_1, X_2 as a pair of two input samples and $Y(X_1, X_2)$ as the label assigned to this pair, we assign $Y = 0$ (positive example) if the inputs X_1 and X_2 are from the same distribution, $Y = 1$ (negative example) otherwise. The euclidean distance between the mapping $S_e(X_1)$ and $S_e(X_2)$ performed by the network is defined as:

$$E_w = |S_e(X_1) - S_e(X_2)| . \tag{2.11}$$

The training procedure consists in minimize the differences of X_1, X_2 for inputs belonging to the same class ($Y = 0$) while maximize the differences for inputs of different classes ($Y = 1$). The loss function used to achieve this minimization is the contrastive loss, described by LeCun et al. in [62]:

$$Loss = (1 - Y)\frac{1}{2}(E_w)^2 + (Y)\frac{1}{2}\{(max(0, m - E_w)\}^2. \tag{2.12}$$

Here the parameter $m > 0$ is the *margin* that allows only negative examples whose distance is less than the radius defined by m itself, to contribute to the loss function.

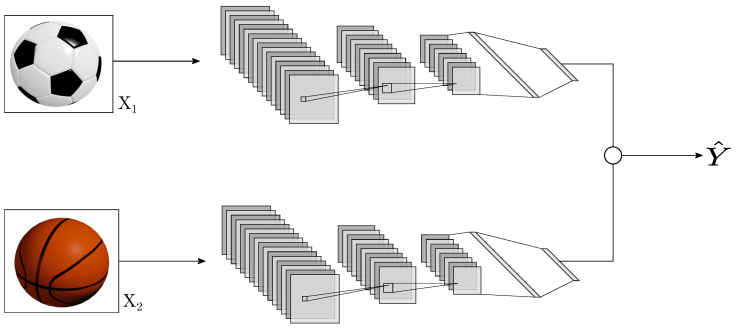


Figure 2.9: Siamese Neural Networks.

2.3.4 Generative Adversarial Networks (GAN)

The Generative Adversarial Networks (GAN) were introduced by Ian Goodfellow et al. in 2014 [63] with the aim to artificially generate photographs with many realistic characteristics, looking authentic to human observers. The architecture is composed of two networks: one generates candidates (genera-

tive), while the other network evaluates them (discriminative). Typically, the generative network learns to map from a latent space to a particular data distribution of interest, while the discriminative network classify the data produced by the generator between instances from the true data distribution and synthetic data. The generative network's training objective is to increase the error rate of the discriminative network (i.e., "fool" the discriminator network by producing novel synthesized instances that appear to have come from the true data distribution). The objective function of the discriminative is instead a more traditional binary classification cost function (i.e., binary cross-entropy).

In [64], adversarial autoencoders have been introduced with the objective of turning an autoencoder into a generative model (Figure 2.10). In particular, the encoder portion of the network learns to convert a data distribution into a prior distribution. On the other side, the decoder learns to map a prior distribution into the data distribution, indeed becoming a generative network. The discriminator network operates between the encoder and the decoder: it takes as input data generated by the encoder or data sampled from the prior distribution and it gives as output the probability of the data of being generated by the encoder.

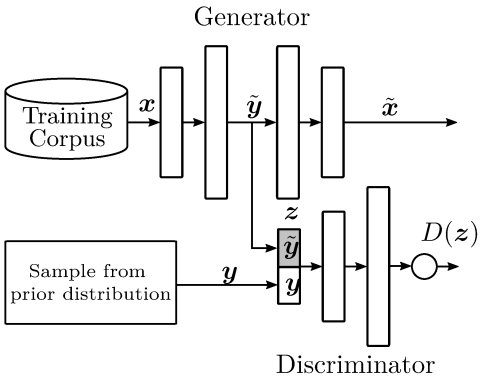


Figure 2.10: The adversarial autoencoder approach proposed in [64]. $\tilde{\boldsymbol{y}}$ is the output of the encoder, and $\boldsymbol{y}$ represents data sampled from a prior distribution.

2.3.5 Recurrent Neural Networks (RNN)

The Recurrent Neural Networks (RNN)s are a particular neural architecture designed to make use of the information obtained from prior network states, therefore giving the network the capability to "remember" past context information. This feature is obtained by adding a set of recursive connections going from and to each hidden neuron. Due this characteristic, the output is now computed only after a batch of input frames are forwarded into the network

so that the final decision will rely on the correlation between different adjacent frames. Indeed, these characteristics place the RNNs in close connection with the audio processing, due to the sequential nature of the signal, which depends of its temporal evolution. In practice, two are the typologies which have been highly employed in addition to the "standard" RNNs: the network based on the Long Short Term Memory (LSTM) units and on the Gated Recurrent Unit (GRU). Both of these architectures rely on computational units which act as memory blocks, thus they are able to encapsulate mid-long term characteristics of the audio signal. In addition to the memory blocks, the bidirectional RNNs [65] are also common architectures. A bidirectional RNN can access context from both temporal directions, which makes it suitable i.e. for speech recognition, where whole utterances are decoded. This is achieved by processing the input data in both directions with two separate hidden layers. Both hidden layers are then fed to the output layer.

Long Short Term Memory (LSTM)

Compared to a conventional RNN, in the LSTM RNN [66] the hidden units are replaced by so-called memory blocks. These memory blocks can store information in the 'cell variable' $\boldsymbol{c}_t$. In this way, the network can exploit long-range temporal context. Each memory block consists of a memory cell and three gates: the input gate, output gate, and forget gate, as depicted in Figure 2.11.

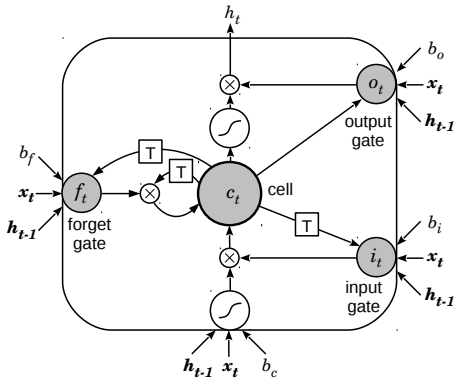


Figure 2.11: Long Short-Term Memory block, containing a memory cell and the input, output, and forget gates

These gates control the behaviour of the memory block. The activation vector of each gate is computed as, for example for the input gate,

$$\boldsymbol{i}_t = \tanh(\boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{ci}\boldsymbol{c}_{t-1} + \boldsymbol{b}_i). \tag{2.13}$$

The forget gate can reset the cell variable which leads to 'forgetting' the stored

input $\boldsymbol{c}_t$, while the input and output gates are responsible for reading input from $\boldsymbol{x}_t$ and writing output to $\boldsymbol{h}_t$, respectively:

$$\boldsymbol{c}_t = \boldsymbol{f}_t \otimes \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \otimes \tanh(\boldsymbol{W}_{xc}\boldsymbol{x}_t + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \tag{2.14}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \otimes \tanh(\boldsymbol{c}_t) \tag{2.15}$$

where $\otimes$ denotes element-wise multiplication and the activation function (here the tanh) is also applied in an element-wise fashion. The variables $\boldsymbol{i}_t$, $\boldsymbol{o}_t$, and $\boldsymbol{f}_t$ are the output of the input gates, output gates and forget gates, respectively, $\boldsymbol{b}_c$ is a bias term, and $\boldsymbol{W}$ is a weight matrix. Each memory block can be regarded as a separate, independent unit. Therefore, the activation vectors $\boldsymbol{i}_t$, $\boldsymbol{o}_t$, $\boldsymbol{f}_t$, and $\boldsymbol{c}_t$ are all of the same size as $\boldsymbol{h}_t$, i. e., the number of memory blocks in the hidden layer. Furthermore, the weight matrices from the cells to the gates are diagonal, which means that each gate depends only on the cell within the same memory block.

Gated recurrent units (GRU)

Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks, introduced in 2014 [67]. Their performance on polyphonic music modeling and speech signal modeling was found to be similar to that of LSTMs. However, GRUs have been shown to exhibit better performance on smaller datasets. GRU layers control the information flow through a gated unit structure, which depends on the layer input, on the activation at the previous frame and on the reset gate. For frame t, the total activation of GRU layer is a linear interpolation of previous activation h_{t-1} and the candidate activation h_t as:

$$\boldsymbol{h}_t = \boldsymbol{u}_t \cdot \boldsymbol{h}_{t-1} + (1 - \boldsymbol{u}_t) \cdot \boldsymbol{h}_t, \tag{2.16}$$

where $\boldsymbol{u}_t$ denotes the update gate. Candidate activation $\boldsymbol{h}_t$ is a function of $\boldsymbol{h}_{t-1}$, the GRU layer's input $\boldsymbol{x}_t$ and the reset gate r_t. *tanh* and *hardsigmoid* activation functions are used for update and reset gates, respectively. GRU's activation is mainly controlled by reset gate when the GRU layer's input $\boldsymbol{x}_t$ is significantly different than in previous frames. When reset gate is closed $r_t = 0$, the candidate activation does not include any contribution from $\boldsymbol{h}_{t-1}$. Fast response to the changes in the input and the previous activation information is fundamental for high performance in the proposed algorithm, where the task is to detect a small chunk of consecutive time frames where the target event is present.

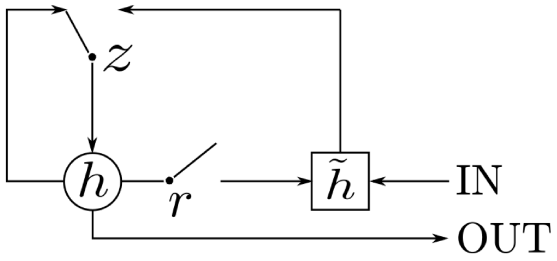


Figure 2.12: The Gated Recurrent Unit, containing the input, output, and forget gates

2.3.6 Capsule Neural Networks (CapsNet)

The CapsNet architecture relies on the CNN architecture and includes its computational units in the first layers of the network as invariant features extractor from the input hereafter referred as $\mathbf{X}$. Following Hinton's preliminary works [53], in the CapsNet presented in [52] two layers are divided into many small groups of neurons called *capsules*. In those layers, the scalar-output feature detectors of CNNs are replaced with vector-output capsules and the dynamic routing, or *routing-by-agreement* algorithm is used in place of max-pooling, in order to replicate learned knowledge across space.

Formally, we can rewrite (2.8) as

$$\mathbf{H}_m = \begin{bmatrix} \alpha_{11}\mathbf{W}_{11}\mathbf{X}_1 + \ldots + \alpha_{M1}\mathbf{W}_{1M}\mathbf{X}_M \\ \vdots \\ \alpha_{1K}\mathbf{W}_{K1}\mathbf{X}_1 + \ldots + \alpha_{MK}\mathbf{W}_{KM}\mathbf{X}_M \end{bmatrix}. \tag{2.17}$$

In (2.17), $(\mathbf{W} * \mathbf{X})$ has been partitioned into K groups, or capsules, so that each row in the column vector corresponds to an output capsule (the bias term $\mathbf{b}$ has been omitted for simplicity). Similarly, $\mathbf{X}$ has been partitioned into M capsules, where $\mathbf{X}_i$ denotes an input capsule i, and $\mathbf{W}$ has been partitioned into submatrices $\mathbf{W}_{ij}$ called *transformation matrices*. Conceptually, a capsule incorporates a set of properties of a particular entity that is present in the input data. With this purpose, coefficients α_{ij} have been introduced. They are called *coupling coefficients* and if we set all the $\alpha_{ij} = 1$, (2.8) is obtained again.

The coefficients α_{ij} affect the learning dynamics with the aim to represent the amount of agreement between an input capsule and an output capsule. In particular, they measure how likely capsule i may activate capsule j, so the value of α_{ij} should be relatively accentuated if the properties of capsule i coincide with the properties of capsule j in the layer above. The coupling coefficients are calculated by the iterative process of dynamic routing to fulfill the idea of assigning parts to wholes. Capsules in the higher layers should com-

prehend capsules in the layer below in terms of the entity they identify, while dynamic routing iteratively attempts to find these associations and supports capsules to learn features that ensure these connections.

Dynamic Routing

After giving a qualitative description of routing, we describe the method used in [52] to compute the coupling coefficients. The activation of a capsule unit is a vector which holds the properties of the entity it represents in its direction. The vector's magnitude indicates instead the probability that the entity represented by the capsule is present in the current input. To interpret the magnitude as a probability, a *squashing* non-linear function is used, which is given by:

$$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1 + \|\mathbf{s}_j\|^2} \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}, \tag{2.18}$$

where $\mathbf{v}_j$ is the vector output of capsule j and $\mathbf{s}_j$ is its total input. $\mathbf{s}_j$ is a weighted sum over all the outputs $\mathbf{u}_i$ of a capsule in the Primary Capsule layer multiplied by the coupling matrix $\mathbf{W}_{ij}$:

$$\mathbf{s}_j = \sum_i \alpha_{ij} \hat{\mathbf{u}}_{j|i}, \;\; \hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij} \mathbf{u}_i. \tag{2.19}$$

The routing procedure works as follows. The coefficient β_{ij} measures the coupling between the i-th capsule from the Primary Capsule layer and the j-th capsule of the Detection Capsule layer. The β_{ij} are initialized to zero, then they are iteratively updated by measuring the agreement between the current output $\mathbf{v}_j$ of each capsule in the layer j and the prediction $\hat{\mathbf{u}}_{j|i}$ produced by the capsule i in the layer below. The agreement is computed as the scalar product

$$c_{ij} = \mathbf{v}_j \cdot \hat{\mathbf{u}}_{j|i}, \tag{2.20}$$

between the aforementioned capsule outputs. It is a measure of how similar the directions (i.e., the proprieties of the entity they represent) of capsules i and j are. The β_{ij} coefficients are treated as if they were log likelihoods, thus the agreement value is added to the value owned at previous routing step r:

$$\beta_{ij}(r+1) = \beta_{ij}(r) + c_{ij}(r) = \beta_{ij}(r) + \mathbf{v}_j \cdot \hat{\mathbf{u}}_{j|i}(r), \tag{2.21}$$

where r represents the routing iteration. In this way the new values for all the coupling coefficients linking capsule i to higher level capsules are computed. To ensure that the coupling coefficients α_{ij} represent log prior probabilities, the softmax function to β_{ij} is computed at the start of each new routing iteration.

Formally:

$$\alpha_{ij} = \frac{\exp(\beta_{ij})}{\sum_k \exp(\beta_{ik})}, \tag{2.22}$$

so $\sum_j \alpha_{ij} = 1$. Thus, α_{ij} can be seen as the probability that the entity represented by capsule Primary Capsule i is a part of the entity represented by the Detection Capsule j as opposed to any other capsule in the layer above.

Margin loss function

The length of the vector $\mathbf{v}_j$ is used to represent the probability that the entity represented by the capsule j exists. The CapsNet have to be trained to produce long instantiation vector at the corresponding k_{th} capsule if the event that it represents is present in the input audio sequence. A separate margin loss is defined for each target class k as:

$$\begin{aligned} L_k = T_k \max(0, m^+ - \|\mathbf{v}_j\|)^2 + \\ \lambda(1 - T_k) \max(0, \|\mathbf{v}_j\| - m^-)^2 \end{aligned} \tag{2.23}$$

where $T_k = 1$ if an event of class k is present, while λ is a down-weighting factor of the loss for absent sound event classes classes. m^+, m^- and λ are respectively set equal to 0.9, 0.1 and 0.5 as suggested in [52]. The total loss is simply the sum of the losses of all the Detection Capsules.

2.4 Optimization Algorithms

Most deep learning training algorithms involve optimization of some sort. The most widely used is the gradient based optimization, which belongs to the first order type. *Optimization* is the task of either minimizing some function $f(x)$ by altering x: $f(x)$ is called *objective function*, but in the case when it has to be minimized, it is also call the *cost function*, *loss function*, or *error function*. The aim of the optimization is reached doing small change ϵ in the input x, to obtain the corresponding change in the output $f(x)$:

$$f(x + \epsilon) \approx f(x) + \epsilon\, f'(x). \tag{2.24}$$

This formulation is based on the calculation of the derivative $f'(x)$. The *gradient descent* is the technique based on the reduction of $f(x)$ by moving x in small steps with the opposite sign of the derivative. The aim is to find the minimum of the cost function: when $f'(x) = 0$, the derivative provides no information about which direction to move, therefore this point is defined as stationary points. A local minimum is a point where $f(x)$ is lower than at all neighbouring and it is no longer possible to decrease $f(x)$ by making infinites-

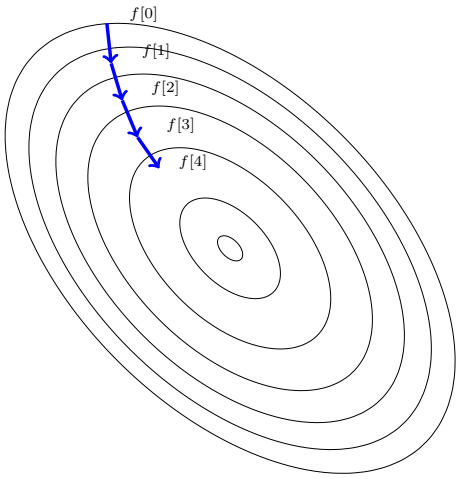


Figure 2.13: An example of the application of gradient descent to search for minimum of a function in a figurative 2-D plane. The process is iterated four times, while the small central ellipse corresponds to the minimum of the function.

imal steps. The absolute lowest value of $f(x)$ is a *global minimum*. For the concept of minimization to make sense, there must still be only one (scalar) output. For functions that have multiple inputs $f : \mathbb{R}^n \to \mathbb{R}$, the concept of *partial derivatives* is introduced. The gradient $\nabla_{\mathbf{x}} f(\mathbf{x})$ is the vector containing all the partial derivatives.

The method of *steepest descent* or *gradient descent* states that decrease f by moving in the direction of the negative gradient.

$$\mathbf{x'} = \mathbf{x} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x}), \tag{2.25}$$

where ϵ is the *learning rate*, a positive scalar determining the size of the step. Large training sets are necessary for good generalization, but large training sets are also more computationally expensive. The cost function decomposes as a sum over training example of per-example loss function: i.e., the negative conditional log-likelihood of the training data is defined as:

$$J(\theta) = \mathbb{E}(L(\mathbf{x}, y, \theta)) = \frac{1}{m} \sum_{i=1}^{m} L(\mathbf{x}^{(i)}, y^{(i)}, \theta), \tag{2.26}$$

where L is the per-example loss $L(\mathbf{x}, y, \theta) = -\log p(y|\mathbf{x}; \theta)$. The gradient de-

scent requires computing:

$$\nabla_\theta J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(\mathbf{x}^{(i)}, y^{(i)}, \theta). \tag{2.27}$$

The computational cost of this operation is proportional to the number of examples m, therefore as the training set size grows the time to take a single gradient step becomes prohibitively long.

2.4.1 Stochastic Gradient Descent (SGD)

Stochastic gradient descent (SGD) is an extension of the gradient descent algorithm: the insight is that the gradient is an expectation estimated using a small set of samples. On each step of the algorithm, a sample of example $\mathbb{B} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m')}\}$, called *minibatch*, is drawn uniformly from the training set. The minibatch size m' is typically chosen to be a relatively small number of examples. The estimate of the gradient is: $\mathbf{g} = \frac{1}{m'} \nabla_\theta \sum_{i=1}^{m'} L(\mathbf{x}^{(i)}, y^{(i)}, \theta)$ using examples from the minibatch $\mathbb{B}$. The SGD algorithm then follows the estimated gradient downhill:

$$\theta \leftarrow \theta - \epsilon\, \mathbf{g} \tag{2.28}$$

where ϵ is the learning rate.

2.5 Generalization Techniques

In order to obtain more robust models, different techniques have been proposed to *regularize* the weight update during the neural networks training. They have the aim to improve the generalization proprieties of the model, i.e., the ability to perform on newly unseen data as well as (in a reasonable manner) on the training set. A brief description of the most common techniques is given in the following paragraphs.

2.5.1 Dropout

Dropout [68] is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. The term "dropout" refers to the operation of randomly exclude units during the training of a neural network.

Practically, during a training iteration, a percentage (drop-rate) of units for each layer where the the dropout is employed are not updated. They are se-

lected randomly at each new step, therefore it is like we were training a network with a different layout. This allows to prevent overfitting on the training data, making the single units more robust to the conditions changing.

2.5.2 Batch Normalization

Batch Normalization [69] is a technique which consists to add a normalization "layer" between each layers, in order to reduce the problem of internal covariate shift. In the case that the input distribution of a learning system, such as a neural network, changes, one speaks of a so-called covariate shift. If this change happens on the input of internal nodes of (deep) neural networks, it is called an internal covariate shift

An important thing to note is that normalization has to be done separately for each dimension (input neuron), over the 'mini-batches', and not altogether with all dimensions. Hence the name 'batch' normalization. Due to this normalization "layers" between each fully connected layers, the range of input distribution of each layer stays the same, no matter the changes in the previous layer.

Formally, given $\mathbf{x}$ inputs from the k^{th} neuron, the *Batch Normalization* is computed as follows:

$$\hat{\mathbf{x}} = \frac{\mathbf{x}^k - \mathrm{E}(\mathbf{x}^k)}{\sqrt{\mathrm{Var}[\mathbf{x}^k]}} \tag{2.29}$$

Normalization brings all the inputs centered around 0. This way, there is not much change in each layer input. So, layers in the network can learn from the back-propagation simultaneously, without waiting for the previous layer to learn. This speeds up the training of networks, leading to the possible usage of higher learning rates.

Chapter 3

Datasets and Evaluation

Every problem to be solved with machine learning and in particular with deep learning techniques requires the availability of a sufficient amount data for algorithm parametrization: the ability to access public dataset, representative of a real scenario, allows to test the approaches, in order to evaluate the effective benefit in real applications, and to compare the performance of existing algorithms on a common comparison basis. A reliable evaluation procedure for a classification or recognition system will involve a standard dataset of example input data along with the intended target output, and well-defined metrics to compare the systems' outputs with this ground truth.

3.1 Datasets

Labelled data has a crucial influence on algorithm development and evaluation in research fields dealing with classification and detection. Any machine learning algorithm is only as good as the data behind it in terms of modeling and generalization properties.

Well-established and easily accessible benchmark databases attract the interest of the research community as readily available support for research, thus accelerating the pace of development for related fields. There are many well-known databases in related research areas, such as TIMIT [70] for speech recognition or the Million Song Dataset [71] for various tasks in music information retrieval. In view of this critical influence, the process of creating a dataset for system developing is, naturally, very delicate. The content must be carefully selected to provide sufficient coverage of the aspects of interest, sufficient variability in characterizing these aspects, and a sufficient quantity of examples for robust modeling. Unfortunately, there is no rule on what "sufficient" means, as it usually depends on the projected use of the data.

Compared to speech, the categories and sequences of sound events are not so straightforward to define, as any object or being may produce a sound. Environmental sounds are often produced in a context of interaction combined with movement when manipulating an object. The sounds can be organized

into categories based on different properties, such as the sound source (e.g., cup), its production mechanism (e.g., hit), or even the source properties (e.g., metal), and the same sound can belong to many different categories, depending on the chosen property (sound when hitting a metallic cup). In addition, there are no specific rules on how environmental sounds co-occur. Environmental sounds belong to different levels of cognitive representation, in connection with the surface characteristics of sounds and the sources that produce them. Studies like [72, 73] concern the perceptive characterization of these sounds and of the cognitive mechanisms used to identify them. The process that combines perception and action of environmental sound is one of the major research themes. This is a vital and unique gateway for research in auditory cognition and for interactive sound design applications.

For this reason, building a database for environmental sounds is a complicated task, with the choice of classes usually dictated by the targeted research, and the size limited by practical aspects of data collection especially compared to well-known datasets available for image processing (i.e., Imagenet[74]). Anyway, very recently a project promoted by Google has been presented [55], with the aim to collect a large human annotated dataset and a relative ontology obtained from YouTube videos.

3.1.1 Dataset Acquisition

Recording real-world audio is the obvious data collection method for obtaining realistic data. Creating a new dataset by recording new data has the advantage of producing a collection with controlled audio quality and content.

Typically, the recording settings such as microphone type, number of channels, sample rate, bit depth are defined in the planning phase of the project, and often they also depend on the final objectives of the project (i.e., a mobile application vs. a research corpus). Anyway, the most common sampling settings are 44.1 kHz, 16 bit respectively for sampling-rate and bit depth. Numerous computational auditory scene analysis research projects make use of binaural heads setups (Figure 3.1), with the aim to replicate the frequency-dependent distortions of phase and amplitude on sounds produced by the human auditory system.

Although the use of the same settings and device(s) throughout the data will result in a better quality set, some recent algorithms are designed to counteract these diversities, leading to the possibility to gather material from different recording campaigns. In fact, one disadvantage of recording new data is that in order to cover as much acoustic variability and diversity as possible, recordings must be done in many different conditions. For location-specific modeling, this may mean different weather or human activity conditions, while for more

Figure 3.1: Dummy head for binaural recording.

general modeling, it would also require traveling to other locations, adding significant time and effort to the data collection procedure.

3.1.2 Dataset Labelling

Labeled data has a crucial influence on algorithm development and evaluation in research fields dealing with classification and detection. The textual labels used in annotation must provide a good description of the associated category and not allow misinterpretation. To allow supervised learning and evaluation, the audio must have corresponding reference annotations. These annotations can be produced manually or in various semi-automatic ways, with the quality and level of detail available in the obtained annotation often depending on the procedure used. Manual annotation involves human annotators that will produce a mapping of the audio content into textual labels. Manually annotating sound scene audio material is relatively fast, while for sound events the process is much slower, with annotation using weak labels being much faster than with strong labels. Manual annotation is prone to subjectivity arising from the selection of words for labels and placing of temporal boundaries.

3.1.3 Acoustic Features

The time domain representation of a sound signal, or waveform, is not easy to interpret directly. Therefore, frequency-domain representations and time-frequency domain

representations (including multiscale representations) have been used for years providing *features* of the sound signals that are more in line with the

human perception. In this section we report a brief description of the most commons.

Mel Frequency Cepstral Coefficients

MFCCs is a well-known set of features widely employed in audio applications, especially for the purpose of representing speech data. Indeed, the weighting operation performed by the Mel bands emulates the frequency response of the human hearing organ, which is sensible at its most to speech frequencies. The extraction procedure requires few stages. An excerpt of the signal is transformed in the frequency domain by means of STFT. The obtained spectrum is hence mapped to the mel scale by using triangular overlapping windows. For each mel frequency the logs of the powers are considered, to whom the Discrete Cosine Transform (DCT) is applied. The resulting spectrum are the MFCCs. Furthermore, a common procedure consists in concatenating MFCCs with their first and second derivatives, in order to provide a temporal evolution of the signal.

LogMel

LogMel features have been recently applied in the field of acoustic modelling and music structure analysis [57, 58], leading to encouraging results. The procedure for LogMel extraction shares several aspects with the one described for MFCCs features. In details, a set of mel-band filters is applied to the spectrogram of the signal, from which the logarithm of the power spectrum for each band is considered. The logarithmic transformation is applied to each sub-band energy in order to match the human perception of loudness. However, due to the absence of the DCT, no spatial compression is performed to the features, which remain correlated in the frequency domain. In this thesis, the employment of LogMel matches the choice of using some of the systems proposed in the next chapters (e.g., CNNs), where the objective is to exploit the intrinsic correlation of input features in

order to highlight repetitive patterns present in the features.

Pitch

The tone of human voice is highly characteristic, for that reason is used i.e. in VAD systems. The Sub-Harmonic-Summation (SHS) method described in [75] is one of the most common algorithms. In particular, the spectrum is shifted along the log-frequency axis, for each shift the spectrum is scaled and then summed. This procedure creates the sub-harmonic summation spectrum, where peak picking is applied to determine pitch. The used frame size is equal to 50 ms.

WC-LPE Feature

The Wavelet Coefficient (WC) and Linear Prediction Error (LPE) feature set has been recently developed and exploited for audio onset detection purpose [76]. WC-LPE carries information on non-stationary parts of the signal, thus, it can facilitate the speech boundaries identification. Firstly, the signal undergoes the Discrete Wavelet Transformation (DWT). Then, each sub-band of the wavelet-domain is filtered by a set of Linear Prediction Error Filter (LPEFs) and Forward Prediction Errors (FPE) are extracted. Finally, the first derivatives are obtained from wavelet coefficients and added to them in order to form the feature set.

RASTA-PLP

With RASTA-PLP we refer to RelAtive SpecTrAl transform - Perceptual Linear Prediction, a feature set often exploited to represent speech [77]. Short-term noise variations are smoothed and the constant offset is removed by RASTA filtering procedure, while the task of PLP is to simulate several well-known properties of the hearing system.

Scattering Transform

The scattering transform [78] is the operation on which the Deep Scattering Spectrum (SCAT) is based. SCAT is a multi-resolution representation of a signal based on a tree of complex wavelet filters followed by a non-linearity. SCAT proves successful in gathering information at multiple resolutions on non-stationary signals thanks to its good properties, such as translation invariance, stability to small diffeomorphisms and uniqueness, e.g., time-warping deformations.

3.1.4 Data Augmentation

Data augmentation refers to methods for increasing the amount of development data available without additional recordings. With only a small amount of data, systems often end up overfitting the training data and performing poorly on unseen data. Thus, it is desirable to artificially increase the amount and diversity of data used in training.

Many techniques have been proposed for data augmentation in domains different from audio. In the case of image processing, affine transformations such as rotation, shear, scaling or zooming are very common perturbations to apply directly on the raw data (i.e., the images composing the dataset) in order to augment the training sets of the systems. Some approaches extend the augmentation in the feature space [79]. Dataset augmentation could be used to

reduce overfitting while training supervised learning models or to counteract the dataset imbalance i.e., if the classes are not approximately equally represented as in the case of SMOTE [80, 81].

In the audio field, data augmentation exploits techniques such as pitch shifting, time stretching or the generation of multi-microphone data by means of simulated impulse response of the acoustic space or performing the transformation not in input space, but in the feature space. Adding random gaussian noise to the input data (raw waveform or acoustic features) can also be seen as a data augmentation technique for the audio domain.

With the development of the artificial neural networks (ANN), some works act this augmentation by means of auto-encoder model also using Generative ANN. The above mentioned approaches have a large impact in the development of data driven approaches to activity recognition, with applications in the Active and Assisted Living (AAL) domain that has a lack of large scale, high quality, and annotated datasets even if open datasets are growing. In particular, these models are trained to reproduce the signal they have been trained with, and if their latent space is properly perturbed, they are able to generate new data, useful to extend the training sets of detection/classification systems. In this context we can also mention the Generative Adversarial Networks (GAN) (cf. Section 2.3.4).

A limited number of repositories have supported the notion of shared datasets, including a small number of activity recognition related resources. To address this lack it is possible to simulate smart environments equipped with heterogeneous sensors and combine the signals from different domains with novel data augmentation techniques exploiting the aforementioned deep learning techniques.

3.2 Evaluation Setup

During system development, iterative training and testing of the system is necessary for tuning its parameters. For this purpose, the labeled data available for development is split into disjoint training and testing sets. However, labeled data is often in short supply, and it is difficult to choose between using more data for training (leading to better-performing systems) or for testing (giving more precise and reliable estimates of system performance). For an efficient use of all the available data, system development can use cross-validation folds to artificially create multiple training/testing splits using the same data; overall performance is then the average of performance on each split. Cross-validation folds also help avoid overfitting and supports generalization properties of the system, so that when the system is used on new data it is expected to have similar performance as seen with the data used for development.

As shown in Figure 3.2, one round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are then combined over the rounds to give an estimate of the model's predictive performance.

Cross-validation folds also help avoid overfitting and supports generalization properties of the system, so that when the system is used on new data it is expected to have similar performance as seen with the data used for development.

If available, a separate set of examples with reference annotation can be used to evaluate the generalization properties of the fully tuned system. We refer to this set as *evaluation set*, and use it to evaluate how the system would perform on new data.

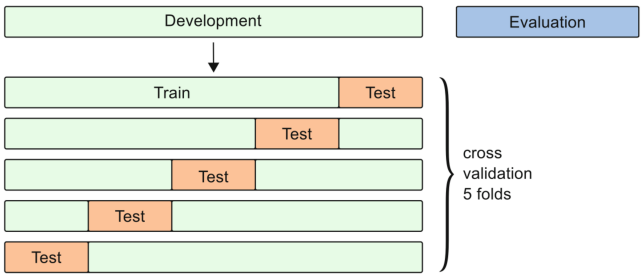


Figure 3.2: Splitting a dataset into development and evaluation data, with five folds for crossvalidation during system development. Picture courtesy of [1].

3.3 Evaluation Metrics

Evaluation is usually referred as estimating the performance of a system under test when confronted with new data. For an objective evaluation, the system is fed previously unseen data for which reference annotations are available. The system output is then compared to the reference to calculate measures of its performance.

What performance means and how it should be measured may vary depending on the specifications and requirements of the developed system: We can measure accuracy to reflect how often the system correctly classifies or detects a sound, or we can measure error rates to reflect how often the system makes mistakes. By using the same data and the same methodology to evaluate different systems, a fair and direct comparison can be made of systems' capabilities.

The metrics used in detection and classification of sound events include accuracy, precision, recall, F-score, area under the curve (AUC) or error rate (ER). There is no metric universally good for every kind of algorithm, as they each reflect different perspectives on the ability of the system.

3.3.1 Metrics Computation

Basically, the evaluation metrics are computed by comparing the prediction of the system under analysis with the respective annotations or *ground truth*. Thus, the metrics are calculated based on counts of the correct predictions and different types of errors made by the system. These counts are referred to as intermediate statistics and are defined depending on the evaluation procedure. These intermediate statistics are defined as follows for a target sound event:

- True positive: A correct prediction, meaning that the system output and the reference both indicate the event present.
- True negative: The system output and the reference both indicate event not present.
- False positive: The system output indicates event present or active, while the reference indicates event not present.
- False negative: The system output indicates event not present or inactive, while the reference indicatesit as present.

Sound event classification is usually a single-label multiclass problem, and the resulting intermediate metrics reflect whether the single true class is correctly recognized for each example. In this task there is no distinction between false positives and false negatives. In sound event detection, the choice of measurement determines the interpretation of the result: With a segment-based

metric, the performance shows how well the system correctly detects the temporal regions where a sound event is active; with an event-based metric, the performance shows how well the system is able to detect event instances with correct onset and offset. Thus, in the segment-based metric the ground truth and system output are compared in a fixed time grid, and sound events are marked as active or inactive in each segment. For the event-based metric the ground truth and system output are compared at event instance level. Specifically, the intermediate statistics for sound event detection are defined as follows:

- Substitutions S: are the number of ground truth events for which we have a false positive and one false negative in the same segment;
- Insertions I: are events in system output that are not present in the ground truth, thus the false positives which cannot be counted as substitutions;
- Deletions D: are events in ground truth that are not correctly detected by the system, thus the false negatives which cannot be counted as substitutions;

If we consider the scenario of polyphonic sound event detection, the segment-based metric essentially splits the duration of the test audio into fixed length segments that have multiple associated labels, reflecting the sound events active anywhere in the given segment. In this respect, evaluation verifies if the system output and reference coincide in the assigned labels, and the length of the segment determines the temporal resolution of the evaluation. Event-based metrics compare event instances one to one. Since the time extents of the events detected by the system may not exactly match the ground truth, a common approach is to allow a time misalignment threshold or *time-collar*.

Performance Metrics

Measures of performance are calculated based on accumulated values of the intermediate statistics. We denote by TP, TN, FP, and FN the sums of the true positives, true negatives, false positives, and false negatives accumulated throughout the test data. In the case of multiclass problem, the accumulation of intermediate statistics can be performed either globally or separately for each class, depending on the nature of the problem (i.e., instance-based or class-based) or datasets characteristics (i.e., highly unbalanced classes). Based on the total counts of the intermediate statistics, many different measures can be derived. We can define:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.1}$$

$$\text{Precision} = \frac{TP}{TP+FP} \tag{3.2}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{3.3}$$

$$\text{F-score} = \frac{2TP}{2TP+FN+FP} \tag{3.4}$$

Accuracy measures how often the classifier makes the correct decision, as the ratio of correct system outputs to total number of outputs. Precision, recall, and F-score were introduced in the context of information retrieval. F-score can be also calculated as the harmonic mean of Precision and Recall scores:

$$\text{F-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.5}$$

F-score has the advantage of being a familiar and well understood metric. Its main drawback is that its value is strongly influenced by the choice of averaging and the data balance between classes: in instance-based averaging the performance on common classes dominates, while in class-based averaging (balanced metrics) it is necessary to at least ensure presence of all classes in all folds in the test data, to avoid cases when recall is undefined. In particular, in this work we consider also the use of the Unweighted Average Recall (UAR), which is more appropriate in the case of classification of samples belonging to an highly unbalanced dataset. UAR is defined as:

$$UAR = \frac{1}{NClass} \cdot \sum_{c=1}^{NClass} \frac{TP_c}{TP_c + FN_c} \tag{3.6}$$

In the case of sound event detection systems, Error Rate score is the most common evaluation metric. Considering a single time frame t_1, the ER is computed from its intermediate statistics, i.e., the number of substitutions ($S(t_1)$), insertions ($I(t_1)$), deletions ($D(t_1)$) and active sound events from annotations ($N(t_1)$). Formally, for the entire evaluation set:

$$ER = \frac{\sum_{t_1=1}^{T} S(t_1) + \sum_{t_1=1}^{T} I(t_1) + \sum_{t_1=1}^{T} D(t_1)}{\sum_{t_1=1}^{T} N(t_1)}, \tag{3.7}$$

where T is the total number of segments t_1.

3.3.2 Detection Metrics

Precision and recall rely on hard decisions made for each trial, they typically depend on a threshold applied to some underlying decision variable, i.e., the

output of the neural network. Lowering the threshold will increase likelihood of accepting both positive and negative examples, improving recall but in many cases hurting precision. Although F-score combines these values at a single threshold in an attempt to balance this tradeoff, a more complete analysis can be provided by plotting a function proportional to the metric over the full range of possible thresholds. Some examples are, the precision-recall (P-R) curve (showed in Figure 3.3) and the receiver operating characteristic (ROC) curve. The latter plots true positive rate (TPR = $TP/TP+FN$) as a function of the false positive rate (FPR = 1 - Recall) as the decision threshold is varied.

These curves carry rich information, they can be difficult to compare, so a single figure of merit summarizing the tradeoff is desirable. The relative "compressed" scores for P-R and ROC curve are respectively Average Precision (AP) score or Area under curve (AUC), defined as:

$$\mathrm{AP} = \sum_n (R_n - R_{n-1}) P_n, \tag{3.8}$$

where R_n and P_n are the Recall and Precision for threshold n respectively and

$$\mathrm{AUC} = \int_{\infty}^{-\infty} \mathrm{TPR}(T)\mathrm{FPR}'(T)\, dT \tag{3.9}$$

Both AUC and AP vary between 0 and 1, with an uninformative classifier yielding 0.5, while the ideal system yelds 1.

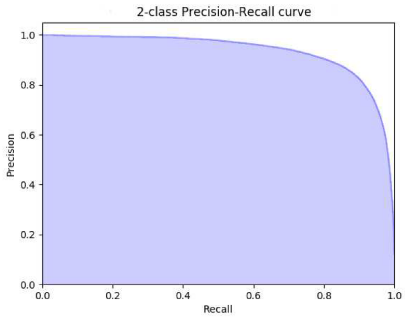


Figure 3.3: Example of 2-class Precision-Recall curve. Relative AP is equal to 0.9341.

3.3.3 Final Remarks

Estimates of metrics on classes with very few examples are also intrinsically noisy. Any dataset of real-world recordings will most likely have unbalanced event classes; therefore, the experiment setup must be built with the choice of

metric in mind. In a cross-validation approach, a more stable result is given by treating the cross-validation folds as single experiment, meaning that metrics are calculated only after training and testing all folds, not as average of the individual folds nor as average of individual class performance. In addition, reporting the variance among the individual folds' contributions to the average can serve as a useful confidence interval. Anyway, if there are multiple scenes in the dataset, typically evaluation metrics are calculated for each scene separately and then the results are presented as the average across the scenes.

Attention should be also paid to statistical significance of the results and it should be used to calculate the theoretical limits of discriminability of the evaluation, especially when two methods/approaches/techniques are compared.

A detailed and visualized explanation of evaluation score in multi label setting for sound event analysis can be found in [82].

Chapter 4

Sound Event Classification

The task of sound event classification consists of categorize an audio recording into one of a set of predefined categories, by associating a textual descriptor.

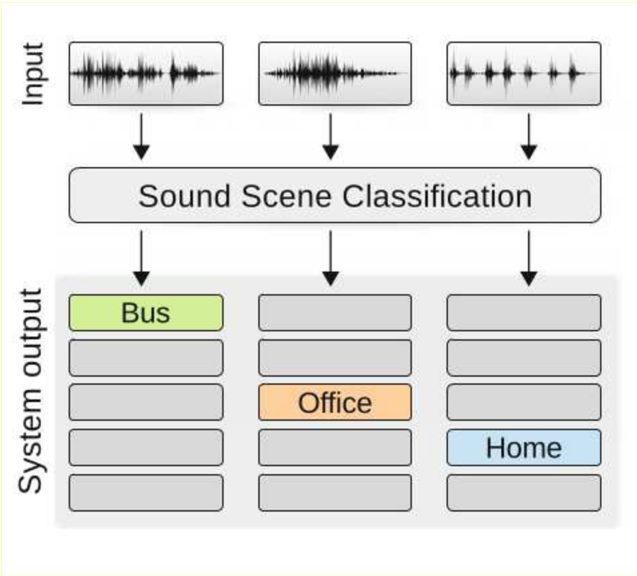


Figure 4.1: System input and output for the sound event classification. Pic. courtesy of [1].

In this chapter three works belonging to this "sub-category" are shown. They are respectively: Snore Sounds Excitation Localization, which has been performed by using *Scattering Transform* and Deep Neural Networks; Road Surface Roughness Classification from Acoustic Signals, which has been performed by means of Convolutional Neural Networks; Bird Audio Detection, which - despite of its name - consists in determining a binary decision for the presence/absence of bird sounds on audio files recorded in very different conditions and it has been performed by using CapsNets.

4.1 Snore Sounds Excitation Localization

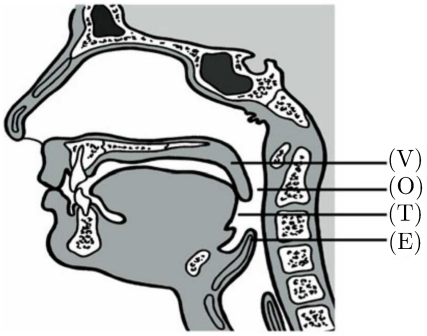


Figure 4.2: Corresponding positions of the VOTE classification in the upper airway. Picture courtesy of [83].

People spend almost a third of their life sleeping, thus the sleep quality is very important to people's health. Most of us have experienced trouble sleeping just sometimes, while for some people sleep problems occur regularly; in former cases a sleep disorder is diagnosed. Among these, one of the most common sleep disorder [84] is the chronic snoring.

Snoring is defined as the emission, during the sleep, of a sound more or less annoying associated to the respiratory activity. This sound is caused by the vibration of soft tissue in the upper airways. It is confirmed that almost a half of the adult population snores occasionally, while approximately 30% of the overall population suffers from chronic snoring almost every night [85]. Sound snoring can have a negative impact on normal physical, mental, social and emotional functioning of the person suffering from it and their bed partner [86]. In addition, it can be associated with Obstructive Sleep Apnea (OSA), a chronic disease that can severely affect health [87]. OSA is characterised by posterior pharyngeal occlusion for at least ten seconds with apnea/hypopnea and attendant arterial oxyhemoglobin desaturation. If left untreated, this life-threatening sleep disorder may induce high blood pressure, coronary heart disease, pulmonary heart failure and even nocturnal death [88]. In addition, OSA is indicated between the main causes of significant morbidity among children [89]. Numerous surgical methods have been proposed to cure snoring, but many of them do not solve completely the issue if the origin of the vibration is not precisely located. Although the exact mechanism of snoring sound generation is highly influenced by the individual anatomy, the typical areas from which the snore noise is produced have been located in the soft palate, the uvula, the palatine tonsils, the base of the tongue, and the epiglottis. Drug induced sleep

endoscopy (DISE) is a standard examination technique used to identify the location and form of vibrations and obstructions [90]. During a DISE procedure, a flexible nasopharyngoscope is introduced into the upper airways while the patient is in a state of artificial sleep. Vibration mechanisms and locations can be observed while video and audio signals are recorded. This procedure has several disadvantages: it is time consuming, the patient is put under strain, and it has to be performed during drug-induced sleep. Recent studies have found that acoustic signal carries important information about the snore source and obstruction site in the upper airway of OSA patients [91], thus acoustic analysis of snoring sound could be an alternate, less-invasive method to identify the kind of snoring pathology [92]. This significant discovery has motivated several researchers to develop acoustic-based approaches that could provide inexpensive, convenient and non-invasive monitoring and screening apparatuses to be combined with traditional diagnostic tools. In addition, it can be performed during natural sleep, avoiding the possibility to induce different muscle relaxation patterns which is a cause for possible diagnosis errors in drug-induced sleep.

Related Works

Several works have been presented in the recent years on multi-feature acoustic analysis methods with the aim to classify and segment snore/non-snore sleep sounds. In [93] consists of the identification and segmentation process by using energy and Zero Crossing Rate (ZCR), which were used to determine the boundaries of sound segments. Episodes have been efficiently represented into two-dimensional spectral features by using principal component analysis, and classified as snores or non-snores with Robust Linear Regression (RLR). The system was tested by using the manual annotations of an Ear-Nose-Troth (ENT) specialist as a reference. The accuracy for simple snorers was found to be 97.3% when the system was trained using only simple snorers' data. It drops to 90.2% when the training data contain both simple snorers' and OSA patients' data. In the case of snore episode detection with OSA patients, the accuracy is 86.8%. In [94] tracheal respiratory signals are recorded and snore segments are detected by extracting 10 temporal and spectral features and an Artificial Neural Network (ANN). In [95] the automatic sound segmentation into snoring/breathing/noise episodes is performed using an adaptive effective-value threshold method for noise reduction, feature extraction of both linear and nonlinear descriptors and a Support Vector Machine (SVM) classifier.

Specifically regarding the determination of the vibration or occlusion mechanisms, the use of different acoustic feature sets has been proposed in [83], while in [96] a k-nearest neighbor (k-NN) classifier is fed with different acoustic features. A performance comparison of different feature sets in combination

with frequently-used classifier model is shown in [97]. Recently some works have been presented in occasion of the *Snoring* task of the Interspeech 2017 Computational Paralinguistics Challenge (ComParE) [98]. The task consisted in identification of the snore type among four classes based on the widely used Velum-Oropharyngeal-Tongue-Epiglottis (VOTE) scheme, distinguishing four structures that can be involved in upper airway narrowing and obstruction. In [99, 100] the authors exploit deep convolutional neural networks pre-trained on image datasets for feature extraction from Short Time Fourier Transform (STFT) representation of the snore sounds. Then, the outputs of the bottom layers are used to feed a SVM model which provides the snore sound classification. An alternative approach used weighted kernel classifiers [101] in order to counteract the natural snore dataset imbalance, such as Extreme Learning Machine and Kernel Partial Least Squares learners. Acoustic low level descriptors encoded over utterances are used as input features vector of the models. The latter algorithm is reported as the winner in the final challenge ratings[1].

4.1.1 Proposed Approach

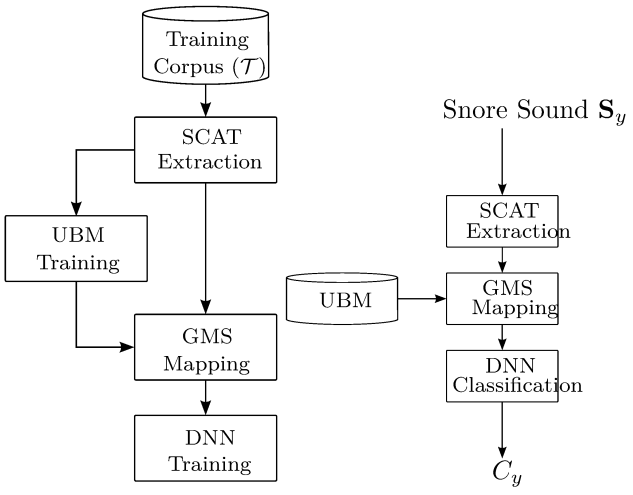
In this section we propose a snore classification algorithm based on Deep Scattering Spectrum (SCAT) [102] and Multi-layer Perceptron (MLP) neural networks. The specific characteristic of the snore sounds and the references reported above demonstrate that typical techniques for speech analysis are required for an accurate classification, in particular feature vectors with high dimensionality. In fact, the snore sound carries salient informations at different time scales, thus in this case it is necessary to capture patterns up to 500 ms. The SCAT provides an efficient representation of an audio signal based on the scattering transform. In particular, it extends the traditional Mel-Frequency Cepstral Coefficients (MFCCs) representation [103] with second-order scattering coefficients which characterize transient phenomena such as attacks and amplitude modulation. The algorithms has been evaluated on the Munich-Passau Snore Sound Corpus (MPSSC) [98] and the results are expressed in terms of Unweighted Average Recall (UAR) in order to compare our approach with the results of the INTERSPEECH 2017 ComParE.

The block diagram of the proposed method is shown in Figure 4.3. The algorithm operates by computing the deep scattering spectrum of the audio signal and then by mapping it into a Gaussian Mean Supervectors (GMS) [104] for classification. In particular, referring to Figure 4.3a, the set of snoring sounds $\mathcal{T} = \{(\mathbf{x}_1, C_1), (\mathbf{x}_2, C_2), \ldots, (\mathbf{x}_K, C_K)\}$ represents the training set of the algorithm, where $\mathbf{x}_k = [x_k(1), x_k(2), \ldots, x_k(T_k)]$ is a snoring sound signal of length T_k and $C_k \in \{V, O, T, E\}$ is the corresponding snoring label. The

[1] `http://emotion-research.net/sigs/speech-sig/is17-compare`

snore audio samples of the MPSSC corpus are of various duration ranging from 0.5 to 3 seconds, thus before computing the SCAT, audio signals shorter than 3 s are padded to the same length equal to 3 seconds. Padding is performed by repeating a signal or part of it until the total amount of audio samples is equal to 48000. Then, by means of the Hamming window, the audio signals are divided into half overlapped frames $x_{k,l}[n] = x_k[n + l(N - P + 1) - P + 1]$, where l is the frame index, $N = 8000$ and $P = 4000$.

In the "SCAT" stage every input audio signal $\mathbf{x}_k$ is converted into informative values that describe the main characteristics of the audio signal concatenating the scattering transform coefficients of each audio frame $SCAT(x_{k,l})$, with $l = 1, \ldots, 11$. In the training phase, a Universal Background Model (UBM) represented by a Gaussian Mixture Model (GMM) is trained on the SCAT vectors computed on the set $\mathcal{T}$ by using the Expectation Maximisation algorithm. Then, the "GMS Extraction" block consists in adapting the UBM with the Maximum a Posteriori (MAP) algorithm by using the SCAT of each snoring sound in $\mathcal{T}$ as input. The resulting mean values of each Gaussian of the GMM are concatenated into a GMS, thus mapping the SCAT coefficients time series into vectors of fixed lengths. The final step of the training phase consists in training the Deep Neural Network (DNN) classifier. In this section we use a Multi-layer Perceptron neural network and we compare the performance with a state-of-the art approach such as Support Vector Machine (SVM). In the classification phase (Figure 4.3b), given a snore sound $\mathbf{x}_y$, the relative GMS is obtained from the UBM as in the training phase, and then it is fed to the classifier in order to find the corresponding label $C_y \in \{V, O, T, E\}$.



(a) Training. (b) Classification.

Figure 4.3: Scheme of the proposed approach.

Deep Scattering Spectrum

The scattering transform [78] is the operation on which the Deep Scattering Spectrum (SCAT) is based. SCAT is a multi-resolution representation of a signal based on a tree of complex wavelet filters followed by a non-linearity. The SCAT up to order M of a signal $x_{k,l}$ is, thus,

$$\begin{aligned}\mathbf{S}(x_{k,l}, M) =& \{x_{k,l} * \phi_M, \\ & |x_{k,l} * \psi_{m_1}| * \phi_M, \\ & ||x_{k,l} * \psi_{m_1}| * \psi_{m_2}| * \phi_M\},\end{aligned} \tag{4.1}$$

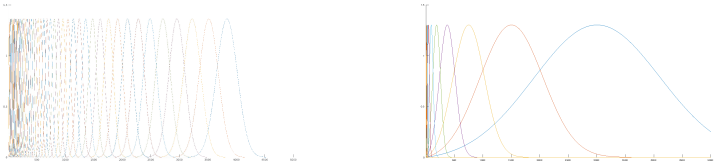
where ϕ_M is a low-pass filter and ψ_{m_i} is a wavelet filter with $i < M$. Time indices are omitted for brevity. SCAT proves successful in gathering information at multiple resolutions on non-stationary signals thanks to its good properties, such as translation invariance up to level M, stability to small diffeomorphisms and uniqueness, e.g., time-warping deformations. It has been applied with success in image [105] and audio signal classification tasks [106], where it is shown that SCAT provides optimal results when $M = 2$ though there is no upper limit on the order of the SCAT. It is showed that a two orders cascade of wavelet filter banks and rectifiers improve results obtained by MFCC and Delta-MFCC descriptors in musical genre classification task, thus the energy remaining on higher order branching is as low as background noise and provides no additional information. The SCAT outputs time-averaged coefficients, providing informative signal invariants over potentially large time scales. To compute the SCAT we used the *ScatNet* open source MATLAB library [107]. The choice of the wavelet filters is not trivial. In audio processing a typical procedure is to define constant-Q filter banks as linear operators that make up the layers of the scattering networks. The quality factors of the filter used in this section are $Q_1 = 8$ and $Q_2 = 1$. We assume 0.5 s as minimum length of snore utterance, thus the filter length N is set equal to 8000 samples.

The wavelet functions are built by dilating a mother wavelet ψ by a factor $2^{1/Q}$, for the quality factors Q, so as to obtain the filter bank:

$$\psi_{m_i}(t) = 2^{-m/Q}\psi(2^{-m/Q}t), \tag{4.2}$$

with t representing the time index.

The mother wavelet ψ may only be dilated up to the maximum scale $M-1$, resulting in 2 wavelets ψ_{m_0}, ψ_{m_1}. The parameter M determines the maximum wavelet time support $a^M = T$, thus the low-pass filter ϕ_M covers the interval $[-\pi a^{-M}, \pi a^{-M}]$ and represents the length of averaging window over a neighborhood of t. In this section, we use $T = N/8 = 1000$ samples, which corresponds to around 60 ms of audio.



(a) First stage. (b) Second stage.

Figure 4.4: The Morlet Wavelet Filterbanks

To improve the classification performance, scattering representation has been processed with the functions of renormalization and logarithmic transformation provided by *ScatNet*. The renormalization consists in dividing second-order scattering coefficients $\mathbf{S}(x_{k,l}, m_2)$ by the first order coefficients $\mathbf{S}(x_{k,l}, m_1)$, in order to decorrelate the amplitudes at the second layer from the first layer. Log-power representation of the frequency distribution is a typical choice in many acoustical classification tasks, thus after the renormalization the logarithm of scattering coefficients is computed.

For each audio frame, we obtain the SCAT representation $\mathbf{S}(x_{k,l}, M) \in \mathbb{R}^{D \times R}$, where $D = 1 + 52 + 167 = 220$ is the total number of scattering coefficients (all orders combined) and R is the number of time points, which is equal to 16. Finally, joining the $\mathbf{S}(x_{k,l}, M)$ for $l = 1, \ldots, 11$ we obtain the deep scattering spectrum of the snore signal $\mathbf{S}(\mathbf{x}_k, M) \in \mathbb{R}^{D \times 11 \cdot R}$.

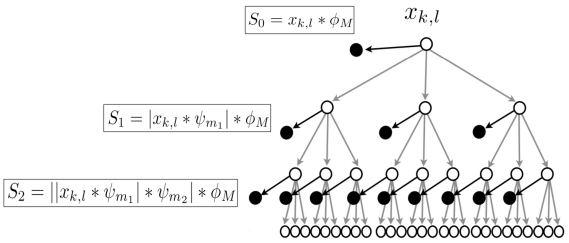


Figure 4.5: The SCAT extraction tree. Picture courtesy of Andén et. al.

Gaussian Mean Supervectors

A GMS is extracted according to the following procedure. Let $\mathbf{s}_r = [\mathbf{S}(\mathbf{x}_k)]_{*,r}$ the vector of the scattering coefficients of size $D \times 1$ where r is the time point index, the GMM representing an UBM from the training corpus $\mathcal{T}$ is given by:

$$p(\mathbf{s}_r|\lambda) = \sum_{j=1}^{J} w_j p(\mathbf{s}_r|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j), \tag{4.3}$$

where $\lambda = \{w_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j | j = 1, 2, \ldots, J\}$, w_j are the mixture weights, and $p(\cdot|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ is a multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}_j$ of size $D \times 1$ and diagonal covariance matrix $\boldsymbol{\Sigma}_j$ of size $D \times D$.

The mean values of the UBM model are, then, adapted with the MAP algorithm and concatenated in order to obtain the GMS:

$$\boldsymbol{\Gamma} = [\boldsymbol{\mu}_1^T, \boldsymbol{\mu}_2^T, \cdots, \boldsymbol{\mu}_J^T]^T, \tag{4.4}$$

where T denotes the transpose operator. Regardless of the length of the input snore sound, $\boldsymbol{\Gamma}$ is a vector of $DJ \times 1$ length. The threshold values of EM and MAP algorithms during the adaptive phase have been set both equal to 0.001, but with different number of iterations, respectively equal 1000 and 5 for EM and MAP.

Multi Layer Perceptron

In this section we exploited the Multi Layer Perceptron (MLP) architecture as DNN Classifier [58]. The MLP is composed of an input layer with number of units equal to the dimension of the input supervector $\boldsymbol{\Gamma}$, followed by a stack of fully connected layers of units, namely the *hidden* layers. Number of hidden layers and their dimensions have been investigated during the experimental analysis. As non-linear activation function in the hidden layers of the MLP we employed both the hyperbolic tangent (*tanh*) and the rectified linear unit (*ReLU*) [108]. The output layer of the MLP is formed by four units with the *softmax* non-linear function. The outputs of the softmax layer represent the probabilities that a sample belongs to the different classes.

The behavior of this architecture is parametrized by the connection weights, which are adapted during the supervised network training, accomplished by using the Adam algorithm [109] for the stochastic gradient-based optimization of the crossentropy loss function. The optimizer parameters were set as follows: learning rate $lr = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The maximum number of training epochs was set equal to 200 with an early stopping strategy in order to reduce the computational burden. Nevertheless, overfitting is well-know a problem affecting DNNs in particular when the number of training samples is limited. In order to prevent overfitting we investigated the use of dropout [68].

The model has been implemented in the Python language using Keras[2] as deep learning library with Theano[110] as back-end.

[2]https://keras.io/

4.1.2 Comparative Approaches

Below is given a brief description of state-of-the-art approaches such as SVM classifiers and algorithms which showed the best performance at the Interspeech 2017 ComParE. They were evaluated on the blind *test* set, thus they are reported for comparative aims.

Support Vector Machines

Without modifying the general structure of the algorithm for supervectors $\mathbf{\Gamma}$ computation, we compared the performance of the MLP based model with the SVMs [111].

SVMs are binary classifiers that map an input example onto a high-dimensional feature space and iteratively searches for the hyperplane that maximises the distance between the training examples from the origin. Given the training data $\{\mathbf{\Gamma}_1, \ldots, \mathbf{\Gamma}_k\}$, where k is the number of observations, the class separation is performed by solving the following equation:

$$\min_{\mathbf{w},\xi,\rho} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{\nu k}\sum_i \xi_i - \rho \tag{4.5}$$

$$\text{subject to: } (\mathbf{w}^T \cdot \Phi(\mathbf{\Gamma}_i)) \geq \rho - \xi_i,\ \xi_i \geq 0, \tag{4.6}$$

where w is the support vector, ξ_i are slack variables, ρ is the offset, and Φ maps $\mathbf{\Gamma}_i$ into a dot product space F such that the dot product in the image of Φ can be computed by evaluating a certain kernel function. The kernel function $K(\cdot,\cdot)$ can assume different forms [112]. In this section two kernel functions have been considered, the Radial Basis Function (RBF), defined as:

$$K(\mathbf{\Gamma}, \mathbf{\Gamma}_i) = \exp(-\gamma\|\mathbf{\Gamma} - \mathbf{\Gamma}_i\|^2) \tag{4.7}$$

and the linear kernel, defined as:

$$K(\mathbf{\Gamma}, \mathbf{\Gamma}_i) = \mathbf{\Gamma}^T\mathbf{\Gamma}_i. \tag{4.8}$$

The decision values are obtained with the following function:

$$f(\mathbf{\Gamma}) = \mathbf{w}^T \cdot \Phi(\mathbf{\Gamma}) - \rho. \tag{4.9}$$

The input vector $\mathbf{\Gamma}$ is classified as $+1$ if $f(\mathbf{\Gamma}) \geq 0$ and -1 if $f(\mathbf{\Gamma}) < 0$. In this section, the multiclass problem has been addressed using the "one versus all" strategy. Implementation of LIBSVM [113] from Python library *scikit-learn* has been employed both in the training and testing phases of the SVM.

Weighted Kernel Classifiers

The approach proposed by Kaya and Karpov [101] exploits a decision fusion between different kernel based classifiers such as regular and weighted Kernel Extreme Learning Machine (KELM) and Kernel Partial Least Squares (KPLS) learners. As acoustic feature representation they used Fisher Vectors (FV), which provides an encoding of local descriptors (e.g., MFCC, RASTA-PLP and their derivatives), quantifying the gradient of the parameters of the background model with respect to the data. In addition, in the final submission system they used both (FV) and the ComParE baseline feature set, which contains 6373 static features resulting from the computation of various functionals over low-level descriptor (LLD) contours.

Image-based Deep Spectrum Features

A different approach is presented in [99], based on image classification convolutional neural network (CNN) descriptors extracted from snore spectrograms. They evaluated different well known DNN architectures typically used for image classification, such as AlexNet and VGG neural network. Both deep CNNs were previously trained on approximately 1.2 million images from the ImageNet corpus, then they compute the power spectral density on the dB power scale of the audio excerpt using Hanning windows of 16 ms width, and 8 ms overlap and they plotted the result in three different colour maps: gray, jet and viridis. The deep CNNs are fed with the spectrograms, then the neurons on the first and second fully connected layers (fc6 and fc7) are extracted as feature vectors. These feature vectors are then classified by means of an SVM model.

4.1.3 Experiments

The MPSSC dataset

The MPSSC dataset is composed of more than 30 hours of audio recordings captured during DISE examinations of 224 subjects from three medical centers recorded between 2006 and 2015. Recording equipment, microphone type, and location differ among the medical centers, so do the background noise characteristics. From the original signals (raw PCM, sample rate 16 000 Hz, quantization 16 bit) 843 early identifiable, single site of vibration snore events have been extracted and manually screened from medical experts. Following the 4-class VOTE scheme, each sound file in the dataset is labelled as V, O, T, E, depending on the tissue from which snore sound originates, as shown in Figure 4.2. They are respectively:

- (V) - Velum (palate), including soft palate, uvula, lateral velopharyngeal walls;

- (O) - Oropharyngeal lateral walls, including palatine tonsils;
- (T) - Tongue, including tongue base and airway posterior to the tongue base;
- (E) - Epiglottis.

The dataset is divided into three subsets: *train*, *devel* and *test*. The number of events per class in the database is strongly unbalanced with a high preeminence of the "Velum" (V)-class and "Oropharyngeal" (O)-class (85% of samples) but in line with the likelihood of occurrence during normal sleep, while 10% and 5% of samples respectively belongs to E-events and T-snores. Details of class occurrences are shown in Table 4.1.

The Munich-Passau Snore Sound Corpus			
#	train	devel	test
V	168	161	155
O	76	75	65
T	8	15	16
E	30	32	27
Σ	282	283	263

Table 4.1: The Munich-Passau Snore Sound Corpus - The table shows the number of events per class in train, devel and test.

As shown in the waveforms and the related spectrograms in Figure 4.6, the main energy components in three of the classes are concentrated in the frequency area below around 2000 Hz. Energy and spectral distribution characteristics are similar, except for the Type T, which shows higher energy content above 2500 Hz compared to the other three.

Experimental Setup

According to the ComParE 2017 guidelines [98], the performance metric for this task is the Unweighted Average Recall (UAR) (cf. Section 3.3.1).

The MLP hyperparameters optimization was obtained by means of a *random search* strategy [114]. The number of layers, the number of units per layers, the non-linear activation function and the dropout rate have been varied for a total of 400 configurations. Details of searched hyperparameters and their ranges are reported in Table 4.2. For all of the configurations, the performance of the MLP classifier has been evaluated by varying the input supervector dimension, which depends on the number of Gaussian components used to represent the UBM $J = 1, 2, 4, 8$. The supervector given as input to the MLP is standardized by

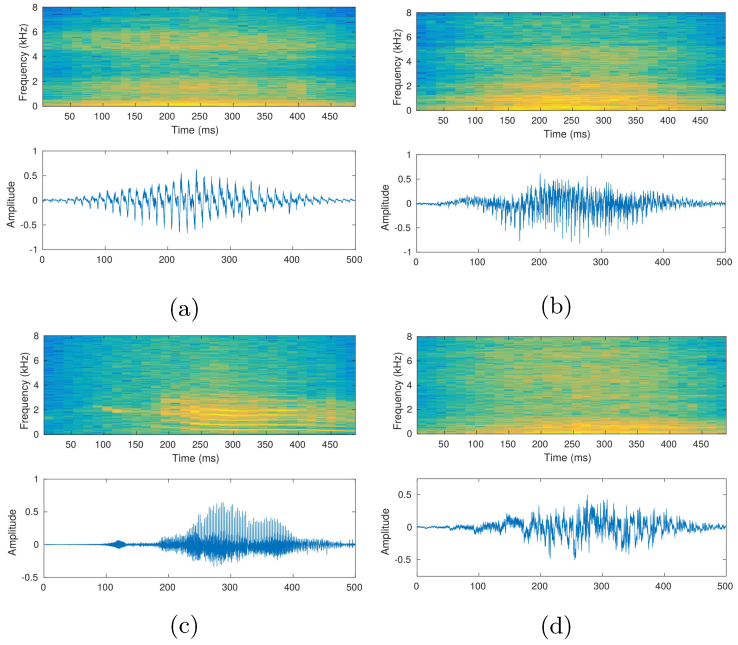


Figure 4.6: Waveforms and spectrograms of VOTE events
(a) V (velum); (b) O (oropharyngeal); (c) T (tongue base); (d) E (epiglottis).

removing the mean value of each $\boldsymbol{\mu}_j$ component and scaled by dividing for their standard deviation.

Parameter	Range	Distribution
MLP layers Nr.	[2 - 5]	uniform
MLP layers dim.	[20 - 256]	log-unifom
Activation	[*tanh* - ReLU]	uniform
Dropout Rate	[0.5 - 0]	uniform

Table 4.2: Hyper-parameters optimized in the random-search phase for the MLP classifier, and their range.

Regarding the SVM classifier optimization, we conducted a preliminary analysis which showed that in this task the linear kernel function provides better performance with respect to the RBF kernel. Then, with a grid search strategy we explored the SVM penalty parameter C which yielded the best results. In particular, C assumed the values $2^{-15}, 2^{-14}, \ldots, 2^{15}$ for a total of 30 different values. The supervectors at the input of the SVM were scaled to the range $[-1, 1]$.

During the training procedure of both classifiers, different weights were given to samples belonging to different classes in order to counteract the dataset

unbalancing. The weight for each class was computed on training set with the following equation:

$$W_c = \frac{N_{TOT}}{N_c}, \tag{4.10}$$

where N_{TOT} is the total number of samples in the training set and N_c the number of samples of the respective class. In this way the classes having a lower number of samples in the training set have a larger effect in the loss computing [115].

4.1.4 Results

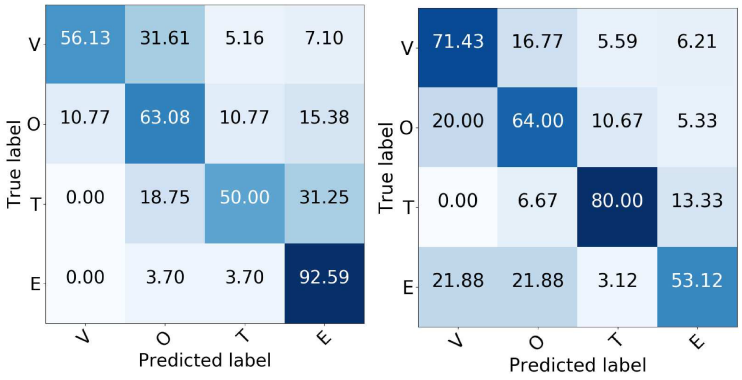
The performance of the proposed algorithm has been assessed firstly by using the *train* subset as training corpus and the *devel* subset for evaluation. Then, the same model was trained with both *train* and *devel* subsets and it was evaluated on the *test* subset. The results of the experiments are shown in Table 4.3. For the sake of conciseness only the best performances of the proposed approach are reported, comparing the results achieved with SVM and MLP classifiers on the two folds. For both *devel* and *test* subsets the best results are obtained with DNN based classifier.

Input	**Classifier**	**UAR devel (%)**	**UAR test (%)**
SCAT + 1 Gaussian UBM	MLP [204,112,99] with *tanh* and dropout	53.16	72.63
SCAT + 2 Gaussians UBM	MLP [249,40,21,21] with *tanh* and dropout	58.20	**74.19**
SCAT + 2 Gaussians UBM	MLP [235,227] with *tanh* and dropout	**67.14**	67.71
SCAT + 4 Gaussians UBM	MLP [156,34,21] with *relu* and dropout	50.30	71.32
SCAT + 8 Gaussians UBM	MLP [66,28] with *tanh* and dropout	55.89	70.18
SCAT + 1 Gaussian UBM	SVM, $C = 2^{-8}$	46.73	65.45
SCAT + 2 Gaussian UBM	SVM, $C = 2^{-10}$	46.54	65.50
SCAT + 4 Gaussian UBM	SVM, $C = 2^{-9}$	44.67	67.22
SCAT + 8 Gaussian UBM	SVM, $C = 2^{-10}$	43.20	65.48

Table 4.3: Comparative results in terms of UAR (%) score of proposed method for MLP and SVM based classifiers for both *devel* and *test* subsets. Best results on each subset are shown in **bold**.

Results on *devel* subset

On this subset the best MLP topology resulting from the random search is composed of 2 hidden layers with respectively $235, 227$ units, *tanh* as non linear activation function and a dropout rate equal to 0.5. The network input consists of supervectors originated from UBM with 2 gaussian components, thus the input size is equal to 440×1. This architecture provides an UAR up to 67.14%. Regarding the SVM based classifier, the best performing model on *devel* subset is fed with a supervector mapped on a UBM of 1 gaussian component and with the parameter $C = 2^{-8}$ obtain an UAR equal to 46.73%. The corresponding confusion matrices are illustrated in Figure 4.7, which exhibits for the MLP (Figure 4.7b) the higher recall (80.00%) for the minority class (T), but a moderate recall (53.12%) for the second smallest class (E). The SVM based classifier obtains the highest recall on the (E) class, while the other classes obtain worse performance. The superiority of the DNN based classifier is motivated by its well known ability to encode in the inner structure the features of the different snore sounds highlighted by the SCAT and better distinguish the samples, notwithstanding the large unbalancing of the dataset.



(a) (b)

Figure 4.7: Normalized confusion matrix of best performing models on *devel* subset. (a) SVM Classifier, (b) MLP Classifier.

Results on *test* subset

The network topology achieving the absolute best results optimized on *test* subset is composed of 4 hidden layers with respectively $249, 40, 21, 21$ units, *tanh* as non linear activation function and a dropout rate equal to 0.5. As in the case of *devel* subset the input size is equal to 440×1, consisting of supervectors originated from UBM with 2 gaussian components. This architecture shows an UAR up to 74.19%. The SVM model which obtains the best performance is

fed with supervectors mapped on 4 gaussians UBM and a parameter $C = 2^{-10}$. The respective UAR is equal to 67.22%. The obtained confusion matrices are illustrated in Figure 4.8. In this case the augmented number of samples has a beneficial effect on the recall score of almost all the four classes for the MLP classifier, while the performance of SVM classifier are similar to the *devel* subset.

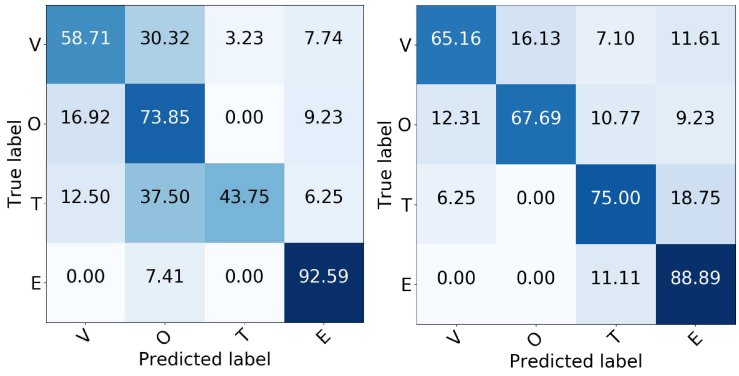


Figure 4.8: Normalized confusion matrix of the best performing models on *test* subset. (a) SVM Classifier, (b) MLP Classifier.

Comparative Results

In Table 4.4 the best performance obtained with methods presented in [101], which resulted the winner of the ComParE challenge, and in [99] are reported. The best UAR score on both the subsets is provided by our proposed algorithm relying on DNN classifiers. The absolute improvement on the state-of-the-art methods on the *devel* subset is remarkable, in fact we obtain in terms of UAR +17.02%. For a fair comparison on the *test* partition we have to consider the score obtained with the best model on the *devel* set. In this case our proposed algorithm overcome the UAR scores of +3.48% and +0.71% with respect to [101] and [99]. These results highlight that the best performance obtained on *devel* set do not provide at the same time the best performance on the *test* set, as it also emerged from the challenge results of reported methods. The motivation of these low generalization proprieties probably relies on the dataset splits composition and their class unbalancing. Anyway, the obtained performances encourage the use of SCAT as snore sound features which results to be effective. In addition, the combination of SCAT and GMS allows reduce dimension of the input vector of the classifier and to obtain an higher classification accuracy with respect to state-of-the-art methods.

Input	**Classifier**	**UAR (%)**	
		devel	***test***
SCAT	MLP	**67.14**	67.71
SCAT	MLP	58.28	**74.19**
Spectrograms + AlexNet fc7	SVM	44.80	67.00
FV + ComParE functionals	(W)KPLS + (W)KELM	50.12	64.23

Table 4.4: Comparative results in terms of UAR (%) score of proposed algorithm with state-of-the-art methods for both *devel* and *test* subsets. Best results on each subset are shown in **bold**.

4.1.5 Conclusion and Outlook

In this section, an approach for snore sound classification based on Deep Scattering Spectrum, Gaussian Mean Supervectors and MLP classifier has been presented. We extracted the SCAT from the audio signals and GMM-based background model was trained to map the sequence of scattering coefficients in a supervector. Classification of input snore sounds has been performed using a MLP neural network and a support vector machine with comparative aims. To assess the performance of the algorithm we conducted experiments on both the *devel* and the *test* subsets of MPSSC dataset. Following the 4-class VOTE scheme, we obtained a UAR of 67.14% on the *devel* set and a respective UAR equal to 67.71% on the *test* set independently of the subject characteristics. The performance upper limit obtained optimizing the models on the *test* set is an UAR up to 74.19%. The obtained results showed that the employment of the DNN based classifier in combination with SCAT is effective and a significant performance improvement with respect to other state-of-the-art approaches was registered. Future work will evaluate strategies of data augmentation to counteract the unbalance of the dataset. In addition, the SCAT representation of the audio signals prompts the exploitation of the 2-D Convolutional neural network (CNN) to obtain a further latent representation by means of the processing taking place in its deep architecture. A deeper focus could be given also to the temporal evolution of the signal by means of recurrent structure, such as Long Short Term Memory (LSTM) Neural Networks [116].

4.2 Road Surface Roughness Classification

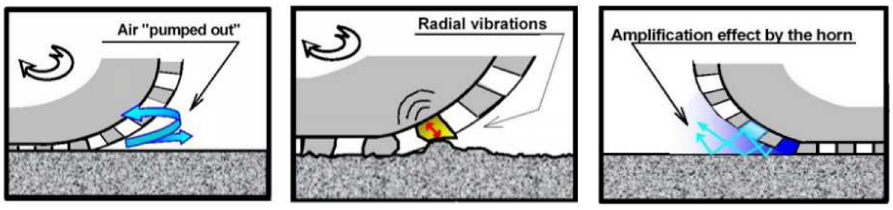


Figure 4.9: The tyre-road noise generation. Left: Air pumping at the entrance of the contact patch; Center: Vibration caused by tread/block pavement impact; Right: The horn effect created by the tyre and road.

This work has been realized in collaboration with the company ASK industries S.P.A., and its aim is to develop an automatic system for the road roughness classification. As first step was conducted an accurate state of the art analysis. Vehicle noise emissions depend on multiple factors, including the power unit noise, aerodynamic noise and tyre-road noise [117]. They are all dependent on speed but have different behaviors. The power unit noise, for instance, depends on the gear engaged and the number of revolutions per minute whereas the tyre-road noise is proportional to the logarithm of the vehicle speed. The balance between these two noise contributions depends, thus, on speed. At low speeds the power unit noise dominates the roadside noise levels, while at high speeds the tyre/road is the predominant source of noise. In [118] it is shown that the tyre-road noise component becomes predominant with modern cars even at speeds above 30 km/h, regardless of the engaged gear.

The tyre-road noise, results from the sum of several acoustic phenomena that concur to the noise emission. The source generation mechanism includes the following elements:

- *Tread Impact*: at the entrance of the interface between tyre and pavement (referred to as contact patch) an impact occurs as the tread hits the pavement. This impact causes vibration of the tyre carcass.
- *Air Pumping*: within the contact patch, the passages and grooves in the tyre are compressed and distorted. The air entrained in this passages will be compressed and pumped in and out of the passages. This rapid exit of air can lead to sound generation.
- *Horn Effect*: sound emitted from the tyre/road contact is reflected multiple times between the road surface and the tyre tread before it propagates further to the receiver. The tyre/road geometry has the shape of a horn.

This horn shape causes a much larger radiation efficiency of the emitted tyre/road noise than in free field. This increased radiation efficiency is referred to as the horn effect.

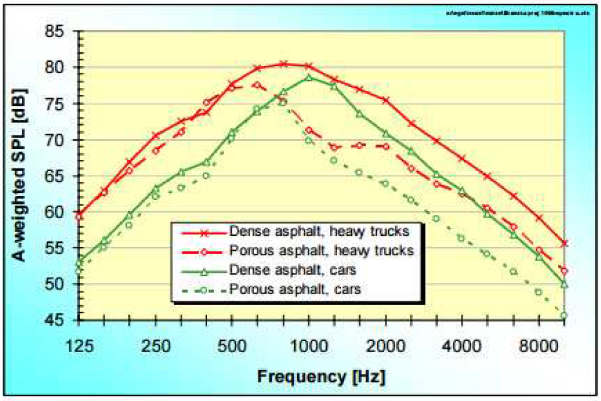


Figure 4.10: Third-octave band spectra of noise measured on a dense asphalt concrete surface and on a porous asphalt concrete surface. Speeds: 80-120 km/h for $>$ 100 cars, 70-90 km/h for $>$ 50 trucks.

Noise spectra have been measured on a large number of tyres (about 50) [119] in a cooperation project between the Technical University of Gdansk (TUG) in Poland and the Swedish National Road and Transport Research Institute (VTI). Despite a wide range of tyre types, the spectral shapes are very similar and the sound concentrates around a peak at 800-1000 Hz. Several studies [120] carried out in roads with different types of surface and age have usually shown that dense asphalt concrete and stone mastic asphalt are the ones that generate more noise contrasting with double and single porous asphalt. Porous pavements are constructed by reducing the amount of small aggregate used in the pavement such that the pavement cannot be tightly compacted. In general, porous pavement reduces tyre/pavement interaction noise above 1000 Hz. Porosity reduces the strength of the air pumping source mechanism by preventing air compression and reduces the enhancement potential of the horn effect as shown in Figure 4.10. Some tests [117] done in Colorado provide a preliminary understanding of the effect of pavement age on noise level. The results shows that, as expected, the older the pavement, the higher the noise level.

In this section, we are interested in inferring the surface roughness by analyzing the near-field acoustic emissions of the vehicle. The surface roughness is characterized by the average size of the gravel base and the presence of filling (asphalt concrete or concrete). These elements have a large impact on the noise emission level and character.

Related Works

A recent paper [121], reports on the use of Support Vector Machines for the goal of classifying several types of road (asphalt, gravel, snow, stony road) using acoustic features. The work employs an electric car for minimal impact of the engine noise on the recording and collects 30 audio fragments from different roads at a fixed speed of 20km/h. The work employs MFCC as they were shown to maximize the Kullback-Leibler distance between all the audio fragments. Classification with the MFCC and an SVM are rather high (between 92.5% and 97.5%), however they decrease to 67%-89% when noise is artificially added to the recordings (such as rain noise or the noise of another car passing by). SVM are employed also in [122], where the main goal, however, is the classification of the road surface wetness and the feature extraction is done with selected 1/3 octave band filters. In that work the recordings are taken in a closed circuit with an internal combustion engine vehicle. A different approach for road dry/wet classification is undertaken in [123] where a dataset is built from recordings of road trips in the US lasting tens of minutes on different types of asphalt in both dry and wet conditions. The work performs classification using a Bi-Directional Long-Short Term Memory (BLSTM) neural network [124] and Auditory Spectral Features (ASF). The paper reports a unweighted average recall of 93.2% as best result, with a large improvement over [122].

These works motivate us to investigate deep learning approaches for the road surface detection problem. In-car equalization profiles may be determined on the basis of the estimated surface roughness in order to improve the listening quality in the cockpit or to obtain suggestions on the driving experience. We propose to extract audio features for the classification of road roughness in two classes corresponding to two extremes. Owing from previous works we build a corpus of data to train an efficient neural network architecture such as a Convolutional Neural Network. The dataset deals with all the issues of a real-world scenario such as the presence of cars passing by, of a combustion engine and the varying speed.

4.2.1 Proposed Method

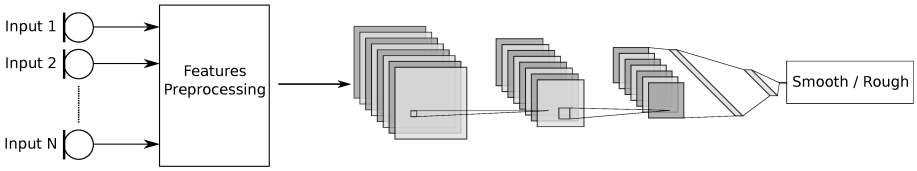


Figure 4.11: Road Surface Roughness Classification - Algorithm Block Diagram

For the road surface classification task we propose an approach based on convolutional neural networks (CNN) [125]. The proposed algorithm is depicted in Figure 4.11. The preprocessing stage works with short audio chunks to detect abrupt changes in the road surface conditions. We use Auditory Spectral Features, that are calculated in the first stage and then arranged in subsequent and non-overlapping chunks of temporal extension equal to one second in the feature extraction stage.

We compared two variants of the algorithm: in the first one, after a supervised training the CNN stage deals with the classification and processes one block at a time; in the second one, we used a Siamese architecture which was trained to produce a measure of "distance" between *smooth* and *rough* samples. In the evaluation phase, we can exploit the pair of inputs to be supplied to the Siamese in order to consider the temporal correlation between adjacent chunks.

We performed experiments also to explore the possibility to realize a *transfer learning* with respect to the tyre types. In detail, we collected the dataset travelling the same road with two sets of tyres, namely *Winter* and *Summer* tyres. Then, we evaluated the performance of the proposed algorithm both using the same tyre set, both training with a set and testing on the other.

Siamese Neural Networks

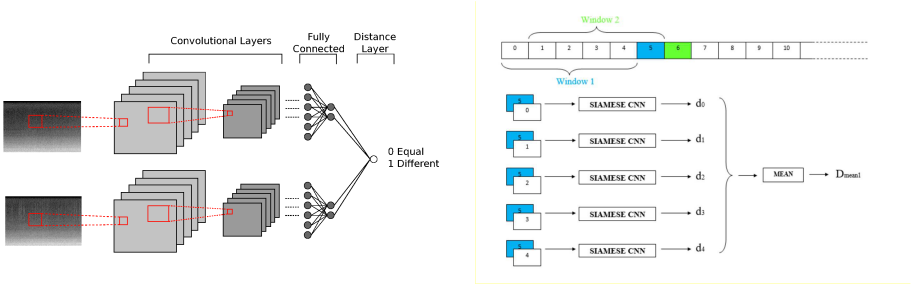


Figure 4.12: Siamese architecture (left) and its prediction procedure (right).

This architecture is designed to accept a pair of audio chunks as inputs which are processed by two CNNs with binded weights. In training phase, the whole network learns to maps these inputs into two different representation vectors, then as output returns the Euclidean distance between them. For further details please refer to (cf. Section 2.3.3).

In this section, we introduce a custom prediction procedure for the Siamese architecture. The idea is shown Figure 4.12: a window composed of n chunks is fixed; in the first case the chunk $\mathbf{X}[n+1]$ is passed iteratively as input to the network together with each of the previous chunks $\mathbf{X}[0,\ldots,n]$. A distance will be associated to each pair, which is the output of the network; then an

average is computed between the n obtained distances. The same is done with the chunk $\mathbf{X}[n+2]$ and the previous n, and so on until the end of test set. In this way every chunk from the n^{th} to the last will be associated to a distance. Analyzing the signal obtained from this procedure we are able to distinguish between *smooth* and *rough* condition. Specifically, the signal obtained from the distances averaging is firstly processed by means of a low-pass filter, then is compared to an adaptive threshold.

Auditory Spectral Features

Auditory Spectral Features (ASF) are acoustic features extracted from audio samples that have been introduced in [126] and are used also in [123]. ASF are computed by applying the Short Time Fourier Transform (STFT) using a frame size of 30 ms and a frame step of 10 ms. Each STFT provides the power spectrogram which is converted to the mel frequency scale using a filter-bank with 26 triangular filters obtaining the mel spectrograms $M_{30}(n, m)$, where n is the frame index, and m is the frequency bin index.

To match the human perception of loudness mel spectrograms are transformed to a logarithmic scale, according to:

$$M_{log}^{30}(n, m) = log(M_{30}(n, m) + 1.0). \tag{4.11}$$

This process yields 26 coefficients, while other 26 are obtained by calculating the positive first order differences $D_{30}(n, m)$ from each logmel spectrogram, as follows:

$$D_{30}(n, m) = M_{log}^{30}(n, m) - M_{log}^{30}(n-1, m), \tag{4.12}$$

for a total of 52 coefficients for frame. The final feature vector is expanded by including the frame energy and its derivative ending up in a total number of 54 coefficients. The features are extracted with an open-source audio analysis toolkit openSMILE v2.3.0 [127] ensuring reproducibility.

In order to feed the CNN, audio chunks of 1 s are obtained from 98 feature vectors, resulting in 2D arrays of 54-by-98 values.

4.2.2 Dataset

The dataset built for this work is done with a multi-channel microphone arrangement, with the prospect of conducting different assessments at once or to exploit microphone diversity to improve the classification. More specifically, two microphones have been placed close to the rear wheels, one in front of the front left wheel, one inside the engine compartment and two inside the cockpit, close to the driver head and close to the right passenger head. The rear wheel microphones have been placed off-axis, in order to avoid dirt from the

wheel and protected by the wheelhouse to reduce the effect of wind. Figure 4.13 shows the positioning of all microphones. External microphones are *PCB Piezotronics* model 130A24. These are IP55 microphones and they have been protected with a melamine resin foam for sound absorption to reduce the effect of wind. The internal microphones are *PCB Piezotronics* model 378C20. The front-right wheel has been excluded from recordings after first informal evaluations because it picked a large amount of engine noise with respect to the other microphones. The rear-right microphone, vice versa, was found to be the best choice because the noise from the engine was the lowest and it has been used for this first evaluation. The engine compartment microphone has been used to record the engine conditions for future use. In Figure 4.14 are shown the images of the microphones installation.

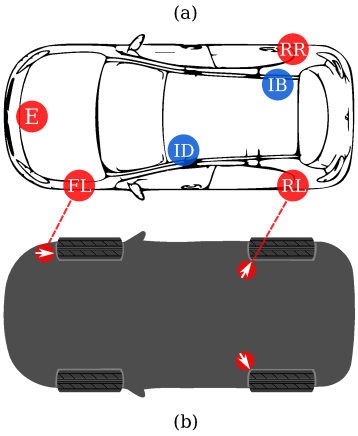


Figure 4.13: Positioning of the microphones in the car used to record the dataset, top view (a) and bottom view (b). The microphones are placed in the engine compartment (E), close to the front-left, rear-left and rear-right tyres (FL, RL, RR), and inside the car close to the driver or in the back seat (ID, IB). The last two microphones are *PCB Piezotronics* model 378C20 type microphones, while all the others are IP55 *PCB Piezotronics* model 130A24 microphones. The microphones are omnidirectional, however the arrows in (b) show how the capsule was positioned to minimize wind effect. The rear microphones are protected in the wheelhouse.

The car employed to build the dataset is Mercedes A Class from 2014. In addition to the audio signals, the GPS signal has been recorded to track down the car speed and position at any given time. A mobile multi-channel front end, *HEAD Acoustics SQuadriga II*, has been employed as acquisition device, being able to monitor and record 8 contemporary channels at different sample rates, and to store GPS antenna and CAN bus signals as depicted in Figure 4.15. All

(a) Rear right tyre microphone. (b) Front left tyre microphone.

Figure 4.14: Pictures of the *PCB Piezotronics* model 130A24 microphones positioned near the rear right and front left tyres, according to "RR" and "FL" red circles in Figure 4.13. The microphones are enclosed by a melamine resin foam with open cell network structure to reduce the wind noise.

audio signals are sampled at 44100 Hz, 24-bits. The external microphones used -26 dBV as input range while the interior microphones had -16 dBV as input range. To facilitate the labelling operations, a camcorder *BC MasterDC10*[3] was installed on the dashboard of the car. In addition to the video, it provides the speed information obtained through its own GPS antenna and it records the cockpit audio, useful for taking vocal notes while driving. The data recorded with the *HEAD Acoustics SQuadriga II* have been exported by means of the software *HEAD Acoustics ArtemiS SUITE* in the uncompressed WAV audio format with a 32-bit float representation.The result of the recording sessions is approximately 100-minutes-long multichannel recordings (6 audio channels and a speed channel) for both the tyre sets. Labels for the roads roughness have been annotated manually.

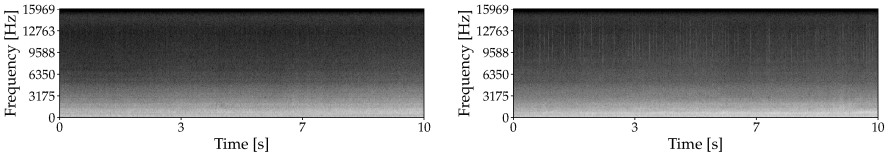
Figure 4.15: Picture of the *HEAD Acoustics SQuadriga II* connected to the GPS-antenna interface.

All recordings were taken in dry conditions in the urban and suburban areas of Ancona (Italy) with variable speed, traffic conditions and pavement rough-

[3] `http://www.bc-master.com/product/car-dash-camera-dc10`

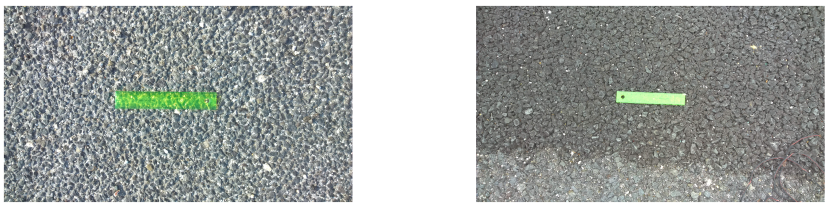
ness. Only roads that had been recently asphalted were considered and multiple takes at different speed for each road have been performed. As aforementioned, the same roads have been travelled in close days with two sets of tyres.

In this section, we considered a subset of the entire dataset, the set "Winter 09-05-2018" and the set "Summer 11-05-2018", respectively composed of approximately 45 minutes of recordings. The two set are not perfectly equal and balanced, although the differences are minimal. The "Winter 09-05-2018" characterized by 51% of rough and 49% of smooth road samples, while the "Summer 11-05-2018" has respectively 56% rough and 44% smooth road samples. The spectrograms from two audio samples belonging to the smooth and rough classes are shown in Figure 4.16, respective asphalt photographs are shown in Figure 4.17.



(a) Smooth road. (b) Rough road.

Figure 4.16: Spectrograms from 10 second samples of (a) smooth urban road, (b) rough highway asphalt.



(a) Smooth road. (b) Rough road.

Figure 4.17: Samples of (a) smooth urban road, (b) rough highway asphalt.

4.2.3 Experiments

Evaluations are reported after a random search of the CNN hyperparameters with different inputs and with both the tyre sets. In Table 4.5 are reported the ranges of the explored hyper-parameters. A 5-fold cross-validation procedure has been performed. The F1-score has been calculated for each combination of training/testing set and then averaged to obtain the unweighted average metrics. The cross-validation has been performed disposing 64% of the dataset to training, 16% to validation and 20% to test. After initial tests, the rear-

right microphone has been selected being the one with the lowest engine noise content. The CNN has been trained using the Adadelta optimization algorithm and binary cross-entropy as a cost function for 1000 epochs, with an early-stopping strategy on the validation set score. The network layouts were the same both for the "traditional" CNN, both for the "basic" network composing the twin architecture in the Siamese approach. In the latter case, the training loss function is the *contrastive loss*, while 1780 and 1916 chunk pairs composed the training sets respectively of the "Winter" and the "Summer" scenarios.

Parameter	Range	Distribution
CNN layers Nr.	[1 - 4]	uniform
CNN kernels Nr.	[4 - 64]	log-uniform
CNN kernels dim.	[3×3 - 6×6]	uniform
Pooling dim.	[1×1 - 2×5]	uniform
CNN activation	[tanh - relu]	random choice
CNN dropout	[0 - 0.5]	uniform
MLP layers Nr.	[1 - 3]	uniform
MLP layers dim.	[20 - 256]	log-unifom
Activation	[$tanh$ - ReLU]	uniform

Table 4.5: Ranges of CNN layout parameters tested in the random-search phase. The kernel size and the stride are expressed as [$time \times features$].

Table 4.6 shows that the Siamese architecture outperforms in all the conditions the traditional CNN approach. In particular, although there is a relative worsening of the Siamese approach when we apply the *transfer learning* in the "Winter" scenario, the absolute performance is significantly better compared to the basic CNN. As shown in Figure 4.18, the combination of Siamese networks and its dedicated post-processing procedure yield very reliable predictions on the experimental dataset. In addition, we note that the models trained with the "Summer" dataset exhibit a poor performance compared to the "Winter" dataset. This is difficult to interpret, but we suppose that be attributed to the tread design of the "Winter" tyres, which enhances the produced noise and the characteristic frequency spectrum of the two road conditions.

TRAINING	TESTING	CNN	Siamese
Winter 09-05-2018	Winter 09-05-2018	85.65	**98.14**
Winter 09-05-2018	Summer 11-05-2018	83.93	**95.08**
Summer 11-05-2018	Summer 11-05-2018	80.65	**93.88**
Summer 11-05-2018	Winter 09-05-2018	76.17	**93.37**

Table 4.6: Results in terms of F-measure (%) for the considered architectures.

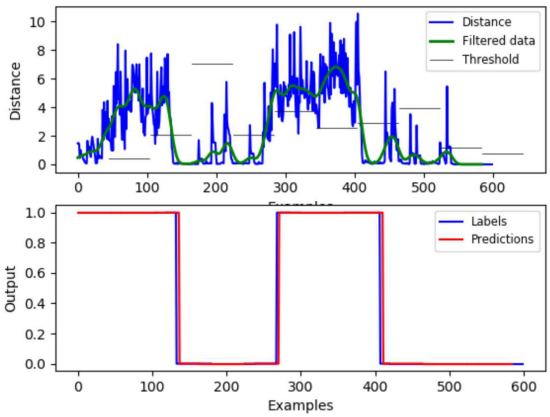


Figure 4.18: Siamese networks output. On top figure, we show the rough signal (blue), the filtered signal (green), and the threshold for each frame (black). On bottom figure, the binarized predictions are compared to the ground truth.

4.2.4 Conclusions

In the present work a data-driven approach for road surface roughness classification is discussed, following seminal works proposed in the last years. The approach is based on a machine listening approach, where the tyre-road noise is analyzed and classified by means of deep neural networks. With respect to previous works a less computational-intensive neural network architecture has been chosen which also allows for on-line processing. A dataset has been created from a real-world scenario by fitting a car microphones and a GPS receiver and driving the car on different roads with varying speed, gathering a multi-channel dataset.

Experimental results are motivating, thus the model based on the Siamese architecture yields an F-measure largely above 90% after a 5-fold cross-validation. In particular, the results of such approach outperform the traditional CNN also when the models are tested on data collected with a different tyre set. This confirms the generalization properties of the method, which allows it to distinguish the acoustical characteristics belonging to the target classes independently from the training set.

Future works may extend the proposed algorithm to process multiple microphone signals, exploiting diversity, thus reducing the number of errors due to cars passing by and other unrelated sound sources. Driving speed, extracted from a GPS receiver or from the CAN bus, may help improve the classification performance. The most challenging objective, however, would be to infer the road roughness from internal microphones, as these are nowadays available in

many vehicles for echo cancellation and noise suppression and are not subject to issues such as dirt, wet, cold, etc. The main issues with this approach, however, are the cockpit isolation from the outside and the interference of speech and music with the road surface noise. Voice activity detection (VAD) [128, 129] algorithms should be employed and extended to detect the presence of music provided by the car infotainment system. Existing works based on CNN architectures [130, 131] could be a starting point as they could be integrated easily with the current framework and extended.

We would like to thank ASK industries S.P.A. for financial support and technical assistance.

4.3 Bird Audio Detection

Automatic wildlife monitoring is a key concern nowadays. Climatic changes, the effects of pollution and alteration of the ecosystems have to be closely controlled in order to be the litmus test for the future sustainable technological and political guidelines. In this context, bird audio analysis is an important task of the bioacoustics for wildlife and biodiversity monitoring, which can easily embrace the deep learning concept. This is confirmed also from the interest received by projects such as "Bird Sounds visualization" supported by the Google Creative Lab [132]. In particular, detecting the presence of bird calls in audio recordings is a very common required first step for a framework that can perform different kind of analysis (e.g. species classification, counting), and makes it possible to conduct work with large datasets (e.g. continuous 24h monitoring) by segmenting the data stream into regions of interests. To encourage the research in automating this task, in 2016 Stowell et al. [133] organized a first edition Bird Audio Detection (BAD) challenge. It has been appreciated to such an extent that a new round has been included in one of the tasks of the 2018 IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE). In fact, Task 3 consists in determining a binary decision for the presence/absence of bird sounds on audio files recorded in very different conditions, comprehending dataset balancing, birds species, background sounds and recordings equipment. Specifically, participants are asked to build algorithms that predict whether a given 10-second recording contains any type of bird vocalization, regardless of the species. Thus, differently from the official name of the task, we can consider it as a *classification* problem. The organizers invite to explore approaches that can either inherently generalize across different conditions (including conditions not seen in the training data), or which can self-adapt to new datasets. The deep neural network based approach we propose has the aim to counteract the generalization problem by means of an innovative learning procedure named "capsule routing" which has shown promising performances since it has been presented [52] and also in pioneering employments in audio tasks [51].

4.3.1 Related Works

In very recent years, a strong growth of deep learning algorithms devoted to the acoustic monitoring has been observed. In particular, works such as [134, 34, 135] represent milestones, involving Convolutional Neural Networks (CNN) for audio signals detection and classification. These deep neural architectures, combined with the increased availability of datasets and computational resources, have allowed large performance improvements, outperforming in most of the cases the human accuracy [43]. This has also motivated

researchers to employ such architecture, eventually combined with recurrent units [45], in almost all of the tasks proposed in the recent editions of research challenges such as the DCASE [136]. These algorithms often result among the strongest-performing systems [47, 137]. Similar results came from the first edition Bird Audio Detection (BAD2017) challenge, which was held in 2016-2017. In this case different novel algorithms have been proposed to create robust and scalable systems able to automate the annotation process of audio sequences containing free-field recordings. The work of Grill and Schlüter [24] should be also mentioned, which obtained the highest score and which is based on CNNs trained on Mel-scaled log-magnitude spectrograms. The outcomes of the BAD2017 are reported in [57].

A team at Google Brain recently has presented a new computational unit [52] called "CapsNet" with the intent to overcome two known limitations of the CNNs: the excessive information loss caused by the pooling and other downscaling operations and the inability to infer part-whole relationships between the elements which the deep neural network (DNN) has to detect. In fact, the layers of a standard CNN are good at detecting space-invariant features which characterize an image (or a spectrogram in the case of audio spectrograms), but are less effective at exploring the spatial relationships among features (perspective, size, orientation). Capsule routing has the aim to learn global coherence implicitly, thereby improving generalization performance. In the BAD application, it means that the DNN is driven to learn a general concept of the entities of "bird song" and "background sounds" without requiring extensive data augmentation or dedicated domain adaptation procedures, thus motivating the use of Capsules for this task.

4.3.2 Proposed Method

The proposed system is a fully data-driven approach based on the CapsNet deep neural architecture presented by Sabour et al. [52]. The novel computational structure of the Capsules, combined to the routing mechanism allows to be invariant to intra-class affine transformations and to identify part-whole relationships between data features. The whole system is composed of a feature extraction stage and a detection stage. The feature extraction stage transforms time-varying audio signal into acoustic spectral features, then the second stage takes the feature vector as input and maps them to a binary estimate of bird song presence. This latter stage is where we introduce the Capsule neural network architecture. The network parameters are obtained by supervised learning using annotations of bird song activity as one hot target vector.

Feature Extraction

The feature extraction stage operates on mono audio signals sampled at 44.1 kHz. For our purpose, we exploit *LogMels* as acoustic spectral representation, following results obtained in various audio tagging and sound event detection tasks. Firstly, the audio signals are down-sampled to 16 kHz, because the most relevant frequency bands related to bird songs are in the range from 2 kHz to 8 kHz [138]. Then, *LogMel* coefficients are obtained by filtering the magnitude spectrum of the STFT with a filter-bank composed of 40 filters evenly spaced in the mel frequency scale. The logarithm of the energy of each band is computed to match the human perception of loudness. In the STFT computation, the used frame size is equal to 40 ms and the frame step is equal to 20 ms. All of the datasets contain 10-second-long WAV files, thus the resulting feature matrix $\mathbf{x} \in \mathbb{R}^{D_1 \times D_2}$ has a shape 501×40. The range of feature values is then normalized according to the mean and the standard deviation computed on the training sets of the neural networks.

CapsNet for Bird Audio Detection

The architecture of the neural network is shown in Fig. 4.19. The first stages of the model are traditional CNN blocks which act as feature extractors on the input *LogMel* coefficients. After each block, max-pooling is used to halve the dimensions. The feature maps obtained by the CNN layers are then fed to the Primary Capsule Layer that represents the lowest level of multi-dimensional entities. Basically it is a convolutional layer whose output is reshaped and squashed using (2.18). The final layer, is a capsule layer and it is composed of two densely connected capsule units. Since the previous layer is also a capsule layer, the dynamic routing algorithm is used to compute the output. The model predictions are obtained computing the the Euclidean length of each output capsule, which represent the probabilities that an input feature vector $\mathbf{x}$ belongs to the background or the bird audio class, thus we consider only the latter as system output prediction.

4.3.3 Experimental Setup

The network hyperparameters optimization was obtained by means of a *random search* strategy [139]. The number and the shape of convolutional layers, the non-linear activation function, the regularizers in addition to the capsules dimensions and the maximum number of routing iterations have been varied for a total of 100 configurations. Details of searched hyperparameters and their ranges are reported in Table 6.6. The neural networks training was accomplished by the AdaDelta stochastic gradient-based optimisation algorithm [140] for a maximum of 100 epochs and batch size equal to 20 on the margin

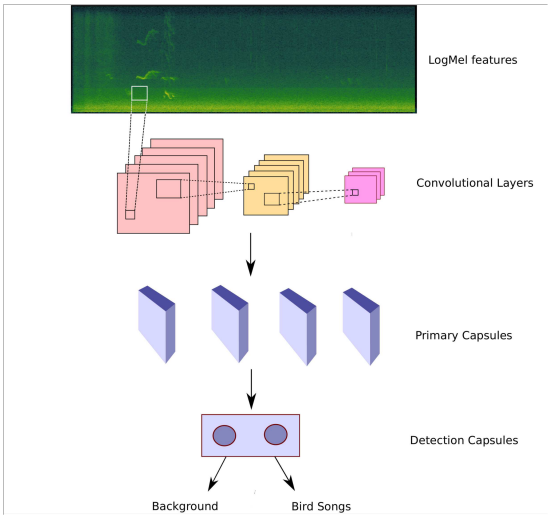


Figure 4.19: Flow chart of the proposed algorithm employing CapsNets for Bird Audio Detection.

loss function. The optimizer hyperparameters were set according to [140] (i.e., initial learning rate $lr = 1.0$, $\rho = 0.95$, $\epsilon = 10^{-6}$). It was chosen because it is well-suited for dealing with sparse data and its robustness to different choices of model hyperparameters. Furthermore no manual tuning of learning rate is required.

An early stopping strategy was employed in order to avoid overfitting. Thus if the validation score does not increase for 20 consecutive epochs, the training is stopped and the last saved model is selected as the final model. In addition, dropout and L2 (with $\lambda = 0.01$) have been used as weights regularization techniques [68]. The algorithm has been implemented in the Python language using Keras [141] and Tensorflow [142] as deep learning libraries.

Dataset

According to the DCASE 2018 guidelines, the performance of the proposed algorithm has been assessed firstly by using the development dataset for training and validation of the system. Then, a blind test on the provided evaluation dataset was performed with the models which achieved the highest performance and submitted to the organizers of the challenge. The complete dataset is composed of recordings belonging to five different collections.

- "freefield1010": a collection of 7690 excerpts from field recordings around the world;

Collection	N. of samples	Balance
Development Dataset		
"warblrb10k"	8000	0.75
"BirdVox-DCASE-20k"	20000	0.5
"freefield1010"	7690	0.25
Total	35690	0.5
Evaluation Dataset		
"warblrb10k_test"	2000	-
"Chernobyl"	6620	-
"PolandNFC"	4000	-
Total	12620	-

Table 4.7: Details of the dataset we used for the algorithm development. The table shows the number of audio files and the ratio between positive/negative samples (if available) of each used data collection.

- "warblrb10k": a crowsourced dataset recorded with the *Warblr*[4] smartphone app. It covers a wide distribution of UK locations and environments and includes weather noise, traffic noise, human speech and even human bird imitations; 8000 samples are used in the development dataset while a held-out set of 2,000 recordings from the same conditions is included in the evaluation split;

- "BirdVox-DCASE-20k": 20000 files containing remote monitoring flight calls collected from recordings units placed near Ithaca, NY, USA during the autumn of 2015;

- "Chernobyl": dataset collected from unattended remote monitoring equipment in the Chernobyl Exclusion Zone (CEZ). A totoal of 6620 audio files cover a range of birds and includes weather, large mammal and insect noise sampled across various CEZ environments, including abandoned village, grassland and forest areas;

- "PolandNFC": 4000 recordings obtained from a project of monitoring of autumn nocturnal bird migration. They were collected every night, from September to November 2016 on the Baltic Sea coast, Poland, using Song Meter SM2 units with microphones mounted on 3–5 m poles.

Further details are reported in Table 4.7. The organizers recommended a 3-way cross-validation (CV) for the algorithms development, thus in each fold we used two sets for training and the other one as validation set in order to have scores comparable with the others challenge participant.

[4]https://www.warblr.co.uk/

Parameter	Range	Distribution	CapsNet1	CapsNet 2	CapsNet3
CNN layers Nr.	[1 - 4]	uniform	3	4	4
CNN kernels Nr.	[4 - 64]	log-uniform	[64,16,8]	[32,16,16,32]	[32,64,4,64]
CNN kernels dim.	[3×3 - 8×8]	uniform	3×3	5×5	6×6
Pooling dim.	[1×1 - 2×5]	uniform	[1×5],[1×4], [1×4]	[1×5],[1×4], [1×2],[1×2]	[1×4],[1×2], [1×2],[1×2]
CNN activation	[tanh - relu]	random choice	tanh	relu	relu
CNN dropout	[0 - 0.5]	uniform	0	0	0
CNN L2	[yes - no]	random choice	no	yes	yes
Primary Capsules channels Nr.	[2 - 8]	uniform	6	2	8
Primary Capsules kernels dim.	[3×3 - 5×5]	uniform	4×4	4×4	3×3
Primary Capsules dimension	[2 - 16]	uniform	8	8	2
Capsules dimension	[2 - 16]	uniform	2	15	10
Capsules dropout	[0 - 0.5]	uniform	0	0.1	0.3
Max routing iterations	[1 - 5]	uniform	2	3	2
Batch Normalization	[yes - no]	random choice	yes	yes	yes
Trainable Params	-	-	113k	282k	424k

Table 4.8: Hyper-parameters optimized in the random-search phase and the resulting best performing models.

Baseline

The baseline system is an adapted version of the method winner of the BAD2017 [24]. The peculiarity of this algorithm is its double training procedure. In a first run, the network is trained on the whole training data. Binary predictions are obtained for the testing data. The more confident predictions (the ones closer to 0 or 1) are then added to the training data as so-called "pseudo-labeled" samples. Thus, a second training run is performed on this extended training set and the final predictions are yielded.

Metric

The performance metric of the DCASE 2018 on this task is the "Area Under the ROC Curve" (AUC). More precisely, it is a stratified AUC: the score is computed separately for each fold of the evaluation set, then the partial scores are averaged. This procedure allows to adapt the "detection threshold" to each dataset conditions, then the performance across datasets are combined in an explicit weighted fashion, thus the final score is not merely influenced by the number of files in each subset.

4.3.4 Results

Results reported in Table 4.9 show both the best performance we obtained on the single CV fold, and the best averaged AUC. We obtain a harmonic mean for AUC equal to 83.72 for a single configuration, whilst if we consider the mean of the best performing models on the single folds we achieve an AUC equal to 85.08.

Conf ID	Fold 1	Fold 2	Fold 3	Avg	Evaluation
Baseline	79.90	88.20	88.20	85.40	88.50
CapNet1	**88.22**	72.78	74.16	78.39	-
CapNet2	81.77	**80.90**	85.52	82.73	-
CapNet3	86.59	78.46	**86.11**	83.72	75.40
Ensemble	-	-	-	**85.08**	**78.80**

Table 4.9: Results on Development dataset in terms of AUC (%).

We considered as candidates for the test on the evaluation procedure [57] both the best performing setups on the single CV folds, and the setups with the best averaged AUC. For the latter, we trained a new model with the same hyperparameters on the whole development dataset before performing the predictions on the evaluation dataset. With an ensemble of the single fold best models trained during the CV procedure we obtain an AUC score equal to 78.80, while for the model with the best averaged AUC we obtain an AUC score equal to 75.40.

Although the system obtained only the 12th place in the final ranking, the performance on challenge data were respectable considering the novelty of the approach. This allowed us to earn the judges award for the most innovative method at the DCASE 2018 Workshop.

4.3.5 Conclusion and Outlook

In this section, we have presented an algorithm for bird audio detection based on the CapsNet architecture. We feed a deep neural network which uses the dynamic routing procedure during the training with the *LogMel* extracted from the audio signals in order to obtain predictions on unseen data recorded in various conditions possibly also very different from the training set. For future work, variants [143] or strategy to customize the dynamic routing can be considered.

Chapter 5

Sound Event Detection

The task of sound event detection (SED) is defined as the task of analysing a continuous audio stream in order to extract a description of the sound events occurring in it. This description is commonly expressed as a label that marks the start, the ending, and the nature of the occurred sound (e.g., children crying, cutlery, glass jingling). Thus, in this case we don't produce a "global" prediction of a short excerpt of audio, but differently from the SEC, the systems yield an activation representative of the targe sound event presence each time frame (i.e., at the maximum time resolution required by the application).

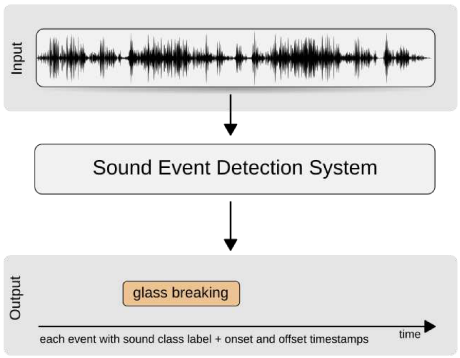


Figure 5.1: System input and output for the sound event detection.

In this chapter three related works are shown, which are respectively: Overnight Snore Sounds Detection by means of Convolutional Recurrent Neural Networks (CRNN) and acoustic data augmentation; Convolutional Neural Networks with 3-D Kernels for Voice Activity Detection in a Multiroom Environment; and Rare Sound Detection, consisting in detection of three sound categories (i.e., gunshot, babycry and glassbreak) from a highly imbalanced dataset. The latter has been performed by means of a hierarchical framework of ConvNets.

5.1 Overnight Snore Sounds Detection

As described in Section 4.1, one of the most common sleep disorder is the chronic snoring. In the occasion of the Interspeech 2017 ComParE challenge [98] and subsequent investigations, different approaches based on Deep Neural Networks (DNNs) have been presented [99], [100], [144] with the aim to classify isolated snore sound events among the four classes based of the VOTE scheme.

In this section, we propose the application of Convolutional Recurrent Neural Networks (CRNNs) to detect snoring episodes from overnight recordings acquired in real life conditions. The method is a two step process: the acoustic spectral features extraction and the CRNN combined with Gated Recurrent Units (GRU) processing. The algorithm is evaluated by using the Average Precision (AP) score. Differently from other deep learning approaches [99], [100], [144], this choice offers a viable and natural solution for jointly learning the spatio-temporal dependencies of audio sequence for discovering snoring event. Additionally, we deal with the high unbalanced setting which exists in the task of snore detection. Thus, one of the main goals of this section is also to evaluate the performance of the overnight snore sound detection for different data augmentation techniques. The original snore/background ratio in the aforementioned signals has been increased by adding isolated snore events from the Munich-Passau Snore Sound Corpus dataset [98] (cf. Section 4.1.3). The reliability of the proposed approach is investigated using the A3-snore dataset, leading to significant improvement in term of Average Precision (AP) with respect to Convolutional Neural Network (up to 9.48%) and other data augmentation techniques.

5.1.1 Proposed Approach

The two-step of the proposed approach are detailed in this section, starting from the spectral features extraction and ending with the Convolutional Recurrent Neural Network. Figure 5.2 shows the overall architecture.

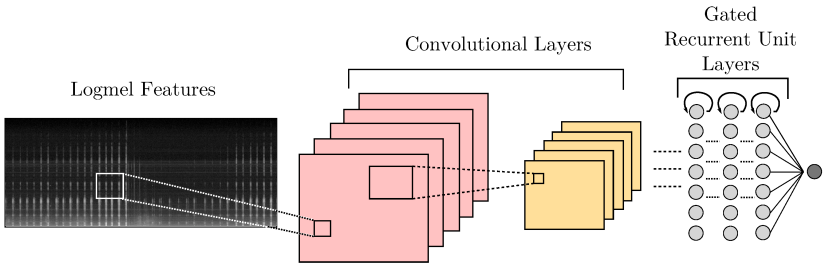


Figure 5.2: The proposed approach scheme for Snoring Detection.

Features Extraction

The feature extraction stage operates on stereo audio signals sampled at 44.1 kHz. Following the results obtained in recent works related to sound event detection [46], we use the log Mel energy coefficients (Logmel) as an efficient representation of the audio signal. The stereo signal is firstly down-mixed to mono by averaging the two channels. The resulting audio signal is split into 30 ms frames and a frame step of 10 ms to compute the STFT spectrogram. We used a filter bank with 40 mel scaled channels, obtaining 40 coefficients/frame.

Convolutional Recurrent Neural Networks

CRNNs used in this section are composed of four types of layers: convolutional layers, pooling layers, recurrent layers and detection layer. Each convolutional layer is followed by batch normalization per feature map [69], a leaky rectified linear unit activation function (LeakyReLU) and a dropout layer [68] with rate equal to 0.3. A frequency domain max-pooling layer is then applied to the resulting feature-map, in order to enhance the relevant information from frequency bands without losing the temporal resolution of the Logmels, as proposed in [48]. The extracted features over the CNN feature maps are stacked along the frequency axis. Max-Pooling operation combined with shared weight in convolutional layers provide robustness to frequency shifts in the input features and this is crucial to overcome the problem of intra-class acoustic variability for snore events. In the recurrent block, the stacked features resulting from the last pooling layer are fed to layers composed of GRUs (cf. Section 2.3.5), where tanh and hard sigmoid activation functions are used for update and reset gates, respectively. Fast response to the changes in the input and the previous activation information is fundamental for high performance in the proposed algorithm, where the task is to detect a small chunk of consecutive time frames where the target event is present. In addition, a previous work [145] demonstrates improvements provided by recurrent architectures in the sound event detection in real-life audio. The detection layer is a feed-forward layer of composed of a single neuron with sigmoid activation function, corresponding to the probability the event onset. The layer is time distributed, this means that while computing the output of the classification layer, the same weight and bias values are used over the recurrent layer outputs for each frame.

In a comparative aim, we implemented also a CNN architecture very similar to the CRNN, the only difference being that the recurrent layers of the CRNN are replaced with time distributed feed-forward layers with ReLU activations. In following section, we will refer it as CNN.

The neural networks training was accomplished by the AdaDelta stochastic gradient based optimisation algorithm [140] for a maximum of 500 epochs on

the binary cross entropy loss function. The optimizer hyperparameters were set according to [140] (i.e., initial learning rate $lr = 1.0$, $\rho = 0.95$, $\epsilon = 10^{-6}$). An early stopping strategy, monitoring the validation AP score, was employed in order to reduce the computational burden and avoid overfitting.

5.1.2 A3 Snore Dataset

The snore detection algorithm has been evaluated on the A3-Snore dataset. A brief description of the acquisition setup and dataset splitting is provided in the following.

Acquisition setup:

In order to capture the overnight audio recordings a ZOOM-H1 Handy Recorder has been used. It is equipped with two unidirectional microphones set at a 90 degree angle relative to one another. The signals are stored in WAV files with a sampling rate of 44.1 kHz and bit depth equal to 16. The input gain is automatically set by the recorder to prevent overload and distortion, while the high-pass filter was enabled in order to eliminate pops, wind noise, blowing, and other kinds of low frequency rumble.

Acquisition environment:

The acquisition environment consists of a simple bedroom, with two access points (door and window). The recorder is placed near the patient, at same height of the bed and in line with the subject's mouth. During the recordings, the patient is the only one that can occupy the bedroom, in order to avoid contaminations on recorded audio signals. The room dimensions are reported in Figure 5.3. Background sounds include traffic noise, breathing and speech signals, house and animal noises. We acquired some samples measurements of the event-to-background (EBR) ratios considering background noise, snoring events and noise events such as "car passing by" or "dog barking". The EBR resulted equal to 6.5 dB and 1.1 dB respectively for noise to background EBR and snore to background EBR.

Dataset splitting:

The original recordings have been manually labelled, annotating the snore events onset and offset with a resolution of 1 second. The audio sequences have been divided into chunks of 10 minutes, and only those with the highest number of snore events have been used in the experiments. The dataset is organized into subjects, which can be respectively used as *training* or *validation*

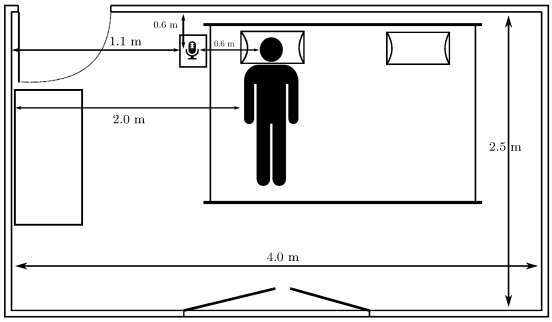


Figure 5.3: Plant of the recording room.

sets in a two fold cross validation strategy (i.e., Leave One Subject Out procedure). The number of events per class in the database is strongly unbalanced as reported in Table 5.1. Thus, the snore detection task is challenging, due to the high number of noises on the A3-SNORE dataset.

A3-SNORE dataset					
#	Gender	Age	Snoring (SN)	Total Duration (Tot)	Ratio (SN/Tot)
Snorer 1	M	48	33m-27s	3h-12m-0s	14.5%
Snorer 2	M	55	21m-21s	3h-50m-0s	11.1%
Total			54m-48s	7h-02m-0s	12.8%

Table 5.1: Difference of recording times for each class, divided by snorers.

5.1.3 Data Augmentation Techniques

In this application, what we are really interested is to detect the minority class (e.g. snoring events) rather than the majority class (e.g. background). Thus, we need to adequately train the models in order to obtain a fairly high prediction for the minority class. In order to counteract the dataset unbalancing existing in the task of snoring detection different techniques of data augmentation have been evaluated. The literature suggests that it is possible to augment training data in data-space or in feature-space. In this section, both data augmentation approach have been evaluated, by using the *Synthetic Minority Over-sampling Technique* (SMOTE) [80] in the feature space, and by generating simulated data with an increased number of snore events. The original snore/background ratio in the acquired signals has been increased with these transformations to approximately 30% [85], maintaining anyway a natural unbalance which is properly of this task. In the following sub-sections, a brief description of each method is provided.

Majority Class under sampling:

it is not a properly data augmentation technique but it is a fast and easy way to balance the data. It consists in randomly selecting a subset of data from the training sets in order to modify the ratio of the sample occurrences in two classes.

SMOTE:

It is an over-sampling approach in which the minority class is over-sampled by creating new synthetic examples. The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors (k-NN). Depending upon the amount of required over-sampling, neighbors from the k-NNs are randomly chosen. In particular, synthetic samples are generated in the following way: the difference between the feature vector (sample) under consideration and its nearest neighbor is multiplied by a random number between 0 and 1, and this is added to the feature vector under consideration. In details, for a sample x_i:

$$x_j^{\text{SMOTE}} = x_i + (\tilde{x}_{i,k} - x_i) \cdot r(j) \tag{5.1}$$

where $r(j) \in [0, 1]$. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general.

Proposed approach - Generating simulated data:

The simulated training sets have been created starting from the folds described in Section 4.3.3. The impulse responses between the snore source and the microphones have been recreated by using the library Pyroomacoustics [146]. Isolated snore sounds have been taken from the Munich-Passau Snore Sound Corpus (MPSSC) dataset [98]. It is composed of 843 snore events which have been extracted and manually screened by medical experts from Drug-Induced Sleep Endoscopy (DISE) examinations of 224 subjects. The augmented training set has been created by convolving the isolated snore sound events of the MPSSC corpus with the synthetic impulse responses. Than, the obtained signals have been mixed with the original recordings without overlap with the already present events. The artificial added event dynamic was normalized to the maximum value observed in the original signals. The resulting total time of snore signals is 55 minutes for Snorer 1, and 56 minutes and 5 seconds for Snorer 2.

5.1.4 Experimental Setup

The performance of the algorithms has been evaluated in term of AP score, a metric that summarizes the Precision and Recall curve (cf. Section 3.3). To assess the performance of the models, we explored different hyper-parameter configurations and, for each of these, we repeated the whole experiments training the models both with the original data and with data processed with techniques described in Section 5.1.3. Table 5.2 shows the hyper-parameter configurations analyzed in our experiments. They regard kernels size, kernel number and GRUs for a total of 120 experiments. In the case of CNN the number of units and layer refers to a Multi Layer Perceptron (MLP) architecture. The experiments were conducted in a 2-fold cross-validation strategy corresponding on a leave one subject out procedure, thus in fold 1 we used Snorer 1 as training set and Snorer 2 as validation set and in fold 2 vice-versa. The models were selected on the performance based on the AP score averaged on the two folds. The algorithm has been implemented in the Python language using Keras [141] as deep learning library. All the experiments were performed on a computer equipped with a 6-core Intel i7, 32 GB of RAM and two Nvidia Titan X graphic cards.

Convolutional Layers Number	3
Kernel Number	4, 8, 16, 32, 64
Kernel Size	5×5, 3×3, 2×2
Pooling Size	5×1, 4×1, 2×1
Recurrent Layers Number	2, 3
Dense Layers Number	2, 3
Number Of Units	4, 8, 16, 32, 64

Table 5.2: Explored network layout parameters.

5.1.5 Results

The performance of the CRNN and CNN architectures using different data augmentation techniques are reported in Figure 5.4. In blue are depicted results with CRNN, in green the results of the CNNs. The CRNNs show to be effective for snore event detection yet with the original data, although the dataset imbalance. The best performing model is composed of 3 CNN layers with respectively [64,64,64] filters of size 3×3 and two GRU layers of 64 units. This configuration obtains an AP up to 82.05%, with a difference of +7.79% with respect to the CNN.

The majority class under-sampling and the SMOTE techniques obtain worst performance with respect to original recordings. For majority class under sam-

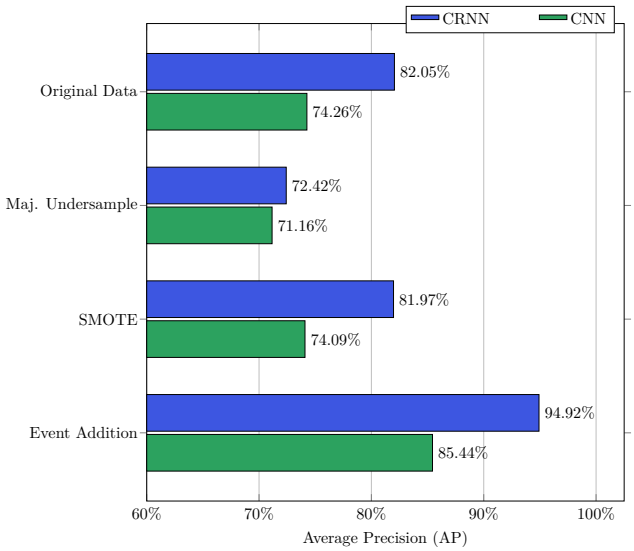


Figure 5.4: Results with different data augmentation techniques for the best models of the evaluated architectures.

pling this can be motivated by the necessity of DNN models of a large amount of data to be trained properly, thus a reduction of data samples cannot benefits to their detection ability. Regarding the SMOTE, the performance reduction is less dramatic (-0.16% and -0.08%, respectively, for CNNs and CRNNs) but its employment remains vain. In this case, the motivation can be found in the complexity to generate new samples of audio signals in the feature space which can really improve a DNN perfomance.

The addition of isolated snore samples convolved with the simulated room impulse response has a tangible beneficial effect on the examined models. In fact, with this technique we obtain an AP improvement equal to 11.18% and 12.87%, respectively, for the CNN and the CRNN. The latter obtains an AP equal to 94.92% with an architecture composed of 3 CNN layers with respectively [64,32,32] filters of size 5×5 and two GRU layers of 32 units. This model is composed of 91,553 free-parameters and occupies approximately 1.2 MB, providing to the algorithm a feasible complexity in an application scenario.

5.1.6 Conclusion

In this section, a deep learning algorithm based on a CRNN architecture fed with Logmel spectral features extracted from the audio signal has been proposed for snore detection. The A3-Snore dataset has been acquired in real-world conditions, containing overnight recordings of two male subjects

and it has been used to assess the performance of the models. The original snore/background ratio has been increased by adding isolated snore events from the Munich-Passau Snore Sound Corpus dataset [98]. The reliability of the proposed approach has been investigated with respect to baseline CNN and different data augmentation techniques such as oversampling (i.e., SMOTE) and downsampling. Results show that the presented snore detection methodology is able to better generalize across different users. In particular, the CRNN is able to extract salient information from the spectral features in order to discriminate snore events, while the implemented data augmentation provides additional samples of the minority class (i.e., snore events). These samples contain supplementary information that can be exploited by the CRNN for learning and discriminate snore events. Future works will be addressed to employ this methodology in a weakly supervised setting. Specifically, in the real-life applications, the precise annotation of existing events from overnight recordings can be onerous and can be result in sparse labeling. Machine learning models trained in a weakly supervised fashion can help to counteract this problem without losing the state of the art performance.

5.2 Rare Sound Event Detection

The "Detection of rare sound events" task of the 2017 Detection and Classification of Acoustic Scenes and Events (DCASE) challenge [46] consisted in determining the presence and the precise onset time of three types of sounds, "baby cry", "glass break" and "gun shot" in artificially generated audio sequences. In detail, the adjective "rare" refers to the dataset unbalance, thus the disproportion between the total duration of the target sounds and the total amount of the recorded data. The task takes into account real-world issues that introduce additional complexity to the problem, such as the acoustic variability of the sounds belonging to each event class, the presence of environmental noise and its variability, etc. The rules of the challenge allow to know *a priori* the event typology possibly present in the audio sequence under examination, thus it is possible to have a separate binary classifier for each class.

5.2.1 Related Works

In the recent era of the "Deep Learning" different approaches to SED have been proposed marking use of the capabilities of deep neural networks (DNNs) to learn the relation between time-frequency features of the raw audio signal and a target vector representing sound events. Although the DNNs based systems are more computationally intensive with respect to widely used statistical modelling methods such as hidden Markov models (HMMs) or Gaussian mixture models (GMMs) [5, 9], a comparative study [38] has highlighted that they are able to achieve top performance in the sound recognition problem.

A well-fitting example of such performance is given in [42], where different DNNs are trained on three datasets recorded in real life environments in order to detect abnormal events or hazardous situations exploiting only the information carried by the acoustic signal. The experimental results show that Deep Recurrent Neural Networks (DRNNs) outperform the probabilistic approaches over the three databases. Another example focuses on employing Convolutional Neural Networks (CNN) for Voice Activity Detection in multi-room domestic scenarios (mVAD) [130]. The CNN-mVAD results to be effective and outperforms the other method with a significant solidity in terms of performance statistics.

In occasion of the DCASE 2017 challenge, many novel systems featuring deep neural networks have been proposed, in particular involving hybrid architectures making use of Convolutional Neural Networks (CNN) and DRNNs. In detail, both the first two classified algorithms make use of mel spectrogram coefficients as spectral representation of the audio signal which is processed by a CNN with 1D filters in the case of the first ranked [47] or by a 2D CNN with frequency pooling in the case of the second classified [48]. The architectures

are, then, combined with recurrent layers to process the features obtained by the convolutional blocks. In [49] the authors propose a hierarchical structure based on CNNs and DNNs trained with multi-task loss functions. Specifically, in the first stage the networks are trained for background noise rejection, using a weighted loss function to penalize the false positive errors. In the second stage the multi-task loss enables the networks to simultaneously perform the event classification task and the onset time estimation. This approach obtained the third place in the final ranking. All of the aforementioned systems largely outperform the baseline system based on a Multi Layer Perceptron architecture (MLP) and Logmel energies as features. The baseline system [136] is based on a Multi Layer Perceptron architecture (MLP) and log mel energies as features. For each audio frame, the input vector is constructed concatenating 5 adjacent log mel vectors for a total of 200 elements. The ANN architecture consists of two dense layers of 50 hidden units each and one output neuron with sigmoid activation, which indicates the activity of the target class.

5.2.2 Proposed Method

The proposed system is a hierarchical algorithm composed of five stages: the acoustic features extraction, the event detection stage 1, which produces an output at frame-rate and a dedicated smoothing procedure of this signal. Then, a refinement of the previous decision stage is performed by a 2D CNN which discards possible false positives detected by the stage 1. The final decision procedure annotates the effective onset time of the active event. In Figure 5.5 the phases of the algorithm are depicted. This is an extended and improved method with respect to our contribution to the DCASE 2017 [147].

Features Extraction

The feature extraction stage operates on mono audio signals sampled at 44.1 kHz. Following the results obtained at the DCASE2017 challenge by [48], we use the log mel energy coefficients (Logmel) as an efficient representation of the audio signal. In addition, we explored the combination of the Logmel with features based on wavelet coefficients and forward prediction errors (WC-LPE) [76]. A brief description of the features extraction procedures is given below.

Logmel coefficients The audio signal is split into frames of 40 ms and a frame step of 20 ms, then the Logmel coefficients are obtained by filtering the power spectrogram of the frame as described in Section 3.1.3. In this section, we used a filter bank with 40 mel scaled channels, obtaining 40 coefficients/frame.

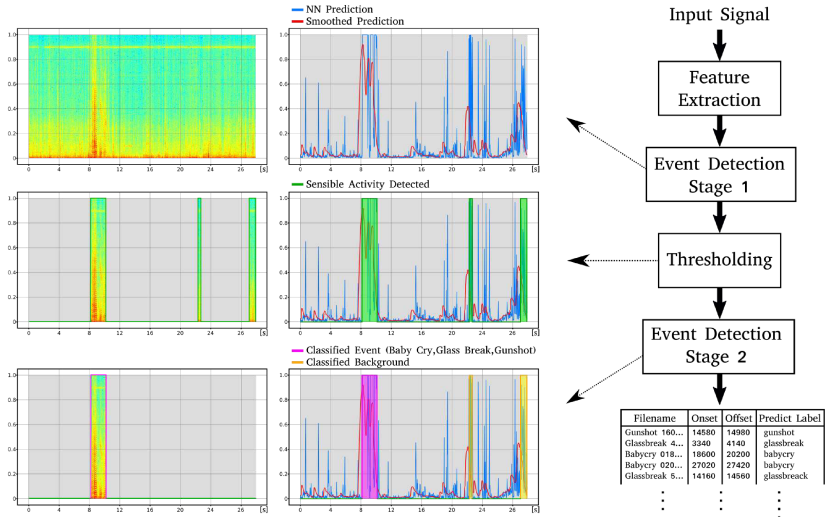


Figure 5.5: Flow chart of the proposed method for rare sound event detection. Each event class implements such a scheme. In the first column are shown the spectrograms of the input signal and of the detected events. In the second column the network outputs at each stage of the algorithm.

WC-LPE Feature The Wavelet Coefficient (WC) and Linear Prediction Error (LPE) feature set relies on non-stationary signal components and it has been successfully exploited for musical note onset detection [76]. WC-LPE extraction is obtained by first processing the input signal with a Discrete Wavelet Transform (DWT) dyadic tree. Then, each DWT sub-band is filtered by a linear prediction error filter (LPEF), obtaining Forward Prediction Errors (FPE). All LPEF outputs and DWT sub-bands are resampled to an intermediate sampling rate and rectified. The feature set is, finally, created from the DWT sub-bands, their first order time derivatives, the FPE and their first order time derivatives.

For both feature sets the range values of each coefficient is normalized independently according to the mean and the standard deviation computed on the training sets of the neural networks.

Event Detection Stage 1

The Event detection (ED) stage 1 has the goal to discard frames containing only background sounds, reducing as much as possible the false negative decisions. We evaluated two DNN architectures as binary classifiers: the MLP and the CNN with 2D kernels and frequency pooling. In both cases, the output layer is formed by two units with the *softmax* non-linear function. Thus, the networks

outputs represent the probabilities that an input feature vector $\mathbf{x}[t]$ at the frame index t belongs to the background or the event class. In our analysis, we evaluated as network input the Logmel coefficients and the combination of the latter with the WC-LPE features.

Deep Neural Network Architectures

For the ED stage 1 we compared the performance of two deep neural networks architectures, respectively the Multi Layer Perceptron (MLP) and the Convolutional Neural Networks (CNN). In both cases, the neural networks training was accomplished by the AdaDelta stochastic gradient-based optimisation algorithm [140] for a maximum of 500 epochs on the binary cross entropy loss function. The optimizer hyperparameters were set according to [140] (i.e., initial learning rate $lr = 1.0$, $\rho = 0.95$, $\epsilon = 10^{-6}$). An early stopping strategy monitoring the validation loss was employed in order to reduce the computational burden. Thus if the validation loss does not decrease for 20 consecutive epochs, the training is stopped and the last saved model is selected as the final model. In addition, dropout is used as regularization technique [68] with rate 0.5.

Multi Layer Perceptron Neural Network The network is designed to consider a temporal context C, thus the network input feature vector $\hat{\mathbf{x}}[t]$ is obtained concatenating $\mathbf{x}[t]$ with the previous $\mathbf{x}[t-c]$, with $c = 1, \ldots, C$.

During the training procedure, additive zero-centered Gaussian noise with $\sigma = 0.1$ was applied to $\hat{\mathbf{x}}[t]$ as a form of data augmentation, improving the generalization capabilities of the DNN and avoiding overfitting [42].

Convolutional Neural Network In our case the convolutional layer input is a matrix $\mathbf{X} \in \mathbb{R}^{F \times T}$, where F and T represent respectively and the number of Logmel channels and the number of frames of the acoustic signal. When we combine the two aforementioned feature sets, we process them with two separate sets of convolutional layers, gathering two feature maps that are concatenated along the feature axis. Before concatenation, batch normalization [69] is applied to each feature map and a leaky rectified linear unit activation function (LeakyReLU) with $\alpha = 0.3$, followed by a feature domain max-pooling layer. Finally, fully connected layers are stacked, applying the same weights and biases to each frame element. The output layer for each of the binary classifier neural networks has two neurons corresponding to the probability of the background or the event onset. We can discard, thus, one of the two neurons without loss of information, and we will consider the output of the neuron corresponding to the event activation $u[t] = y_{t,2}$, as the output of the network at frame t.

Post Processing

In the post processing stage, each network output is convolved with an exponential decay window of length M defined as:

$$\mathbf{w}[t] = e^{-\frac{t}{\tau}} \quad \text{with}\, \tau = \frac{-(M-1)}{log_e(0.01)} \tag{5.2}$$

The result is processed with a sliding median filter with local window-size k. Finally, a decision threshold θ is applied.

Event Detection Stage 2

The aim of the event detection stage 2 is to eliminate false positives, by removing the events wrongly detected at the previous stage. This is done by feeding a binary-classifier CNN with chunks of features in correspondence to the detected events (colored region in the bottom right spectrogram of Figure 5.5). At this stage only Logmel coefficients are used as input features, in order to reduce the computational burden of the model. Non-overlapping feature matrices $\mathbf{X}$ of size $F \times 20$ are used during training, while 95%-overlapping feature matrices are employed during testing (1-frame shift). A chunk size of 20 corresponds to 0.4 seconds of audio, i.e. half the minimum possible length of the occurring events, leading to an analysis of the audio event at different time and frequency resolutions with respect to previous stages. The ED Stage 2 NN is trained for 100 epochs on the binary cross entropy loss function with the AdaDelta gradient descent algorithm.

Final Decision

For each audio sequence, we perform a classification on contiguous blocks of frames detected as event by the ED stage 1. Among contiguous frame chunks classified as "event" by the CNN, the first frame with highest network output is indicated as event onset.

5.2.3 Experimental Setup

According to the DCASE 2017 guidelines, the performance of the proposed algorithm has been assessed by using the development dataset for training and validation of the system. Furthermore, a blind test on the provided evaluation dataset has been performed. The performance metric of the DCASE 2017 challenge is the event-based error rate (ER) calculated using onset-only condition with a collar of 500 ms. The algorithm has been implemented in the Python language using Keras [141] as deep learning library. All the experiments were

performed on a computer equipped with a 6-core Intel i7, 32 GB of RAM and two Nvidia Titan X graphic cards.

Dataset

The DCASE2017 challenge dataset [136] has been used to develop and evaluate the algorithm. The dataset consists of 30-second long sequences of background acoustic scenes recorded in different public or domestic spaces (park, home, street, cafe, train etc.)[56], some of which have been added with isolated recordings from at most one of the three different target sound event classes: baby crying, glass breaking and gun shot. The presence probability of a sound event in each mixed sequence of the original Development set was 0.5, thus we kept only sequences containing a sound event of the original training set and we generated additional mixtures assigned to the training and the validation sets. For the development set a total number of sequences respectively equal to 2750 for training, 300 for validation and 1496 for test have been employed. This change increases the percentage of the frames including a target event in the training data, which helps to ease the problem of data imbalance. In addition, due to the fast decay of the "gun shot" sound, we generated more sequences containing this event class compared to the others, in order to maintain approximately the same percentage between frames containing event samples and backgrounds.

In the evaluation set, the training and test sequences of the development set are combined into a single training set, while the validation set is the same used in the Development dataset. The system is evaluated against an "unseen" set of 1500 samples (500 for each target class) with a sound event presence probability for each class equal to 0.5.

First Event Detection Stage

To assess the performance of the MLP employed in the event detection stage 1 we resorted to a random search strategy [139]. Table 5.3 shows the parameters explored in the random search, as well as the prior distribution and ranges. We evaluated 300 sets of layout parameters (100 for each event class) repeated for

Parameter	Range	Distribution
MLP layers Nr.	[2 - 7]	uniform
MLP layers dim.	[20 - 4048]	log-unifom
MLP Context	[1 - 7]	uniform
Activation	[tanh - relu]	uniform

Table 5.3: Hyper-parameters optimized in the random-search phase for the MLP ED stage 1, and their range.

	Development Dataset				Evaluation Dataset			
Features	Babycry	Glassbreak	Gunshot	**Average**	Babycry	Glassbreak	Gunshot	**Average**
MLP ED Stage 1								
Logmel	0.19	0.12	0.16	**0.16**	0.64	0.54	0.58	0.59
Logmel + WC-LPE	0.23	0.10	0.19	0.17	0.76	0.55	0.55	0.62
CNN ED Stage 1								
Logmel	0.23	0.13	0.18	0.18	0.48	0.23	0.44	0.38
Logmel + WC-LPE	0.25	0.09	0.16	0.17	0.46	0.10	0.36	**0.31**
MLP ED Stage 1 + CNN ED Stage 2								
Logmel	0.14	0.08	0.16	**0.13**	0.31	0.25	0.44	0.33
Logmel + WC-LPE	0.20	0.09	0.19	0.16	0.37	0.27	0.40	0.35
CNN ED Stage 1 + CNN ED Stage 2								
Logmel	0.19	0.10	0.16	0.15	0.31	0.17	0.39	0.29
Logmel + WC-LPE	0.18	0.08	0.17	0.14	0.25	0.10	0.31	**0.22**

Table 5.4: Results in terms of ER score for all the evaluated combination of proposed ANNs and features used in Event Detection Stage 1.

the two input features combination.

Regarding the CNNs, we explored the hyper-parameters space by means of a grid search for a total of 75 experiments (25 for each event class) covering the number of convolutional filters per layer $\{16, 32, 64\}$, the kernels shape $\{3 \times 3, 5 \times 5\}$, the number of MLP layers $\{1, 2, 3\}$ and their respective number of units $\{16, 32, 64, 128\}$. The feature max-pool sizes after each convolutional layer were $\{5, 4, 2\}$ for all the explored layouts. Also in this case the experiments were repeated for both the input features combination.

A successive grid search was performed for each network configuration evaluated, in order to find the post-processing parameters that yielded the minimum error rate. Investigated parameters in the grid search were: exponential window length w in the range $\{10, 20, \ldots, 90\}$, median filter kernel k in the range $\{9, 11, \ldots, 31\}$ and threshold θ in the range $\{0, 0.05, \ldots, 0.5\}$.

Once the best models on the Development dataset were found, a fine tuning of the post processing parameters was done during the validation stage, in order to assess the performance of the whole system. In fact, the hierarchical architecture of the algorithm allows to set a lower threshold in the first decision stage in order to reduce the deletions at the expenses of some insertions. These will be removed by the ED stage 2.

Training set for CNN based ED Stage 2 To compose the dataset for training and evaluation of the CNNs dedicated to each target audio event we proceeded as follows: the samples of each event class were selected between the audio sections respectively labelled as "baby cry", "glass break" and "gun shot" from the mixtures of the DCASE 2017 development dataset, in addition with the isolated events source signals. To obtain the background samples, we processed with the first stage of our algorithm sequences from the same dataset which do not contain events. Thus, the frames detected as event in this case represent the "false positive" or "insertions" of the stage 1. We used those frames as background samples in the CNN training phase to improve its event classification

abilities and balancing the dataset.

Refinement Stage

To design the best refinement CNN model for our purposes, we generated a shuffle stratified validation split from the dataset composed as described above. We left out the 30% of the samples as validation set for the CNN model and we selected the layout parameters of the neural network based on the F-measure score obtained on this data sub-set. The best performing model was the same for all the target audio events and was composed as follows: three convolutional layers with $\{32, 32, 32\}$ filters, respectively, of size 5×5. The convolutional layers were followed by a feature max pooling layer with kernels of size $\{5, 4, 2\}$, respectively. Three dense layers composed of 32 neurons with *tanh* activation functions were applied before the network output layer, for a total number of network parameters equal to 35K.

5.2.4 Results

Results reported in Table 5.4 are obtained as follows: we selected the models with lowest ER for each combination of DNN architecture and input features operating in the ED stage 1 and we evaluated the systems separately for each target class before the ED stage 2 on the Evaluation set, keeping ED stage 1 post processing parameters fixed. Then, with the same settings we obtained the performance of the whole system both on Development and Evaluation datasets. The architecture composed of a first stage with 2D CNN fed by Logmel and WC-LPE features resulted the best performing on the Evaluation dataset, obtaining an average ER equal to 0.17. Details of these architectures are reported in Table 5.5.

The experimental results show how this combination improves generalization properties of the algorithm. In fact, the MLP based stage 1 with only Logmel features obtains the best overall ER equal to 0.13 on the Development dataset, but the performance decreases significantly on the Evaluation set. In addition, the number of free parameters of the best performing MLP models was always

Hyper-parameters	Babycry	Glassbreak	Gunshot
Conv. Kernels	5×5, 3×3, 3×3	3×3, 3×3, 3×3	3×3, 3×3, 3×3
Kernel shape	32, 16, 16	64, 64, 64	32, 16, 16
MLP Layers size	32, 32	128, 128	32, 32
# Parameters	18k	185k	17k

Table 5.5: Details of models for CNN based ED stage 1 with the lowest ER on Development set. All of them use a combination of log mel energies and WC-LPE as input features.

Approach	Evaluation ER	# Parameters
Lim et al. [47]	**0.13**	6200K
Cakir et al. [48]	0.17	756K
Proposed system	0.22	**108K**
Phan et al. [49]	0.27	2100K

Table 5.6: Comparison between the obtained ER scores and the number of parameters with the first three ranked approaches at the DCASE2017 Challenge.

an order of magnitude greater w.r.t. the CNN models. Regarding the stage 2, its beneficial effect is supported especially with the Evaluation dataset: in this case, the improvement in terms of ER given by the joint detection procedure is evident and it gives additional robustness to the system in terms of generalization.

In Table 5.6 the overall results between best ranked systems of the DCASE 2017 Challenge are compared. It can be observed that the best two scores have been obtained with ensemble methods, involving the additional computational cost of running several architectures in parallel, while the table reports the number of parameters per architecture. Although the proposed system does not outperform the first two methods, the average number of network parameters is significantly lower. This provides greater scalability in real-world applications.

5.2.5 Conclusion

In this section, a framework that makes use of hierarchical CNN classifiers fed with Logmel and WC-LPE features has been proposed for rare SED, providing significantly improved performance over the baseline system for every target sound event class in DCASE 2017 challenge dataset. The system also provides a significant reduction of the network parameters w.r.t. other competitive algorithms. The multi-scaled approach inherent to the two different CNN architectures results to be effective.

For future work, strategies to customize the loss function embedding the evaluation metric into the training procedure can be considered. Specifically, this task is particularly affected by the dataset unbalancing: to counteract this problem an alternative to the data augmentation is to design tailored loss functions which enhance the detection of the rare events.

5.3 Convolutional Neural Networks with 3-D Kernels for Voice Activity Detection in a Multiroom Environment

The Voice Activity Detection (VAD) element is considered fundamental in systems for automatic-assisted home environments, since the speech signal exhaustively characterizes the human activity. In a multi-room domestic environment, Automatic Speech Recognition (ASR) engines can use the information of both the speech segments time boundaries and the room in which the speaker is located in order to improve the word recognition performance. In this context, the recent success encountered by deep learning motivated the investigation of completely data-driven approaches [21, 131], specially when is useful to exploit the information contained in multiple audio signals. In this section, we focus on the use of three-dimensional kernels for Convolutional Neural Networks (CNN), taking advantage of an arrangement of the input data to the network rarely used in the audio field. Thus, due to speech signal degradation caused by background noise and reverberation, a multiple sensor (i.e., microphone arrays) deployment is necessary, leading to a rapid increase of data to process.

A multi-room domestic scenario requires the room localization and the time detection of speech events. For this purpose we propose the investigation of a 3-D Convolutional neural network (CNN-mVAD) for multichannel audio processing. A similar architecture employed in image classification was presented in [148] with remarkable performance. Our interest goes to the exploitation of the peculiarities of this technology compared to a typical neural network architecture, the Multi Layer Perceptron (MLP-mVAD). This paper contribution is on the choice of a CNN with 3-D kernels. They lead to the possibility of jointly processing simultaneous information from different audio channels, similarly to what occurs in image processing with RGB channels. In addition, CNNs are able to exploit the temporal evolution of the audio signal, and this is an useful feature for the VAD purpose [149].

The state-of-the-art VADs require many processing-stages to obtain the final decision, including the computation of typical characteristics of the acoustic wave or signal statistical descriptors [150]. In recent times, promising VAD approaches take advantage of deep neural networks. A speech/non-speech model based on a Multi-Layer Perceptron (MLP) neural network is proposed in [151], while in [152] multiple features are feed to a Deep Belief Neural Network (DBN) to segment the signal in multichannel utterances. CNNs have been recently employed in VAD tasks [134, 153] with encouraging results. In [154], the authors use a CNN to relabel training examples for a feedforward neural network, obtaining relative reductions in equal error rate of up to 11.5%.

5.3.1 Proposed Approach

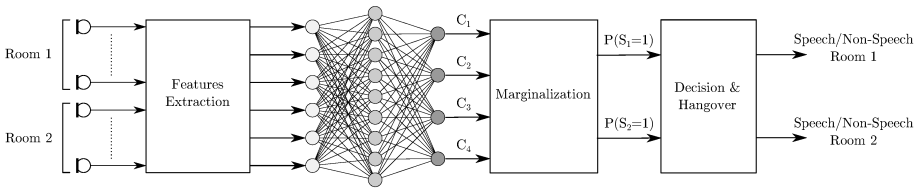


Figure 5.6: Block diagram of the proposed Neural Network Multi-Room VAD in a 2 rooms application. Several microphones can be exploited for each room, consisting in the multi-channel approach. Analysed classifiers are MLP and CNN.

The algorithm proposed in this section is suitable for speech detection in a n room context. In Figure 5.6 the block diagram of the NN-mVAD is depicted. Initially, features are extracted from the input audio signals. Successively, the Neural Network adopts a multi-class strategy in order to perform the classification task. In particular, the NN has an output layer of $K = 2^n$ units, where e.g., $n = 2$ due to the chosen rooms in our case of study. This leads to 4 output classes, one for each condition of speech/non-speech in the 2 considered rooms. Due to softmax behaviour, the 4 classes mean the joint probability of the 4 different events. Marginalization is then applied, obtaining separated probabilities for each room. Finally, the outputs are processed by a threshold block and a hangover scheme, with the focus on handling isolated speech detections.

Feature Extraction

For our purpose, we exploit *LogMel* as feature set. The feature extraction stage operates on signals sampled at 16 kHz. The used frame size is equal to 25 ms and the frame step is equal to 10 ms. 40 LogMel coefficients are extracted as described in Section 3.1.3. The range of feature values is then standardized to have zero mean and unitary standard deviation.

Due to multi-channel approach, features are structured in a specific order. For the MLP case, features of the different microphones and rooms are concatenated for each frame, resulting in a vector. For the CNN case a temporal context is exploited [153]. In particular, considered the current feature vector $\mathbf{x}[t]$ at the frame index t and a context size equal to C, the feature vector $\mathbf{x}[t]$ is concatenated with the previous and successive feature vectors $\{\mathbf{x}[t-c], \ldots, \mathbf{x}[t-1], \mathbf{x}[t+1], \ldots, \mathbf{x}[t+c]\}$, with $c = 1, \ldots, \frac{(C-1)}{2}$. This procedure leads to a 2-D feature matrix for each microphone. Finally, the 2-D matrices are stacked together, resulting in a 3-D matrix, where the dimensions correspond to the length of the feature set, the context and the number of selected microphones.

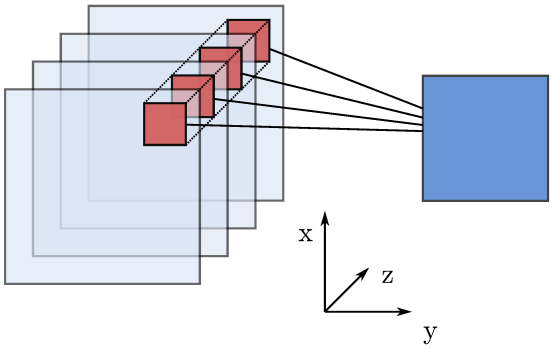


Figure 5.7: Convolution process for a 3-D kernel (red) over a 3-D input matrix (light blue). The result is a 2-D matrix (blue).

Convolutional Neural Network.

In this section, Convolutional Neural Network is compared to a Multi Layer Perceptron. A particular attention goes to the 3-D convolutional kernel. It processes a 3-D input matrix, as depicted in Figure 5.7. Convolution is performed along x and y axis. In z axis, the input matrix and the kernel have both the same number of layers. As result, for each z layer, a 2-D convolution is evaluated, leading to a number of *feature maps* equal to the number of layers. Finally, a 2-D output matrix is obtained by summing the feature maps in the z axis. As conclusion, for our case, the 3-D convolution process is suitable only for the first convolution layer of the CNN, successively, a 2-D convolution is performed in the following layers. Since we focus on audio application, it is interesting to analyse a signal excerpt which is extended in time. Thus, we make use of *strides* combined with frame context. In particular, this operation does not merge adjacent frames in order to obtain the input matrix, but it selects frames with a jump equal to the stride value.

Marginalization

The joint probabilities of the two rooms are marginalized by summing the conditional probabilities related to a specific room:

$$P(S_1 = 1) = P(S_1 = 1, S_2 = 0) + P(S_1 = 1, S_2 = 1), \tag{5.3}$$

$$P(S_2 = 1) = P(S_1 = 0, S_2 = 1) + P(S_1 = 1, S_2 = 1). \tag{5.4}$$

denoting with $S_i = 1$ the presence of speech in the room i and with $S_i = 0$ its absence. A threshold is then applied to the marginalization probabilities, leading to a binary signal. A smoothing step handles errors produced by the classifiers.

Decision and Hangover

We exploit a *hangover* technique, which relies on a counter. In particular, for two consecutive speech frames the counter is set to a predefined value. On the contrary, for each non-speech frame, the counter decreases by 1. If the counter is negative, the actual frame is classified as non-speech. The value of the counter is set to $\eta = 8$.

5.3.2 DIRHA Dataset

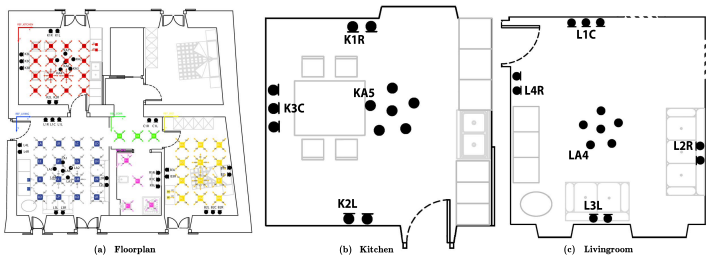


Figure 5.8: Layout of the apartment used as experimental set-up.

The dataset we used for our experiments was provided by the DIRHA project [155], it contains signals recorded in an apartment equipped with 40 microphones installed on the walls and the ceiling of each room[1]. The whole dataset is composed of two subsets called *Real* and *Simulated*, but we used only the latter since it contains more data, it is characterised by higher noise source rate and a wide variety of background noises. The Simulated dataset counts 80 scenes 60 seconds long consisting in localized acoustic and speech events in Italian language (23.6% of the total time), on which different real background noise with random dynamics are superimposed. It is artificially built: the signals are convolved with some available measured room impulse responses, simulating the acoustic wave propagation from the sound source to each single microphone.

5.3.3 Experiments

The analysis of proposed mVADs relies on a two-stage strategy: a network size selection and a microphone combination selection. The experiments are conducted by means of the k-fold cross-validation technique to reduce the performance variance. In this case we choose $k = 10$, a validation set is also employed during the training, thus, 64-8-8 scenes respectively compose the training, validation and test sets. The performance has been evaluated using

[1] `http://dirha.fbk.eu/simcorpora`

the false alarm rate (FA), the deletion rate (Del) and the overall speech activity detection (SAD) defined as follows:

$$\text{Del} = \frac{N_{del}}{N_{sp}}, \quad \text{FA} = \frac{N_{fa}}{N_{nsp}}, \quad \text{SAD} = \frac{N_{fa} + \beta N_{del}}{N_{nsp} + \beta N_{sp}}, \tag{5.5}$$

where N_{del}, N_{fa}, N_{sp} and N_{nsp} are the total number of deletions, false alarms, speech and non-speech frames, respectively. The term $\beta = N_{nsp}/N_{sp}$ acts as regulator term for the class unbalancing. Two different GPU-based toolkits have been employed for the experiments: a custom version of GPUMLib [156] for MLP-mVAD and *Keras* (*Theano-based*)[2] for CNN-mVAD. The MLP networks were trained with a fixed momentum of 0.9, learning rate equal to 0.01 and a Gaussian distribution with zero mean and standard deviation of 0.1 for weight initialization. For the CNN networks we used a fixed learning rate of $2.5 \cdot 10^{-3}$ and a random weight initialization.

CNN						
		1R 1MxR		2R 1MxR	2R 2MxR	2R 3MxR
		Kitchen	Living Room			
Input Params	Strides	8	10	8	8	8
	Context	17	23	25	23	23
First Convolutional Layer	N Kern	16	16	32	128	256
	Size	6×6	6×6	4×4	4×4	4×4
	Pooling	2×2	2×2	2×2	-	-
Second Convolutional Layer	N Kern	24	16	64	64	32
	Size	4×4	4×4	3×3	3×3	3×3
	Pooling	-	-	-	-	-
Third Convolutional Layer	N Kern	24	16	128	32	32
	Size	3×3	3×3	3×3	3×3	3×3
	Pooling	-	-	-	-	-
Fully Connected Layers	Num. of Units	100	100	500	250	500
		20	20	100	100	100
SAD Min (%)		9.0	10.7	9.3	8.1	7.0
MLP						
Fully Connected Layers	Num. of Units	10	15	10	8	8
		-	-	-	-	-
SAD Min (%)		11.8	13.3	11.7	8.8	7.4

Table 5.7: Network topology parameter for CNN- and MLP-mVAD.

5.3.4 Results

In this section, the obtained results in terms of SAD are discussed and compared for the two different architectures of neural network. The analysis of proposed mVADs relies on a multi-stage strategy, where the best network size and microphone channel combination are searched. The steps are the following:

[2] `http://keras.io/`

1. one network per room, one microphone per room;
2. one network per two room: one, two and three microphone per room.

Regarding network size selection, MLP-mVAD network topologies are explored by means of 1 or 2 hidden layers with respectively $4, 8, 10, 15, 20, 25, 40$ units per layer and all their combinations. For CNN-mVAD, due to their greater number of hyperparameters and increased training time, a comprehensive grid search was not reasonable, thus we adopted a progressive strategy, based on intermediate results.

Concerning audio channels selection, we initially selected a subset of 9 microphones: 4 in the kitchen (i.e., K2L, K1R, K3C, KA5) and 5 in the living room (i.e., L1C, L2R, L3L, L4R, LA4). In the experiments with one microphone per room, we evaluated the performance for all of them, successively, in the following stages we analyse only combinations obtained with the best performing ones.

One network per room, one microphone per room (1R 1MxR).

In this step we evaluated the performance considering two different VADs, one for the kitchen and one for the living room. In the network size selection, the best MLP-VAD resulted to have one layer with 10 units and 8 units respectively for the kitchen and the living room. In the second stage, the best performing microphone for the kitchen was the KA5, while for the living room the LA4: both of them are placed at the center of the room ceiling and the averaged SAD was equal to 12.5%. The two networks exploited for the CNN-VAD are reported in Table 5.7. As for MLP-VAD, best microphones are KA5 and LA4, with an average 9.9% SAD.

One network per two rooms, one microphone per room (2R 1MxR).

From this step we started to evaluate the performance of properly mVAD, using both in training and in test audio channels coming from the two rooms. First of all we used only one channel per room: the best MLP-mVAD has one layer with 15 units and the audio captured by the pair KA5, LA4 (confirming the result of the previous step), leading to a SAD equal to 11.7%. For the CNN-mVAD, SAD equal to 9.3% is again obtained with the pair of microphones KA5 and LA4. CNN topology is reported in Table 5.7.

One network per two rooms, two microphones per room (2R 2MxR).

We progressively introduced one more audio channel per room, primarily by repeating the network topology selection. Compared to the previous step, the best configuration for MLP-mVAD has only one hidden layer with 8 neurons.

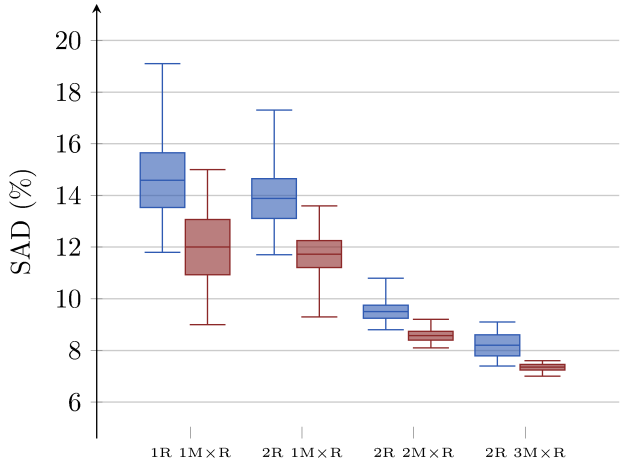


Figure 5.9: Box-plot of the resulting SADs for the microphone selection experiments in the different steps. Evident is the improvement given by increasing the microphone number, and, for the CNN-mVAD, the related statistical robustness.

In the microphone selection, on the basis of the above analysis we evaluate the 12 combinations of double pairs of channels, achieving with the couple KA5, K1R (from the kitchen) and LA4, L2R (from the living room) an absolute improvement of -2.9% of SAD in respect to the case with one microphone per room. Settings of the CNN-mVAD are shown in Table 5.7. Again, best microphones are the same of the MLP-VAD: KA5, K1R, LA4, L2R. The resulting SAD is 8.1%.

One network per two rooms, three microphones per room (2R 3MxR).

In the last step experiments we evaluate the performance of mVADs that process three audio channels per room. For the MLP-mVAD the network topology remains the same as in the case with two microphones per room and the best result (SAD = 7.4%) is obtained with the combination K1R, K2L, KA5, L1C, L2R, LA4. The CNN-mVAD achieves 7.0% SAD with topology shown in Table 5.7, selected microphones are: K1R, K3C, KA5, L2R, L4R, LA4.

5.3.5 Conclusion

A neural network approach for voice activity detection in a domestic environment is presented in this section, paying specific attention to a smart use of the input features. In particular, a multi-channel strategy is implemented, consisting in the usage of multiple microphones as input of the algorithm. Two networks are investigated, which are MLP and CNN. The latter was recently

exploited for audio task, with remarkable results. Moreover, due to the suitable CNN structure for multi-channel investigation, we make use of 3-D convolutional kernel, whose dimensions correspond to frequency, time and used microphones. In detail, LogMel features are chosen to represent the frequency domain, in order to convolve the CNN kernel with correlated inputs. Time is explored by means of a temporal context plus strides, allowing the CNN to process an excerpt of the signal with duration about 2 s. Multi-channel features are stacked together, leading to a 3-D input matrix.

The optimization strategy consists in two steps, a network size selection and a microphone selection. Four different studies are conducted, which are a two network approach for single room VAD with only one microphone, and a unique network for the two rooms VAD, featuring one, two and three microphones. The latter achieves the best performance in terms of SAD, leading to 7.4% for MLP and 7.0% for CNN. A remarkable aspect of the CNN mVAD is the robustness to the microphone choice, with lower mean and standard deviation. The independence from the audio source positioning is an interesting applicative result. On the contrary, due to the dimension of the CNN, simulation time is considerably longer compared to MLP.

Future works will be oriented to the employment of raw audio data as input for the CNN, in order to exploit the network feature extraction capability.

Chapter 6

Polyphonic Sound Event Detection

Sound event detection (SED) algorithms in a real-life scenario face many challenges. These include environmental noise, events of the same class produced by different sources (i.e., intra-class variability) and simultaneous events. In particular, since multiple events are very likely to overlap, a *polyphonic* SED algorithm, i.e., an algorithm able to detect multiple simultaneous events, needs to be designed. A polyphonic SED algorithm can be considered as system which is able to perform contemporary detection - determining the starting and ending point of multiple events - and classification - assigning a label to each of the sound events occurring in the audio stream. In this chapter we show two algorithms for polyphonic SED in real life audio. The solutions typically adopted in the literature for this applications require a two-stage system, i.e., one algorithm for detection and one for classification. According to this, the first we present is based on a two-step algorithm: the detection task is performed by an adaptive energy Voice Activity Detector (VAD) system, while the active-event classification is acted by a deep neural network. The second system is a fully data-driven approach which is totally based on the CapsNet deep neural architecture presented by Sabour et al. [52].

6.0.1 State-of-the-art Overview

In occasion of the DCASE 2016, many novel systems featuring recurrent neural networks (RNNs) and multilayer perceptrons (MLPs) have been proposed, even though only one of them [157] managed to outperform the baseline system (based on a GMM) thus reaching the first rank in the third task of the challenge. In our opinion, this proves that there is still a lot of space for research in approaching SED with ANNs. During our experiments we compare different well-established audio representations, *i.e.,* STFT Spectrograms, log-mel energies and mel-frequency cepstral coefficients (MFCCs), extracted in both monaural and binaural configuration.

We here propose two algorithms, one which relies on a voice activity detection (VAD) algorithm for the detection of acoustic events which are then

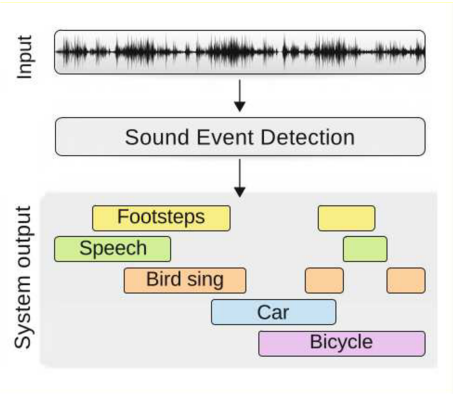


Figure 6.1: System input and output for polyphonic sound event detection. Pic. courtesy of [1].

classified by an ANN, and one "unified" system which performs both detection and classification. Therefore, we evaluate als different ANN architectures, *i.e.*, MLPs, RNNs, CNNs and the novel CapsNets. Our aim is therefore to give a novel contribution by presenting a robust system capable to improve the results obtained until now.

6.1 Sound Event Detection in Real Life Audio - DCASE 2016

In this section we present and compare two neural-based algorithms designed for sound event detection in real life audio. Both systems have been developed and evaluated with the material provided for the third task of the Detection and Classification of Acoustic Scenes and Events (DCASE) 2016 challenge. For the first algorithm, we make use of an artificial neural network trained on different features extracted from the down-mixed Monaural-channel audio. Secondly, we analyse a binaural algorithm where the same feature extraction is performed on four different channels: the two binaural channels, the averaged monaural signal and the difference between the binaural channels. The proposed feature set comprehends, along with mel-frequency cepstral coefficients and log-mel energies, also activity information extracted with two different voice activity detection (VAD) algorithms.

6.1.1 Proposed method

As we can see from Figure 6.2 it is possible to divide the system functioning into two phases: training and testing. During training we do not need to use

any algorithm for event detection, since onset and offset instants are already provided in the ground truth, therefore we can simply train an ANN to recognise the different events. At test time, on the other hand, onset and offset instants are not given, therefore we firstly make use of a VAD algorithm for determining them, and secondly we feed the corresponding audio sequences to the ANN classifier.

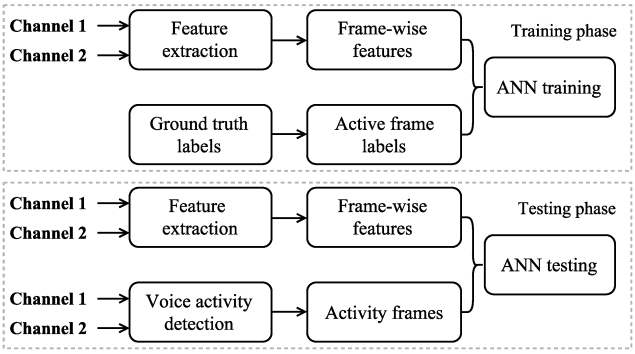


Figure 6.2: Block diagram of training and testing phases. At test time a VAD algorithm is used to determine onset and offset instants of the detected events.

Feature representations

In order to perform the SED task with ANNs, a set of one or more audio representations have been extracted from the raw audio signal. Aiming to evaluate the impact of binaural information on the classification performance, we will, in the first instance, distinguish the proposed sets between monaural and binaural feature sets. We highlight that, for all the extracted feature sets, a frame-wise short-time Fourier transform (STFT) is firstly applied to the audio signal on frame windows of 40 ms with 50% overlap. Moreover, all feature extraction processes described hereafter have been performed with openSMILE [127], a license-free software package developed by the Technical University of Munich.

The first monaural set is known as log-mel spectrogram. In this case a downmixing of the two audio channels is required before calculating the STFT coefficients. After the STFT coefficients are extracted, we apply a mel conversion of the frequency scale with a 26-bands mel-scale filter bank and compute the logarithm of all the energies so obtained. To complete the set, we also extract the first order delta coefficients operating on a context window of 10 frames. Given the log-mel coefficients and their respective deltas, this first set is composed of 52 coefficients for each frame.

The second monaural set is composed of another set of widely used features, that is mel-frequency cepstral coefficients (MFCCs). Starting from the same STFT coefficients previously obtained, we now compute the log-mel spectrogram with a 40-bands mel-scale filter bank. Then, we apply a 20-points discrete cosine transform (DCT) to each energy vector and, after excluding the 0^{th} order coefficient, we obtain a feature vector of 20 MFCCs. In order to complete the set, we also calculate the first and second order delta coefficients, therefore obtaining a 60-coefficients feature vector.

For the two binaural sets we decided to extract the log-mel (or the MFCC) features not only from the average of the two channels, but also from their difference and the two separate channels; this gives us a total of four channels. We decided to do so because, for example, if an important event is predominant in only one of the two channels, averaging them could lower the signal-to-noise ratio, thus increasing the system failure probability. For the first binaural set we extract the log-mel coefficients as for the first monaural set, but, given the presence of four channels, we now have a total of 108 coefficients for each frame. In order to avoid an excessive feature vector dimension, in this set we decide to avoid using the first and second order delta coefficients. Similarly, the second binaural set is obtained by extracting 20 MFCCs for each channel; since again we avoid using the delta coefficients, this leads us to a total of 80 coefficients for each frame.

Neural networks

In this section, two different ANNs architectures are tested for the SED problem, *i.e.,* MLPs and RNNs. The first layer of all proposed ANNs consists of a set of nodes to which the audio representation (taken on a frame scale) is applied, with the number of nodes varying from 52 to 108, depending on the chosen feature representation. The input is then propagated to the following three hidden layers, composed of 512 *tanh* neurons each for MLPs and 54 rectifier neurons for RNNs. Finally, the last layer of our networks is designed to output the class associated by the network with the given input. To do so, this layer is composed of a number of softmax neurons equal to the number of possible classes, *i.e.,* 11 if we are dealing with a "home" scenario, and 7 in case of a "residential area" (see Table 6.1). We highlight that, in case of MLPs, we obtain one label for each frame, whereas RNNs are able to output one label also for a batch of sequent frames.

The standard algorithm used for training the proposed MLPs is the back-propagation (BP) algorithm, whereas for RNNs the "BP through time" is used. After a first feed-forward phase, in which a batch of input is propagated through the networks, these algorithms exploit the derivative "chain rule" to back-propagate the error computed at the output layer and sequentially update all

neuron weights [158].

Sound event detection and classification

At test time we want the system to detect and correctly classify as many acoustic events as possible occurring in raw audio files of 30 seconds. Since no onset nor offset instants are given, we decide to use VAD algorithms in order to detect these instants. With this intent, two different VAD algorithms have been tested, *i.e.,* AE, and Sohn's VAD.

The AE approach makes use of two energy thresholds in order to determine the starting and ending point of an event-active audio sequence, *i.e.,* the "mean plus variance" (MPV) and the "mean minus variance" (MMV) thresholds. These thresholds are firstly calculated over all the training dataset and then used to extract information about events activity: whenever a frame's energy exceeds the MPV threshold an onset event is triggered, then the event detection remains positive until the energy content drops below the MMV threshold.

Sohn's VAD [159], on the other hand, is a method based on a statistical modelling of the audio in the time-frequency domain, with the model parameters being estimated with a maximum likelihood (ML) method. With this technique, the decision regarding the event's activity is devolved to a comparison between the averaged log-likelihood ratio (containing the a-priori and a-posteriori signal-to-noise ratios) and a fixed threshold $\eta \in (0, 1)$.

Whenever an audio file is processed by one of the two VAD algorithms we are able to extract the starting and ending instants between which an audio event has (supposedly) occurred. Hence, we can feed the network with the feature representation of the corresponding frames and finally obtain the event classification. We remind that, in case of MLPs, we obtain one label for each frame, whereas RNNs are able to output only one label for the whole frame batch. Due to this, we need to average all MLP outputs so to obtain the event's acoustic label, while RNN's outputs will need no further processing.

6.1.2 Experiments

Datasets and metrics

The data we use during our experiments consist of two datasets provided for the third task (SED in real life audio) of the DCASE 2016 challenge [56]. The former dataset, called *development dataset*, was at first provided in order to make all challengers able to compare their development results, while the latter, the *evaluation dataset*, was used for the final evaluation of the submitted systems, so its ground truth was made public only a few weeks after the end of

the challenge. These acoustic scenes were selected from the challenge organizers to represent common environments of interest in applications for safety and surveillance (outside home) and human activity monitoring or home surveillance [56].

Both datasets contains recordings of 3-5 minutes divided into two different acoustic scenarios: "home" and "residential area". Specific classes for each scenario are reported in Table 6.1.

Home	*Occurrences*	*Residential area*	*Occurrences*
rustling	60	banging	23
snapping	57	bird singing	271
cupboard	40	car passing by	108
cutlery	76	children shouting	31
dishes	151	people walking	52
drawer	51	people speaking	44
glass jingling	36	wind blowing	30
object impact	250		
people walking	54		
washing dishes	84		
water tap running	47		

Table 6.1: Classes and their occurrences for the "home" and "residential area" scenarios for the SED in real life audio task of the DCASE 2016 challenge.

The development dataset consists of 10 recordings for the "home" scenario, and 12 for the "residential area", and for both a four-folds cross-validation data splitting is provided by the organizers of the challenge. While creating the cross-validation folds, the challenge organizers imposed the only condition that the test subset does not contain classes unavailable in training subset, therefore the class distribution between the test subsets is not assumed to be uniform. The evaluation dataset contains 5 recordings for both the "home" and the "residential area" scenarios each. For this dataset no cross-validation is performed, so it is possible to train only one system with all the development dataset (including files previously meant for testing purpose) and then test it with the evaluation files.

Scores used to evaluate all systems are the well known F1 and error rate (ER) scores, which are used to evaluate the system over segments of one second. In obtaining the final score for the development dataset, we average the four per-fold scores as described in [56].

6.1.3 Results

Experimental setup

Concerning the network training, we initialize all weights according to a normal distribution with zero mean and 0.1 variance. We then train the networks following the adam [109] method for stochastic optimization, for which we keep the default hyper-parameter configuration. In order to prevent overfitting, for each fold we check the network performance on the respective fold's test set after each training epoch. If no improvement on this set is encountered for 60 consecutive epochs, the training is forced to an early stop. After this phase we perform experiments on the evaluation data, for which we use the whole development training and test sets as training and validation data respectively. Then, at test time, we evaluate the system on the secret challenge data.

Development results

During our experiments we tested and compared different neural architectures, VAD algorithms, and feature representations. In Table 6.2 we report the results obtained with Sohn's VAD for 16 different system configurations, whereas in Table 6.3 the same classifier and feature configurations are analysed in conjunction with AE VAD.

Table 6.2 highlights that the use of binaural audio features always enhances the system's performance in terms of both F1 and ER scores. Moreover, we can also notice that MLPs generally perform better than RNNs, in particular according to F1 scores, where no RNN manages to achieve more than 34.4% F1 score. Finally, we report that the best system's configuration featuring Sohn's VAD is a MLP trained with binaural MFCC features, with a VAD threshold equal to 0.70. This system manages to reach 0.88 ER and 39.8% F1 score, both averaged on the four folds.

Table 6.3 mostly confirms what emerged from the analysis of the previous table. Also with adaptive evergy VAD, the use of binaural features always improves the classification accuracy, even if differences are now less marked, with the highest improvement in F1 scores being +2%. Moreover, it is interesting to notice that the difference between MLPs and RNNs accuracies is now reduced, maybe highlighting that the difference between their classification power thins if a better VAD algorithm leads to a better event detection. The best performing system featuring AE VAD is again a MLP which, with binaural log-mel features, manages to reach 0.78 ER and 43.1% F1 scores, averaged on the four folds as for the previous results.

Features	η	*Classifier*	*ER*	*F1* (%)
Monaural log-mel	0.98	MLP	0.93	34.6
Binaural log-mel	0.98	MLP	0.89	38.6
Monaural log-mel	0.70	MLP	0.90	35.4
Binaural log-mel	0.70	MLP	0.89	39.4
Monaural MFCC	0.98	MLP	0.92	35.7
Binaural MFCC	0.98	MLP	0.88	39.6
Monaural MFCC	0.70	MLP	0.91	36.2
Binaural MFCC	**0.70**	**MLP**	**0.88**	**39.8**
Monaural log-mel	0.98	RNN	0.91	29.6
Binaural log-mel	0.98	RNN	0.88	35.6
Monaural log-mel	0.70	RNN	0.95	28.2
Binaural log-mel	0.70	RNN	0.88	34.4
Monaural MFCC	0.98	RNN	0.98	30.5
Binaural MFCC	0.98	RNN	0.88	34.1
Monaural MFCC	0.70	RNN	0.91	31.2
Binaural MFCC	0.70	RNN	0.88	31.0

Table 6.2: Comparison of Scores obtained on the development dataset using different Features, Classifiers and Sohn's VAD Thresholds (η). Scores are Averaged among the four Cross-Validation Folds.

Evaluation results

In Table 6.4 we report the main results for the most promising system configurations tested on the evaluation dataset. As we can see, scores tend to be higher than the ones obtained on the development dataset, especially for MLPs, highlighting the benefit introduced by the addition in the training set of those files previously used for testing. The expansion of the training set can be viewed as the expansion of the "knowledge" from which the network can learn at training time, therefore, when this happens, it is expectable to reach a better generalization performance. This behaviour is confirmed, the best performing configuration manages to achieve a 0.79 ER and 48.1% F1 scores, and it consists of a MLP classifier trained on binaural MFCC features.

Table 6.5 compares our best system to the three best performing ones proposed for the third task of the DCASE 2016 challenge. The first and the third ranks were achieved by Adavanne *et al.*, which made use of RNN-LSTM architectures trained on spatial and harmonic features [157] extracted from the two binaural channels. On the other hand, the second best system is the baseline proposed in [56], based on a GMM modelling of each acoustic event, plus one for the absence of sound events, which was trained with the non-labelled

Features	*Classifier*	*ER*	*F1* (%)
Monaural log-mel	MLP	0.78	41.2
Binaural log-mel	**MLP**	**0.78**	**43.1**
Monaural MFCC	MLP	0.81	40.1
Binaural MFCC	MLP	0.82	42.1
Monaural log-mel	RNN	0.85	41.2
Binaural log-mel	RNN	0.82	43.1
Monaural MFCC	RNN	0.92	40.7
Binaural MFCC	RNN	0.89	41.0

Table 6.3: Comparison of Scores obtained on the development dataset using different Features, Classifiers and AE VAD. Scores are Averaged among the four Cross-Validation Folds.

Features	*VAD*	*Classifier*	*ER*	*F1* (%)
Monaural log-mel	Sohn ($\eta = 0.70$)	MLP	0.80	40.2
Binaural log-mel	Sohn ($\eta = 0.70$)	MLP	0.78	46.5
Monaural MFCC	AE	MLP	0.79	45.1
Binaural MFCC	**AE**	**MLP**	**0.78**	**48.1**
Monaural MFCC	AE	RNN	0.82	41.0

Table 6.4: Comparison of Scores obtained on the Evaluation Dataset using different Features, Classifiers and VAD Algorithms.

frame's features (MFCCs and their delta/delta-deltas were used). As we can see from the table, the proposed system manages to improve the F1 score by 0.3% while reducing the error rate by 0.02.

6.1.4 Conclusion

In this section we have proposed and evaluated a system for SED in real life audio. We compared different audio features, extracted in both monaural and binaural configurations, with which we trained different neural network classifiers. Moreover, we tested two different VAD algorithms for detecting sound activities to be classified by the proposed networks at test time. The proposed best performing system achieves an improvement on the winner of the third task in the DCASE 2016 challenge, thus highlighting the competitiveness of the proposed approach. In addition, the results show that in our experiments the MLPs generally perform better than RNNs, in particular according to F1 scores. In this case, it can be motivated by the limited size of the available data, which could be not sufficient to train appropriately a deep RNN.

Features	*VAD*	*Classifier*	*ER*	*F1* (%)
Binaural log-mel	**AE**	**MLP**	**0.79**	**48.1**
Binaural mel energy	-	RNN [157]	0.81	47.8
Binaural mel energy	-	GMM [56]	0.88	23.7
Binaural mel energy + TDOA	-	RNN [157]	0.89	34.3

Table 6.5: Comparison Between the proposed System and the three (out of 17) best performing DCASE 2016 Systems proposed for SED in real life audio.

6.2 Polyphonic Sound Event Detection by using Capsule Neural Networks

In this section, we present an extensive analysis of SED conducted on real-life audio datasets and compare the results with state-of-the-art methods. In addition, we propose a variant of the dynamic routing procedure which takes into account the temporal dependence of adjacent frames. The proposed method outperforms previous SED approaches in terms of detection error rate in the case of polyphonic SED, while it has comparable performance with respect to CNNs in the case of monophonic SED.

The proposed system is a fully data-driven approach based on the CapsNet deep neural architecture presented by Sabour et al. [52]. This architecture has shown promising results on highly overlapped digital numbers classification. In the audio field, a similar condition can be found in the detection of multiple concomitant sound events from acoustic spectral representations, thereby we propose to employ the CapsNet for the polyphonic-SED in real-life recordings. The novel computational structure based on capsules, combined to the routing mechanism allows to be invariant to intra-class affine transformations and to identify part-whole relationships between data features. In the SED case study, it is hypothesized that this characteristic confers to CapsNet the ability to effectively select most representative spectral features of each individual sound event and separate them from overlapped descriptions of the other sounds in the mixture.

6.2.1 Related Works

Encouraging polyphonic SED performance have been obtained using CapsNets in preliminary experiments conducted on the Bird Audio Detection task in occasion of the DCASE 2018 challenge [50], confirmed by the results reported in [51]. The CapsNet [52] is a recently proposed architecture for image classification and it is based on the grouping of activation units into novel structures introduced in [53], named *capsules*, along with a procedure called dynamic routing. The capsule has been designed to represent a set of properties for an entity of interest, while dynamic routing is included to allow the network to implicitly learn global coherence and to identify part-whole relationships between capsules.

The authors of [52] show that CapsNets outperform state-of-the-art approaches based on CNNs for digit recognition in the MNIST dataset case study. They designed the CapsNet to learn how to assign the suited partial information to the entities that the neural network has to predict in the final classification. This property should overcome the limitations of solutions such as max-pooling,

currently employed in CNNs to provide local translation invariance, but often reported to cause an excessive information loss. Theoretically, the introduction of the dynamic routing can supply invariances for any property captured by a capsule, allowing also to adequately train the model without requiring extensive data augmentation or dedicated domain adaptation procedures. This hypothesis is supported by previously mentioned related works. Specifically, in [50], the CapsNet is exploited in order to obtain the prediction of the presence of heterogeneous polyphonic sounds (i.e., bird calls) on unseen audio files recorded in various conditions. In [51] the dynamic routing yields promising results for SED with a *weakly labeled* training dataset, thus with unavailable ground truths for the onset and offset times of the sound events. The algorithm has to detect sound events without supervision and in this context the routing can be considered as an attention mechanism.

6.2.2 Proposed Method

The aim of polyphonic SED is to find and classify any sound event present in an audio signal. The algorithm we propose is composed of two main stages: sound representation and polyphonic detection. In the sound representation stage, the audio signal is transformed in a two-dimensional time-frequency representation to obtain, for each frame t of the audio signal, a feature vector $\mathbf{x}_t \in \mathbb{R}^F$, where F represents the number of frequency bands.

Sound events possess temporal characteristics that can be exploited for SED, thus certain events can be efficiently distinguished by their time evolution. Impulsive sounds are extremely compact in time (e.g., gunshot, object impact), while other sound events have indefinite length (i.e., wind blowing, people walking). Other events can be distinguished from their spectral evolution (e.g., bird singing, car passing by). Long-term time domain information is very beneficial for SED and motivates for the use of a temporal *context* allowing the algorithm to extract information from a chronological sequence of input features. Consequently, these are presented as a context window matrix $\mathbf{X}_{t:t+T-1} \in \mathbb{R}^{T\times F\times C}$, where $T \in \mathbb{N}$ is the number of frames that defines the sequence length of the temporal context, $F \in \mathbb{N}$ is the number of frequency bands and C is the number of audio channels. Differently, the target output matrix is defined as $\mathbf{Y}_{t:t+T-1} \in \mathbb{N}^{T\times K}$, where K is the number of sound event classes.

In the SED stage, the task is to estimate the probabilities $p(\mathbf{Y}_{t:t+T-1}|\mathbf{X}_{t:t+T-1}, \boldsymbol{\theta})$ $\in \mathbb{R}^{T\times K}$, where $\boldsymbol{\theta}$ denotes the parameters of the neural network. The network outputs, i.e., the event activity probabilities, are then compared with a threshold in order to obtain event activity predictions $\hat{\mathbf{Y}}_{t:t+T-1} \in \mathbb{N}^{T\times K}$. The parameters $\boldsymbol{\theta}$ are trained by supervised learning, using the frame-based annotation of the sound event class as target output, thus, if class k is active during frame

t, $Y(t,k)$ is equal to 1, and is set to 0 otherwise. The case of polyphonic SED implies that this target output matrix can have multiple non-zero elements K in the same frame t, since several classes can be simultaneously present.

Indeed, polyphonic SED can be formulated as a multi-label classification problem in which the sound event classes are detected by multi-label annotations over consecutive time frames. The onset and offset time for each sound event are obtained by combining the classification results over consequent time frames. The trained model will then be used to predict the activity of the sound event classes in an audio stream without any further post-processing operations and prior knowledge on the events locations.

Feature Extraction

For our purpose, we exploit two acoustic spectral representations, the magnitude of the Short Time Fourier Transform (STFT) and the *LogMel* coefficients, obtained from all the audio channels and extensively used for other SED algorithms. Except where differently stated, we study the performance of binaural audio features and compare it with those extracted from a single channel audio signal. In all cases, we operate with audio signals sampled at 16 kHz and we calculate the STFT with a frame size equal to 40 ms and a frame step equal to 20 ms. Furthermore, the audio signals are normalized to the range $[-1, 1]$ in order to have the same dynamic range for all the recordings.

The STFT is computed on 1024 points for each frame, while LogMel coefficients are obtained by filtering the STFT magnitude spectrum with a filterbank composed of 40 filters evenly spaced in the mel frequency scale. In both cases, the logarithm of the energy of each frequency band is computed. The input matrix $\mathbf{X}_{t:t+T-1}$ concatenates $T = 256$ consequent STFT or LogMel vectors for each channel $C = \{1, 2\}$, thus the resulting feature tensor is $\mathbf{X}_{t:t+T-1} \in \mathbb{R}^{256 \times F \times C}$, where F is equal to 513 for the STFT and equal to 40 for the LogMels. The range of feature values is then normalized according to the mean and the standard deviation computed on the training sets of the neural networks.

CapsNet for Polyphonic Sound Event Detection

The architecture of the neural network is shown in Figure 6.3. The first stages of the model are traditional CNN blocks which act as feature extractors on the input $\mathbf{X}$. After each block, max-pooling is used to halve the dimensions only on the frequency axis. The feature maps obtained by the CNN layers are then fed to the Primary Capsule Layer that represents the lowest level of multi-dimensional entities. Basically, it is a convolutional layer with $J \cdot M$ filters, i.e., it contains M convolutional capsules with J kernels each. Its output is then

reshaped and squashed using (2.18). The final layer, or Detection Capsule layer, is a time-distributed capsule layer (i.e., it applies the same weights and biases to each frame element) and it is composed of K densely connected capsule units of dimension G. Since the previous layer is also a capsule layer, the dynamic routing algorithm is used to compute the output. The *background* class was included in the set of K target events, in order to represent its instance with a dedicated capsule unit and train the system to recognize the absence of events. In the evaluation, however, we consider only the outputs relative to the target sound events. The model predictions are obtained by computing the Euclidean norm of the output of each Detection Capsule. These values represent the probabilities that one of the target events is active in a frame t of the input feature matrix $\mathbf{X}$, thus we consider them as the network output predictions.

In [52], the authors propose a series of densely connected neuron layers stacked at the bottom of the CapsNet, with the aim to regularize the weights training by reconstructing the input image $\mathbf{X} \in \mathbb{N}^{28\times 28}$. Here, this technique entails an excessive complexity of the model to train, due to the higher number of units needed to reconstruct $\mathbf{X} \in \mathbb{R}^{T\times F\times C}$, yielding poor performance in our preliminary experiments. We decided, thus, to use dropout [68] and L_2 weight normalization [160] as regularization techniques, as done in [51].

6.2.3 Experimental Setup

In order to test the proposed method, we performed a series of experiments on three datasets provided to the participants of different editions of the DCASE challenge [136, 56]. We evaluated the results by comparing the system based on the Capsule architecture with the traditional CNN. The hyperparameters of each network have been optimized with a *random search* strategy [139]. Furthermore, we reported the baselines and the best state-of-the-art performance provided by the challenge organizers.

Dataset

We assessed the proposed method on three datasets, two containing stereo recordings from real-life environments and one artificially generated monophonic mixtures of isolated sound events and real background audio.

In order to evaluate the proposed method in polyphonic real-life conditions, we used the TUT Sound Events 2016 & 2017 datasets, which were included in the corresponding editions of the DCASE Challenge. For the monophonic-SED case study, we used the TUT Rare Sound Events 2017 which represents the task 2 of the DCASE 2017 Challenge.

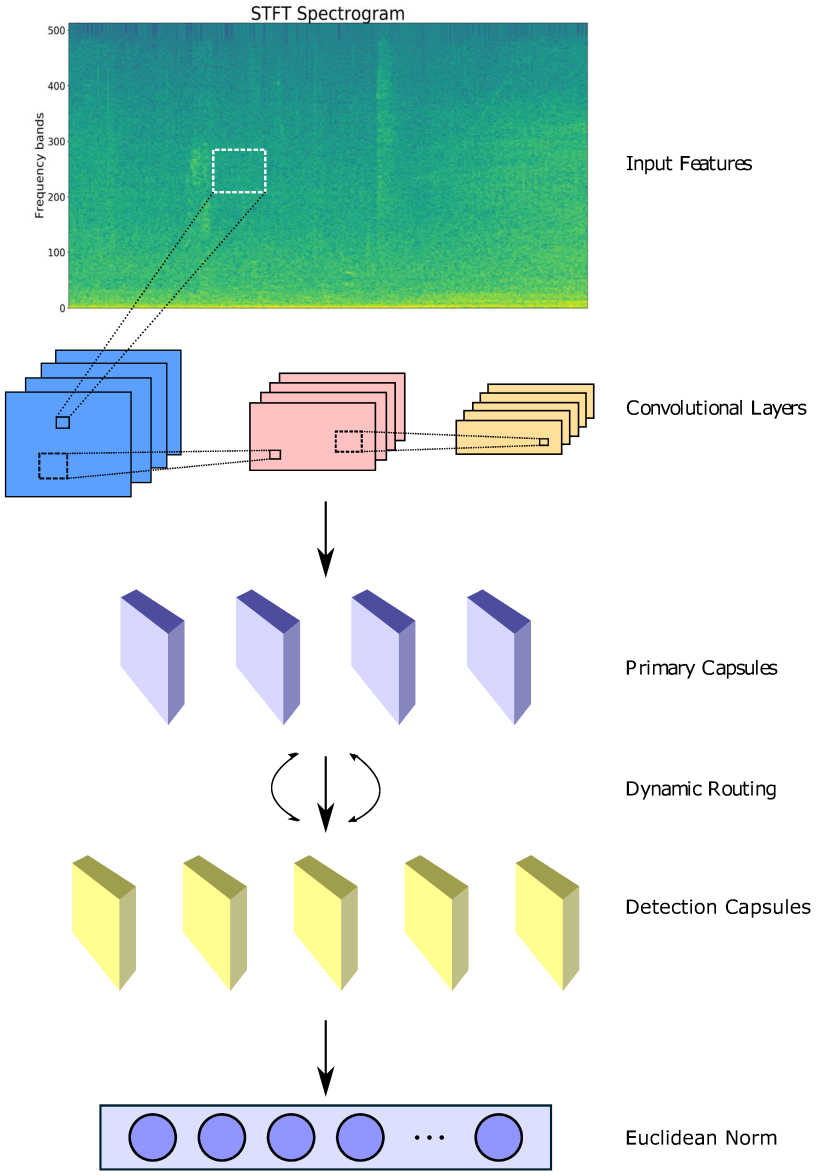


Figure 6.3: Flow chart of the Capsule neural network architecture used for polyphonic sound event detection.

TUT Sound Events 2016

The TUT Sound events 2016 (TUT-SED 2016)[1] dataset consists of recordings from two acoustic scenes, respectively "Home" (indoor) and "Residential area" (outdoor) which we considered as two separate subsets. A total amount of around 54 and 59 minutes of audio are provided respectively for "Home" and "Residential area" scenarios. Sound events present in each recording were manually annotated without any further cross-verification, due to the high level of subjectivity inherent to the problem. For further details, please refer to Section 6.1.2.

TUT Sound Events 2017

The TUT Sound Events 2017 (TUT-SED 2017)[2] dataset consists of recordings of street acoustic scenes with various levels of traffic and other activities, for a total of 121 minutes of audio. The scene was selected as representing an environment of interest for detection of sound events related to human activities and hazard situations. It is a subset of the TUT Acoustic scenes 2016 dataset [56], from which also TUT-SED 2016 dataset was taken. Thus, the recording setup, the annotation procedure, the dataset splitting, and the cross-validation setup is the same described above. The 6 target sound event classes were selected to represent common sounds related to human presence and traffic, and they include brakes squeaking, car, children, large vehicle, people speaking, people walking. The evaluation set of the TUT-SED 2017 consists of 29 minutes of audio, whereas the development set is composed of 92 minutes of audio which are employed in the cross-validation procedure.

TUT Rare Sound Events 2017

The TUT Rare Sound Events 2017 (TUT-Rare 2017)[2] [136] consists of isolated sounds of three different target event classes (respectively, baby crying, glass breaking and gunshot) and 30-second long recordings of everyday acoustic scenes to serve as background, such as park, home, street, cafe, train, etc. [56]. In this case we consider a *monophonic*-SED, since the sound events are artificially mixed with the background sequences without overlap. For further details, please refer to Section 5.2.3.

6.2.4 Comparative Algorithms

Since the datasets we used were employed to develop and evaluate the algorithms proposed from the participants of the DCASE challenges, we can com-

[1] `http://www.cs.tut.fi/sgn/arg/dcase2016/`
[2] `http://www.cs.tut.fi/sgn/arg/dcase2017/`

pare our results with the most recent approaches in the state-of-the-art. In addition, each challenge task came along with a baseline method that consists in a basic approach for the SED. It represents a reference for the participants of the challenges while they were developing their systems.

TUT-SED 2016

The baseline system is based on mel frequency cepstral coefficients (MFCC) acoustic features and multiple GMM-based classifiers. In detail, for each event class, a binary classifier is trained using the audio segments annotated as belonging to the model representing the event class, and the rest of the audio to the model which represents the negative class. The decision is based on likelihood ratio between the positive and negative models for each individual class, with a sliding window of one second. To the best of our knowledge, the most performing method for this dataset is the algorithm we showed in Section 6.1.

TUT-SED 2017

In this case the baseline method relies on an MLP architecture using 40 LogMels as audio representation [136]. The network is fed with a feature vector comprehending 5-frame as temporal context. The neural network is composed of two dense layers of 50 hidden units per layer with the 20% of dropout, while the network output layer contains K sigmoid units (where K is the number of classes) that can be active at the same time and represent the network prediction of event activity for each context window. The state of the art algorithm is based on the CRNN architecture [161]. The authors compared both monaural and binaural acoustic features, observing that binaural features in general have similar performance as single channel features on the development dataset although the best result on the evaluation dataset is obtained using monaural LogMels as network inputs. According to the authors, this can suggest that the dataset was possibly not large enough to train the CRNN fed with this kind of features.

TUT-Rare 2017

The baseline [56] and the state-of-the-art methods of the DCASE 2017 challenge (Rare-SED) were based on a very similar architectures to that employed for the TUT-SED 2016 and described above. For the baseline method, the only difference relies in the output layer, which in this case is composed of a single sigmoid unit. The first classified algorithm [47] takes 128 LogMels as input and process them frame-wise by means of a CRNN with 1D filters on the first stage.

6.2.5 Neural Network configuration

We performed a hyperparameter search by running a series of experiments over predetermined ranges. We selected the configuration that leads, for each network architecture, to the best results from the cross-validation procedure on the development dataset of each task, and used this architecture to compute the results on the corresponding evaluation dataset.

The number and shape of convolutional layers, the non-linear activation function, the regularizers in addition to the capsules dimensions and the maximum number of routing iterations have been varied for a total of 100 configurations. Details of searched hyperparameters and their ranges are reported in Table 6.6. The neural networks training was accomplished by the AdaDelta stochastic gradient-based optimization algorithm [140] for a maximum of 100 epochs and batch size equal to 20 on the margin loss function. The optimizer hyperparameters were set according to [140] (i.e., initial learning rate $lr = 1.0$, $\rho = 0.95$, $\epsilon = 10^{-6}$). The trainable weights were initialized according to the *glorot-uniform* scheme [162] and an early stopping strategy was employed during the training in order to avoid overfitting. If the validation ER did not decrease for 20 consecutive epochs, the training was stopped and the last saved model was selected as the final model. In addition, dropout and L_2 weight normalization (with $\lambda = 0.01$) have been used as weights regularization techniques [68]. The algorithm has been implemented in the Python language using Keras [141] and Tensorflow [142] as deep learning libraries, while Librosa [163] has been used for feature extraction.

For the CNN models, we performed a similar random hyperparameters search procedure for each dataset, considering only the first two blocks of the Table 6.6 and by replacing the capsule layers with feedforward layers with *sigmoid* activation function.

On TUT-SED 2016 and 2017 datasets, the event activity probabilities are simply thresholded at a fixed value equal to 0.5, in order to obtain the binary activity matrix used to compute the reference metric. On the TUT-Rare 2017 the network output signal is processed as proposed in [164], thus it is convolved with an exponential decay window then it is processed with a sliding median filter with a local window-size and finally a threshold is applied.

6.2.6 Results

In this section, we present the results for all the datasets and experiments described in Section 4.3.3. The evaluation of Capsule and CNNs based methods have been conducted on the development sets of each examined dataset using random combinations of hyperparameters given in Table 6.6.

Parameter	Range	Distribution
Batch Normalization	[yes - no]	random choice
CNN layers Nr.	[1 - 4]	uniform
CNN kernels Nr.	[4 - 64]	log-uniform
CNN kernels dim.	[3×3 - 8×8]	uniform
Pooling dim.	[1×1 - 2×5]	uniform
CNN activation	[tanh - relu]	random choice
CNN dropout	[0 - 0.5]	uniform
CNN L2	[yes - no]	random choice
Primary Capsules Nr. M	[2 - 8]	uniform
Primary Capsules kernels dim.	[3×3 - 5×5]	uniform
Primary Capsules dimension J	[2 - 16]	uniform
Detection Capsules dimension G	[2 - 16]	uniform
Capsules dropout	[0 - 0.5]	uniform
Routing iterations	[1 - 5]	uniform

Table 6.6: Hyperparameters optimized in the random-search phase and the resulting best performing models.

6.2.7 TUT-SED 2016

Results on TUT-SED 2016 dataset are shown in Table 6.8, while Table 6.7 reports the configurations which yielded the best performance on the evaluation dataset. All the found models have ReLU as non-linear activation function and use dropout technique as weight regularization, while the batch-normalization applied after each convolutional layer seems to be effective only for the CapsNet. In Table 6.8 results are reported considering each combination of architecture and features we evaluated. The best performing setups are highlighted with bold face. The use of STFT as acoustic representation results to be beneficial for both the architectures with respect to the LogMels. In particular, the CapsNet obtains the lowest ER on the cross-validation performed on Development dataset when is fed by the binaural version of such features. On the two scenarios of the evaluation dataset, a model based on CapsNet and binaural STFT obtains an averaged ER equal to 0.69, which is largely below both the challenge baseline [56] (-0.19) and the best score reported in literature [145] (-0.10). The comparative method based on CNNs seems not to fit at all when LogMels are used as input, while the performance is aligned with the challenge baseline based on GMM classifiers when the models are fed by monaural STFT. This discrepancy can be motivated by the enhanced ability of CapsNet to exploit small training datasets, in particular due to the effect of the routing mechanism on the weight training. In fact, the TUT-SED 2016 is composed of a small amount of audio and the sounds events occur sparsely (i.e., only 49

minutes of the total audio contain at least one event active), thus, the overall results of the comparative methods (CNNs, Baseline and SoA) on this dataset are quite low compared to the other datasets.

Another CapsNet property that is worth to highlight is the lower number of free parameters that compose the models compared to evaluated CNNs. As shown in Table 6.7, the considered architectures have 267K and 252K free parameters respectively for the "Home" and the "Residential area" scenario. It is a relatively low number of parameters to be trained (e.g., a popular deep architecture for image classification such as AlexNet [148] is composed of 60M parameters), and the best performing CapsNets of each considered scenario have even less parameters with respect to the CNNs (-22% and -64% respectively for the "Home" and the "Residential area" scenario). Thus, the high performance of CapsNet can be explained with the architectural advantage rather than the model complexity. In addition, there can be a significant performance shift for the same type of networks with the same number of parameters, which means that a suitable hyperparameters search action (e.g., number of filters on the convolutional layers, dimension of the capsule units) is crucial in finding the best performing network structure.

	TUT-SED 2016				**TUT-SED 2017**	
	Home		Residential		Street	
	CapsNet	CNN	CapsNet	CNN	CapsNet	CNN
CNN kernels Nr.	$[32, 32, 8]$	$[64, 64, 16, 64]$	$[4, 16, 32, 4]$	$[64]$	$[4, 16, 32, 4]$	$[64, 64, 16, 64]$
CNN kernels dim.	6×6	5×5	4×4	5×5	4×4	5×5
Pooling dim. (F axis)	$[4, 3, 2]$	$[2, 2, 2, 2]$	$[2, 2, 2, 2]$	$[2]$	$[2, 2, 2, 2]$	$[2, 2, 2, 2]$
MLP layers dim.	-	$[85, 65]$	-	$[42, 54, 66, 139]$	-	$[85, 65]$
Primary Capsules Nr. M	8	-	7	-	7	-
Primary Capsules kernels dim.	4×4	-	3×3	-	3×3	-
Primary Capsules dimension J	9	-	16	-	16	-
Detection Capsules dimension G	11	-	8	-	8	-
Routing iterations	3	-	4	-	4	-
# Params	267K	343K	252K	709K	223K	342K

Table 6.7: Hyperparameters of the best performing models on the TUT-Polyphonic SED 2016 & 2017 Evaluation datasets.

Closer Look at Network Outputs

A comparative study on the neural network outputs, which are regarded as event activity probabilities is presented in Figure 6.4. The monaural STFT from a 40 seconds sequence of the "Residential area" dataset is shown along with event annotations and the network outputs of the CapsNet and the CNN best performing models. For this example, we chose the monaural STFT as input feature because generally it yields the best results over all the considered datasets. Figure 6.4 shows *bird singing* lasting for the whole sequence and correctly detected by both the architectures. When the *car passing by* event

TUT-SED 2016 - Home								
	Development				Evaluation			
Features	LogMels	Binaural LogMels	STFT	Binaural STFT	LogMels	Binaural LogMels	STFT	Binaural STFT
CNN	11.15	11.58	1.06	1.07	6.80	8.43	0.95	0.92
CapsNet	0.58	0.59	0.44	**0.39**	0.74	0.75	**0.61**	0.69
TUT-SED 2016 - Residential Area								
Features	LogMels	Binaural LogMels	STFT	Binaural STFT	LogMels	Binaural LogMels	STFT	Binaural STFT
CNN	3.24	3.11	0.64	1.10	2.36	2.76	1.00	1.35
CapsNet	0.36	0.34	0.32	**0.32**	0.72	0.75	0.78	**0.68**
TUT-SED 2016 - Averaged								
CNN	7.20	7.35	0.85	1.09	4.58	5.60	0.98	1.14
CapsNet	0.47	0.47	0.38	**0.36**	0.73	0.75	0.70	**0.69**
Baseline [56]			0.91				0.88	
SoA [145]			0.78				0.79	
TUT-SED 2017								
	Development				Evaluation			
Features	LogMels	Binaural LogMels	STFT	Binaural STFT	LogMels	Binaural LogMels	STFT	Binaural STFT
CNN	1.56	2.12	0.57	0.60	1.38	1.79	0.67	0.65
CapsNet	0.45	0.42	**0.36**	0.36	**0.58**	0.64	0.61	0.64
Baseline [136]			0.69				0.93	
SoA [161]			0.52				0.79	

Table 6.8: Results of best performing models in terms of ER on the TUT-SED 2016 & 2017 dataset.

overlaps the *bird singing*, the CapsNet detects more clearly its presence. The *people speaking* event is slightly detected by both the models, while the *object banging* activates the relative Capsule exactly only in correspondence of the event annotation. It must be noted that the dataset is composed of unverified manually labelled real-life recordings, that may present a degree of subjectivity, thus, affecting the training. Nevertheless, the CapsNet exhibits remarkable detection capability especially in the condition of overlapping events, while the CNN outputs are definitely more "blurred" and the event *people walking* is wrongly detected in this sequence.

6.2.8 TUT-SED 2017

The bottom of Table 6.8 reports the results obtained with the TUT-SED 2017. As in the TUT-SED 2016, the best performing models on the Development dataset are those fed by the Binaural STFT of the input signal. In this case we can also observe interesting performance obtained by the CNNs, which on the Evaluation dataset obtain a lower ER (i.e., equal to 0.65) with respect to the state-of-the-art algorithm [161], based on CRNNs. CapsNet confirms its effectiveness and it obtains lowest ER equal to 0.58 with LogMel features, although with a slight margin with respect to the other inputs (i.e., -0.03 compared to the STFT features, -0.06 compared to both the binaural version of LogMels and STFT spectrograms).

It is interesting to notice that in the development cross-validation, the Cap-

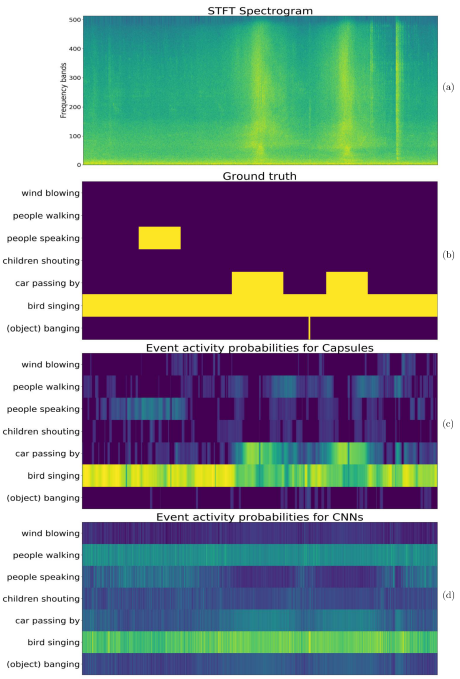


Figure 6.4: STFT Spectrogram of the input sequence (a), ground truth (b) and event activity probabilities for CapsNet (c) and CNN (d) from a sequence of test examples from TUT-SED 2016 dataset.

sNet models yielded significantly better performance with respect to the other reported approaches, while the CNNs have decidedly worse performance. On the Evaluation dataset, it was not possible to use the early-stopping strategy, thus the ER scores of the CapsNets suffer more relative deterioration with respect to the CNNs scores, symptom of the sensibility of the CapsNets to the number of training iterations.

Notwithstanding this weakness, the absolute performance obtained both with monaural and binaural spectral features is consistent and improves the state-of-the-art result, with a reduction of the ER of up to 0.21 in the best case. This is particularly evident in Figure 6.5, that shows the output of the two best performing systems for a sequence of approximately 20 seconds length which contains highly overlapping sounds. The event classes "people walking" and "large vehicle" are overlapped for almost all the sequence duration and they are well detected by the CapsNet, although they are of different nature: the "large vehicle" has a typical timber and is almost stationary, while the class "people walking" comprehend impulsive and desultory sounds. The CNN seems not to be able to distinguish between the "large vehicle" and the "car" classes, detecting confidently only the latter, while the activation corresponding "people

walking" class is modest. The presence of the "brakes squeaking" class, which has a specific spectral profile mostly located in the highest frequency bands (as shown in the spectrogram), is detected only by the CapsNet. We can assume this as a concrete experimental validation of the routing effectiveness.

The number of free parameters amounts to 223K for the best configuration shown in Table 6.7 and it is similar to those found for the TUT-SED 2016, which consists also in this case in a reduction equal to 35% with respect to the best CNN layout.

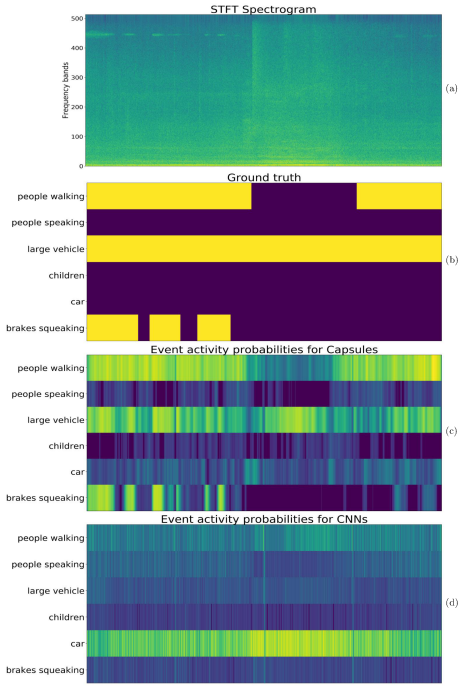


Figure 6.5: STFT Spectrogram of the input sequence (a), ground truth (b) and event activity probabilities for CapsNet (c) and CNN (d) from a sequence of test examples from TUT-SED 2017 dataset.

6.2.9 TUT-Rare SED 2017

The advantage given by the routing procedure to the CapsNet is particularly effective in the case of polyphonic SED. This is confirmed by the results obtained with the TUT-Rare SED 2017 which are shown in Table 6.10. In this case the metric is not anymore segment-based, but it is the event-based ER calculated using onset-only condition. We performed a separate random-search for each of the three sound event classes both for CapsNets and CNNs and we report the averaged score over the three classes. The setups that obtained the

best performance on the Evaluation dataset are shown in Table 6.9. This is the largest dataset we evaluated and its characteristic is the high unbalance between the amount of background sounds versus the target sound events.

From the analysis of partial results on the Evaluation set (unfortunately not included for the sake of conciseness) we notice that both architectures achieve the best performance on *glass break* sound (0.25 and 0.24 respectively for CNNs and CapsNet with LogMels features), due to its clear spectral fingerprint compared to the background sound. The worst performing class is the *gunshot* (ER equal to 0.58 for the CapsNet), although the noise produced by different instances of this class involves similar spectral components. The low performance is probably motivated due to the fast decay of this sound, which means that in this case the routing procedure is not sufficient to avoid confusing the *gunshot* with other background noises, especially in the case of dataset unbalancing and low event-to-background ratio. A solution to this issue can be found in the combination of CapsNet with RNN units, as proposed in [45] for the CNNs which yields an efficient modelling of the *gunshot* by CRNN and improves the detection abilities even in polyphonic conditions. The *babycry* consists of short, harmonic sounds is detected almost equally from the two architectures due to the property of frequency shift invariance given by the convolutional kernel processing.

Finally, the CNN shows better generalization performance to the CapsNet, although the ER score is far from state-of-the-art which involves the use of the aforementioned CRNNs [47] or a hierarchical framework [164]. In addition, in this case are the CNN models to have a reduced number of weights to train (36%) with respect the CapsNets, except for the "gunshot" case but, as mentioned, it is also the configuration that gets the worst results.

TUT-Rare SED 2017 Monophonic SED						
	Babycry		Glassbreak		Gunshot	
	CapsNet	CNN	CapsNet	CNN	CapsNet	CNN
CNN kernels Nr.	$[16, 64, 32]$	$[16, 32, 8, 16]$	$[16, 64, 32]$	$[16, 32, 8, 16]$	$[16, 16]$	$[16, 64, 32, 32]$
CNN kernels dim.	6×6	8×8	6×6	8×8	8×8	7×7
Pooling dim. (F axis)	$[4, 3, 2]$	$[3, 3, 2, 2]$	$[4, 3, 2]$	$[3, 3, 2, 2]$	$[5, 2]$	$[5, 4, 2, 1]$
MLP layers dim.	-	$[212, 67]$	-	$[212, 67]$	-	$112, 51$
Primary Capsules Nr. M	7	-	7	-	8	-
Primary Capsules kernels dim.	3×3	-	3×3	-	3×3	-
Primary Capsules dimension J	8	-	8	-	8	-
Detection Capsules dimension G	14	-	14	-	6	-
Routing iterations	5	-	5	-	1	-
# Params	131K	84K	131K	84K	30K	211K

Table 6.9: Hyperparameters of the best performing models on the TUT-Rare 2017 Monophonic SED Evaluation datasets.

TUT-RareSED 2017 - Monophonic SED				
	Development		Evaluation	
Features	LogMels	STFT	LogMels	STFT
CNN	0.29	0.21	**0.41**	0.46
CapsNet	**0.17**	0.20	0.45	0.54
Baseline [56]	0.53		0.64	
Hierarchic CNNs [164]	0.13		0.22	
SoA [47]	**0.07**		**0.13**	

Table 6.10: Results of best performing models in terms of ER on the TUT-RareSED 2017 dataset.

6.2.10 Alternative Dynamic Routing for SED

We observed that the original routing procedure implies the initialization of the coefficients β_{ij} to zero each time the procedure is restarted, i.e, after each input sample has been processed. This is reasonable in the case of image classification, for which the CapsNet has been originally proposed. In the case of audio task, we clearly expect a higher correlation between samples belonging to adjacent temporal frames $\mathbf{X}$. We thus investigated the chance to initialize the coefficients β_{ij} to zero only at the very first iteration, while for subsequent $\mathbf{X}$ to assign them the last values they had at the end of the previous iterative procedure. We experimented this variant considering the best performing models of the analyzed scenarios for polyphonic SED, taking into account only the systems fed with the monaural STFT. As shown in Table 6.11, the modification we propose in the routing procedure is effective in particular on the evaluation datasets, conferring improved generalization properties to the models we tested even without accomplishing a specific hyperparameters optimization.

6.2.11 Conclusion

In this section, we proposed to apply the CapsNet architecture to the polyphonic SED task.

Part of the novelty of this work resides in the adaptation of the CapsNet architecture for the audio event detection task, with a special care on the input data, the layers interconnection and the regularization techniques. The routing procedure is also modified to confer a more appropriate acoustic rationale, with a further average performance improvement of 6% among the polyphonic-SED tasks.

An extensive evaluation of the algorithm is proposed with comparison to recent state-of-the-art methods on three different datasets. The experimental results demonstrate that the use of dynamic routing procedure during the training is effective and provides significant performance improvement in the

	Development		Evaluation	
TUT-SED 2016 - Home				
CapsNet	0.44		0.61	
CapsNet - NR	0.41	-6.8%	**0.58**	-4.9 %
TUT-SED 2016 - Residential				
CapsNet	0.32		0.78	
CapsNet - NR	0.31	-3.1%	**0.72**	-7.7 %
TUT-SED 2016 - Average				
CapsNet	0.38		0.70	
CapsNet - NR	0.36	-5.3%	**0.65**	-7.1 %
TUT-SED 2017 - Street				
CapsNet	0.36		0.61	
CapsNet - NR	0.36	-0.0%	**0.58**	-4.9 %

Table 6.11: Results of test performed with our proposed variant of routing procedure.

case of overlapping sound events compared to traditional CNNs, and other established methods in polyphonic SED. Interestingly, the CNN-based method obtained the best performance in the monophonic SED case study, thus emphasizing the suitability of the CapsNet architecture in dealing with overlapping sounds. We showed that this model is particularly effective with small sized datasets, such as TUT-SED 2016 which contains a total 78 minutes of audio for the development of the models of which one third is background noise. Furthermore, the network trainable parameters are reduced with respect to other deep learning architectures, confirming the architectural advantage given by the introduced features also in the task of polyphonic SED. Despite the improvement in performance, we identified a limitation of the proposed method. As presented in Section 6.2.6, the performance of the CapsNet is more sensible to the number of training iterations. This affects the generalization capabilities of the algorithm, yielding a greater relative deterioration of the performance on evaluation datasets with respect to the comparative methods.

The results we observed in this work are consistent with many other classification tasks in various domains [165, 166, 167], and prove that the CapsNet is an effective approach which enhances the well-established representation capabilities of the CNNs also in the audio field. As a future work, regularization methods can be investigated to overcome the lack of generalization which seems to affect the CapsNets. Furthermore, regarding the SED task the addition of recurrent units may be explored to enhance the detection of particular (i.e.,

inpulsive) sound events in real-life audio and the recently-proposed variant of routing, based on the Expectation Maximization algorithm (EM) [143], can be investigated in this context.

Chapter 7

Other contributions

7.1 Acoustic Novelty Detection with Adversarial Autoencoders

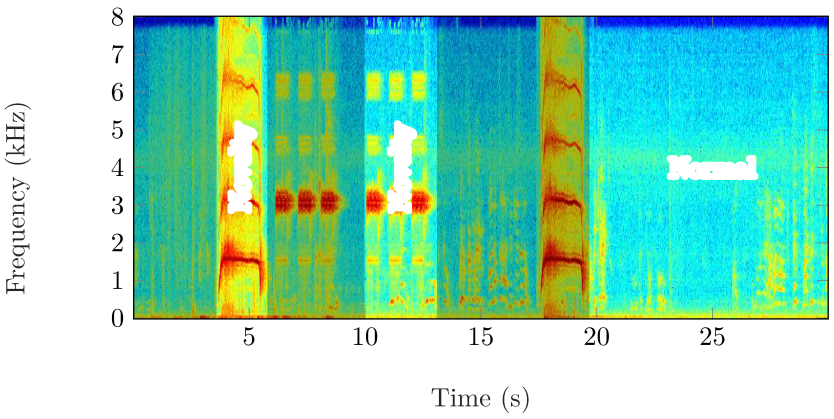


Figure 7.1: Spectrogram of an audio signal with novel and normal data.

Novelty detection is the task of recognising events the differ from a model of normality. This paper proposes an acoustic novelty detector based on neural networks trained with an adversarial training strategy. The proposed approach is composed of a feature extraction stage that calculates Log-Mel spectral features from the input signal. Then, an autoencoder network, trained on a corpus of "normal" acoustic signals, is employed to detect whether a segment contains an abnormal event or not. A novelty is detected if the Euclidean distance between the input and the output of the autoencoder exceeds a certain threshold. The innovative contribution of the proposed approach resides in the training procedure of the autoencoder network: instead of using the conventional training procedure that minimises only the Minimum Mean Squared Error loss function, here we adopt an adversarial strategy, where a discriminator network is trained to distinguish between the output of the autoencoder and data sampled

from the training corpus. The autoencoder, then, is trained also by using the binary cross-entropy loss calculated at the output of the discriminator network.

The performance of the algorithm has been assessed on a corpus derived from the PASCAL CHiME dataset. The results showed that the proposed approach provides an F1-score equal to 93.28%, with a relative performance improvement equal to 0.26% compared to the standard autoencoder. The significance of the improvement has been evaluated with a one-tailed z-test and resulted significant with $p < 0.001$. The presented approach thus showed promising results on this task and it could be extended as a general training strategy for autoencoders if confirmed by additional experiments.

7.1.1 Details

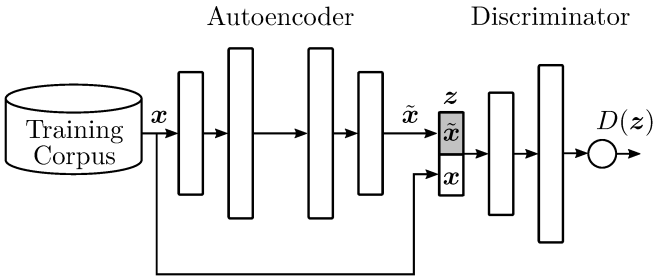
In this section, we are not interested in the generative capabilities of the network, since in the novelty detection task the objective is to minimise the error made by the autoencoder in reconstructing normal data. Thus, the architecture and the training strategy are different from [64]: the discriminator network is trained to discriminate between data from a training set and data reconstructed by the autoencoder (Figure 7.2a). The final layer of the discriminator is a single neuron with sigmoid activation function and its output represents the probability of the data of being sampled from the training set. At the end of the training phase, the discriminator network should not be able to distinguish training set data and reconstructed data, i.e., its output should be constantly equal to 0.5.

Similarly to [63], the training process can be viewed as a min-max game between the autoencoder and the discriminator network. Let $\boldsymbol{x}$ be an input feature vector, $\tilde{\boldsymbol{x}} = A(\boldsymbol{x})$ the output of the autoencoder, and $D(\boldsymbol{x})$ the output of the discriminator network, i.e., the probability of $\boldsymbol{x}$ of being sampled from the training data. The training procedure is composed of three main phases: the first phase and the third phase consist in updating the autoencoder and the discriminator respectively by minimising the reconstruction error and the classification error (binary cross-entropy). The middle phase incorporates the interaction between the two networks: the autoencoder weights are updated based on the output the discriminator network.

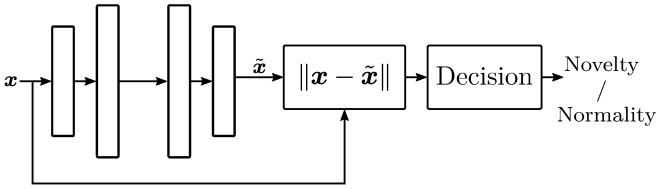
Whether a feature vector is a novel sound or a normal background sound is determined by calculating the reconstruction error, i.e., the Euclidean distance between the output produced by the autoencoder and the input data itself. If the distance exceeds a predefined threshold, the input data is classified as novelty, otherwise as normal. The threshold is calculated as follows: for each input sequence the threshold θ is calculated with the following expression:

$$\theta = \beta \cdot \text{median}\{e(1), \ldots, e(N)\}, \tag{7.1}$$

where $e(i)$ is reconstruction error of feature vector i in the signal, N is the number of features, and $\beta \in [1, 2]$ is a predefined constant.



(a) Proposed architecture for training an autoencoder with an adversarial discriminator network.



(b) Novelty detection phase.

Figure 7.2: The proposed approach for novelty detection with adversarial autoencoders. For the sake of simplicity, the feature extraction stage is not shown.

7.2 Few-shot Siamese Neural Networks employing Audio features for Human-Fall Detection

Nowadays, the detection of human fall is a problem recognized by the entire scientific community. Methods that have good performance use human falls samples in the train set, while methods that do not use it, can only work well under certain conditions. Since examples of human falls are very difficult to retrieve, there is a strong need to develop systems that can work well event with few or no data to be used for their training phase. In this section, we show a first study on few-shot learning Siamese Neural Network applied to human falls detection by using audio signals. This method has been compared with algorithms based on SVM and OCSVM, all evaluated starting from the same conditions. The proposed approach is able to learn the differences between signals belonging to different classes of events. In classification phase, using only one human fall signal as a template, it achieves about 80% of $F_1 - Measure$ related to the human fall class, while the SVM based method gets around 69%, when it is trained in the same data knowledge conditions.

7.2.1 Proposed Approach

In this section we propose a Siamese Neural Network able to learn a latent representation of an audio event. In particular, a SNN is composed of two twin networks with binded weights. A pair of inputs is provided to the system, one to each twin network. Downstream, the network maps these inputs into two different representation vectors. Then, a certain type of distance between those two representations is computed. In this section, euclidean distance was used. In Figure 7.3, are reported two examples of mel-spectrograms: the spectrum that is given as input to the function first network represents a chair that is overturned. The other inputs instead represent a human fall. As can be seen, the signals are not distinguishable at a glance, thus we think that the differential approach of the SNN, described below, seems to be appropriate.

Our Siamese network has been trained on a corpus of labelled object fall events and not including any human fall. Pairs of events belonging to the same class correspond to the positive examples while pairs of events belonging to the different class a negative one. In particular, the term few-shot comes from the fact that although, in this case study, human falls have not been used for training, some of them are used in the optimization phase, before the final test.

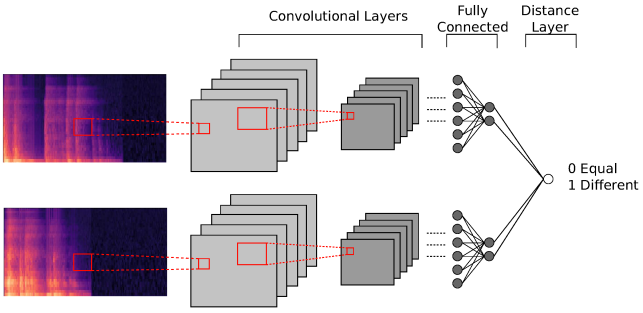


Figure 7.3: Proposed approach for Human-Fall Detection.

7.2.2 Results

The results obtained for each method are reported in Figure 7.4. The figure shows the $F_1 - Measure$, false negative rate (miss rate) and false positive rate (false alarm rate) referred to the human fall class. It is clear that the supervised SVM method outperforms all other in terms of $F_1 - Measure$ as expected. The OCSVM instead is the worst method if used in this context, because its training procedure does not include any human fall, but the normality model is composed of other types of falls, making it difficult to identify the human fall as "novelty". The Siamese and the SVM-unbalanced, which start from the same data for the training, are classified in the intermediate positions as expected. However, we note that the proposed approach achieves a better result, exceeding the $F_1 - Measure$ of SVM-unbalanced of about 11%.

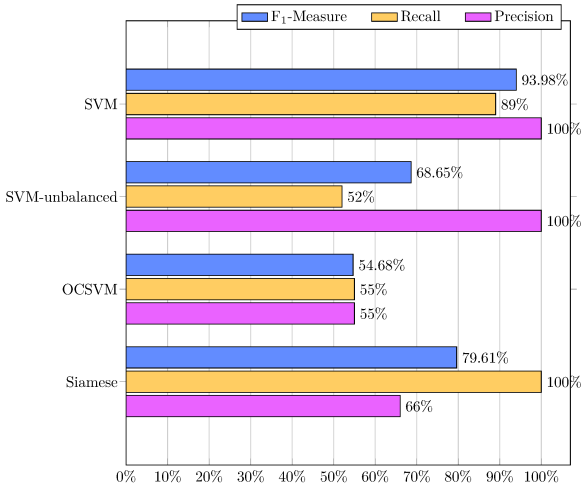


Figure 7.4: $F_1 - Measure$, precision and recall: the metrics are referred to the human fall class

7.3 Generative Raw Audio Synthesis

Biologically-inspired algorithms such as artificial neural networks have been used extensively by computer music researchers, for generative music and algorithmic composition. The recent introduction of raw audio Machine Learning (ML) techniques, however, represents a significant leap because they seem to be able to learn both high-level (event) and low-level (timbre) information at once. Employing such techniques for creative purposes is very challenging at this early stage since there is lack of method, experience, and their computational cost is very high.

To the best of our knowledge, three main architectures have been proposed to learn and reproduce features from raw audio signals, namely WaveNet [168], sampleRNN [169], and WaveGan [170]. Recently, another architecture, named FFTnet, has been proposed [171] that draws some concepts from WaveNet but shows a lower computational cost.

These architectures show the ability of learning features from raw audio data and generate similar signals drawing from a *latent* representation of these data directly in the time domain. This seems to be possible thanks to a hierarchical representation of the signal. In other words, architectures such as sampleRNN and WaveNet process the signal at multiple levels, enabling the network to store significant features from the micro-scale (sample level) up to the macro-scale (1 second and more). Experiments with these two architectures show that after training the network with a few hours of piano music, the networks are able to reproduce piano tones in a somewhat organized way. They are, thus, able to learn the timbre of the piano, the piano tone envelope (attack and decay), and finally, they learn that piano notes have rhythmical patterns, can be played in clusters, chords and sequences.

Stemming from these results, the authors decided to explore the possibility of running these algorithms for tone generation. At the time of writing, these algorithms are not suitable for synthesis from a score. However, they have the autonomous ability of generating a sequence of sounds, similarly to generative algorithms, but working at several time scales.

7.3.1 Algorithm Selection

A careful evaluation of currently available machine learning algorithms for raw-audio generation has been carried on. The evaluated algorithms were:

- WaveNet [168],
- sampleRNN [169],
- WaveGAN [170].

At the time of setting up the experiments the FFTNet algorithm [171] was not yet published.

The WaveNet architecture allows for modeling complex data such as music and speech. It is based on a stack of causal dilated convolutional layers for feature extraction. The dilated convolutions are key to extract features at different levels (closely resemblant to a dyadic wavelet filter bank [171]), however, the experiments in [172] show that the use of dilated convolutions alone do not allow modelling complex raw audio sequences, thus, residual blocks and the use of skip connections after the convolutional layers are necessary to speed up convergence and allow the training of a deep model. The ability of the network to model meaningful sequences of samples relies on causal probability conditioning, i.e. the last output sample is conditioned by the probability of all previous output samples. Additionally, it has been shown with speech synthesis that the output sample probability can be conditioned with, e.g. speaker timbre and phonemes. In our case, however, we are not interested in leveraging this feature, leaving the network to generate freely.

The main issue preventing adoption of WaveNet is its high computational cost and informal reports of their performance seems to confirm this. To test the feasibility of the approach and verify these claims we adapted one of the several open-source implementations of WaveNet currently available. We conducted preliminary experiments on a Titan X GPU employing Theano as backend. After 72 hours of training using the same piano dataset suggested by [168] the network was still unable to provide results vaguely similar to the ones shown by the authors of [168]. Furthermore, some of the hyperparameters were extrapolated from the paper, but most of them are not available, so a hyperparameters search would be required, increasing the training times far over our available resources.

A very different approach is followed by the authors of sampleRNN [169]. This architecture is based on the same principle of causal probability conditioning, however, in this case the network consists of several tiers of recurrent neural networks (RNN) working at different temporal resolutions. Each RNN conditions the RNN at the lower tier. SampleRNN has been tested with both speech and music database. Both databases are a few hours long.

We also performed preliminary experiments employing the open-source implementation provided by the authors of [169] on the same hardware reported above. This implementation is based on Theano and Lasagne. Results are on par with those provided by the authors and training times can have length of 1-4 days depending on the input material and the degree of fitness that is required.

Finally, we considered WaveGAN for sound generation. This architecture is based on the generative adversarial networks (GAN) paradigm. The authors

of [170] devised two models, WaveGAN, working on the raw audio with 1D convolutions and SpecGAN, working with 2D spectrograms. The outcomes of their work are very promising, since these two architecture are able to learn from a much shorter database than WaveNet and sampleRNN. However, they are inherently designed to synthesized audio of a specific length (16384 samples for WaveGAN or a 128x128 spectrogram for SpecGAN). For this reason they are not well suited for generative audio. Running a WaveGAN several times would introduce issues in concatenating each output.

At the end of this evaluation stage, we decided to use sampleRNN.

7.3.2 Input Material and Dataset Creation

The input material for this work has been provided by artist *Økapi* in form of *stems* of his musical project Rima Glottidis. The main goal of the work was to generate vocal textures from his material to be arranged in form of a site-specific sound installation. The stems came from cut recordings of male and female speech of several speakers showing proper Italian pronunciation vs. unclear spelling and regional accent.

Additionally they presented bell tones, singing choirs, sung phonemes, and silence, revealing part of the musical structure of the original project. The whole material length was 14 minutes. A subset of this material, totalling 7 minutes, was obtained by leaving only the speaking voices and cutting all other musical material.

From previous experience with sampleRNN the input material was judged to be too short. Informal experiments on voice synthesis were conducted in 2016 by the first author showing that sampleRNN can faithfully learn the vocal timbre of a speaker only from a sufficiently large and homogeneous dataset of speech. Specifically, a 3 hours dataset gathered from publicly available speeches of a well known Italian comedian and politician was used to train a three-tier sampleRNN model. At the end of the training, a convincing babbling was obtained with random phonemes. Most subjects presented with the synthesized speech could recognize the identity of the public figure. Reducing the dimension of the dataset to 1 hour or less degraded the performance of the network making the utterances noise-like.

To increase the chances to obtain interesting audio material from sampleRNN, other audio material has been selected. Data augmentation has been tested to enlarge the training corpus without adding spurious material that would reduce the portion of material from *Økapi*. Data augmentation has been done by pitch shifting audio data by -3 and +3 semitones.

Additionally, 22 minutes of choral works from Arvo Pärt were added. A subset of this dataset has been randomly selected, totalling 2 minutes. These

datasets have been used for training 2 different sampleRNN models with hyperparameters suggested by the authors of [169]. Training of these models was conducted with a Titan X GPU and left running for 2 days for each one, approximately lasting for 220 epochs (approx $3 \cdot 10^6$ iterations). For each epoch, several 10s-long pieces of randomly generated material were extracted for later use.

7.3.3 Results

Discussion about the network outcomes will be qualitative, as there is no objective means to assess the quality of the material. The loss employed by sampleRNN, the categorical crossentropy, does not clearly state the quality of the sound material. The largest excursion of the loss seen during training ranged from 5 to 3 for the training set and from 4.5 to 3.9 for the validation set. Notwithstanding this, the first tones generated were pure noise, while the last ones had some of the character of the original dataset. The training loss did not necessarily decrease with time and samples with interesting features emerged at different epochs. Most notably, samples generated at a given epoch or at consecutive epochs, have all very similar features that later disappear with the training.

The outcomes from these trainings can be divided in several classes: speech-like babbling (a), whistling (b), rumbles and impulses (c) or mixtures of the above. Figure 7.5 shows spectrograms of each class. Please note that the whistling tones do not appear in the output generated from the Rima Glottidis speech-only subset. It was noted, indeed, that these have the same frequency of bell tones found in the complete Rima Glottidis dataset. Bell tones appear on the 5% of the Rima Glottidis dataset. This means that very simple and repetitive material can be easily reproduced by sampleRNN even though it does appear on a small portion of the dataset.

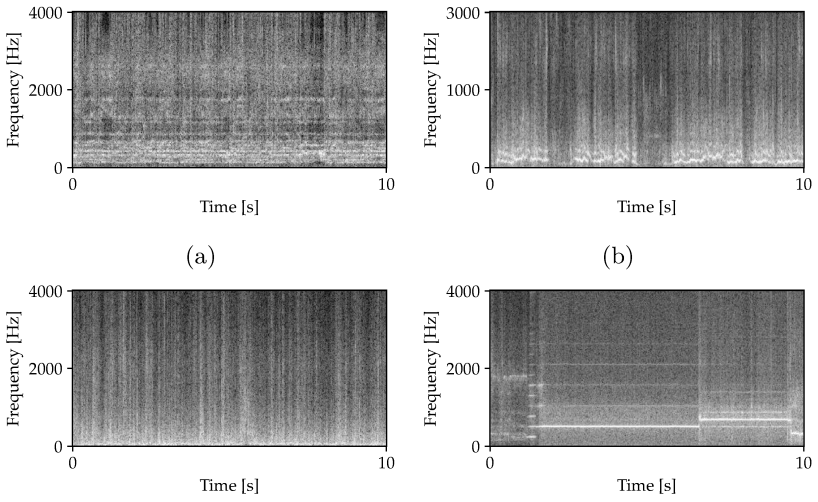


Figure 7.5: Spectrograms of several classes of outputs from sampleRNN: choir textures (a), speech-like babbling (b), rumble (c), whistle (d). The choir textures are obtained from the Chorales training set, while the others are from the Rima Glottidis training set. The main two tones in Figure (d) are a C5 and a F5.

Chapter 8

Conclusions

In this dissertation, Deep Learning approaches for Sound Event Detection and Classification have been studied.

Deep Neural Networks (DNNs) have been largely investigated and exploited in this thesis, driven by the promising results achieved from these architecture for the computational analysis of sound scenes and events in the recent years. This task is characterized by many issues, especially when the systems are developed for real-life environments (i.e., when some assumptions made for controlled laboratory conditions are not valid anymore). Among these issues there are: unbalanced datasets available to train models; different acquisition setups (microphones, environment, ...); acoustic disturbance, such as background noise, reverberation and cross-talk; polyphony, i.e., the simultaneous occurrence of target sound events. Indeed, the choice of DNNs models for this aim is motivated by their proved generalization capability, which is commonly weak in classical classification algorithms based on support vector machines or statistical modelling.

The first three chapters of this thesis aim to give an overview of the main motivations behind this thesis, and to describe in general the elements which compose the design process of a system. Indeed, Section 1 gives an introduction of the computational methods for the analysis of sound events applications and recent results in DNN-based approaches are vastly discussed. After that, in Section 2 a theoretical description of the principal DNN architectures is given. In Section 3 a brief description of typical dataset characteristics (acquisition, labelling and augmentation) is provided, in combination with standard procedures to evaluate the systems outputs.

Within all the tasks explored by the scientific community, this thesis has been focused on a subset of problems. They can be grouped into three main categories: the sound event detection (cf. Section 5), the sound event classification (cf. Section 4) and the polyphonic sound event detection (cf. Section 6), which can be seen as a combination of the previous ones. They have been faced by means of Deep Learning approaches and in all cases the proposed algorithms have been compared to the most recent state of the art benchmarks.

In particular, most of the time systems have been designed in the occasion of international challenges [56, 136, 98, 57]. This allowed access to datasets without the need to invest huge resources for their collection, and to be able to compare systems proposed by the most competitive research teams on a common basis. When this was not possible, we proceeded to collect the data ourselves, following the best-practices described in Section 3.

The results of these studies confirm the effectiveness of the DNNs, in particular comparing the performance of novel architectures such as CapsNets in the case study of polyphonic sound event detection (cf. Section 6.2), or in the rather new field of road surface roughness classification (cf. Section 4.2), where the acoustic differences between samples belonging to the two target class are minimal.

8.1 Future Perspective

Future works will concern the testing of the proposed algorithms against new environments, along with ad hoc solutions for data augmentation, transfer learning or domain adaptation.

Furthermore, it could be interesting to combine some of the proposed algorithms, in order to obtain joint models gradually more similar the concept of "Artificial Intelligence". For example, this can be applied to the analysis of snore signals: the detection (cf. Section 5.1) and the classification (cf. Section 4.1) system described in this dissertation can be combined into a unique model used in an unobtrusive diagnosis procedure.

On the other hand, end-to-end approaches can be evaluated for particular context where traditional sound representations (such as mel-scaled spectrograms) may not be the best choices, and it could be more convenient to design a DNN networks able to automatically learn the best representation of the directly from the raw signal. Detection of specific sound events (cf Section 5.2), or bioacoustic monitoring (cf. Section 4.3) could be addressed with this technique. In addition, in this case the same feature extraction procedure performed by the network could be suitable for other audio-related case studies.

Last but not least, the next step for sound event analysis systems is to even out the performance gap between the fully supervised and the training *weakly* labelled data, i.e, labels that merely indicate whether a particular sound event is present within a recording, without specifying additional details, such as the precise location of the event or even the number of times it occurs.

8.2 A Zoom Out

The studies which have been conducted on the presented applications of the deep neural networks confirm their potential in many different contexts. This is particularly valuable when the DNNs are employed to extract a high-level information dependent from characteristics difficult to define a-priori, such as in the case of physiological sounds (e.g, Snoring, etc.,), the studies on the Affective Computing or cases which would require in-depth acoustic studies (i.e., the Road Surface Roughness classification).

The main limit to the diffusion of these technologies mainly concerns the availability and quality of datasets. If in the near future efforts will be made to collect and label data possibly also with semi-supervised techniques, it will be possible to use these databases to pre-train DNNs specifically on audio signals and subsequently to specialize them for specific contexts, as is currently happening in the image processing.

List of Publications

[1] E. Marchi, F. Vesperini, S. Squartini, and B. Schuller, "Deep recurrent neural network-based autoencoders for acoustic novelty detection," *Computational Intelligence and Neuroscience*, 2016.

[2] F. Vesperini, P. Vecchiotti, E. Principi, S. Squartini, and F. Piazza, "Localizing speakers in multiple rooms by using deep neural," *Computer Speech and Language*, 2017.

[3] F. Vesperini, L. Gabrielli, E. Principi, and S. Squartini,"Polyphonic sound event detection by using capsule neural networks," *Journal of Selected Topics in Signal Processing*, 2018, submitted.

[4] E. Marchi, F. Vesperini, F. Eyben, S. Squartini, and B. Schuller, "A Novel Approach for Automatic Acoustic Novelty Detection Using a Denoising Autoencoder with Bidirectional LSTM Neural Networks," *Proc. of ICASSP*, Brisbane, Australia, 19-24 Apr. 2015, IEEE.

[5] E. Marchi, F. Vesperini, F. Weninger, F. Eyben, S. Squartini, and B. Schuller, "Non-Linear Prediction with LSTM Recurrent Neural Networks for Acoustic Novelty Detection," *Proc. of IJCNN*, Killarney, Ireland, 12-16 Jul. 2015, IEEE.

[6] F. Vesperini, P. Vecchiotti, E. Principi, S. Squartini, and F. Piazza, "Deep neural networks for multi-room voice activity detection: Advancements and comparative evaluation," *Proc. of IJCNN*, Vancouver, Canada, 24-29 Jul. 2016, IEEE, pp. 3391–3398.

[7] M. Gasparini, F. Vesperini, S. Cecchi, S. Squartini, F. Piazza, and R. Toppi, "Combining evolution strategies and neural network procedures for compression driver design," *Proc. of IJCNN*, Vancouver, Canada, 24-29 Jul. 2016, IEEE, pp. 3385–3390.

[8] P. Vecchiotti, F. Vesperini, E. Principi, S. Squartini, and F. Piazza, "Convolutional neural networks with 3-D kernels for voice activity detection in a multiroom environment," *Multidisciplinary Approaches to Neural Computing*, pp. 161–170. Springer, 2018.

[9] F. Vesperini, P. Vecchiotti, E. Principi, S. Squartini, and F. Piazza, "A neural network based algorithm for speaker localization in a multi-room environment," *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*. IEEE, 2016, pp. 1–6.

[10] E. Principi, F. Vesperini, S. Squartini, and F. Piazza, "Acoustic novelty detection with adversarial autoencoders," *Proc. of IJCNN*, Anchorage, Alaska, 14-19 May 2017, IEEE, pp. 3324–3330.

[11] M. Valenti, D. Tonelli, F. Vesperini, E. Principi, and S. Squartini, "A neural network approach for sound event detection in real life audio," *Proc. of EUSIPCO*, Kos, Greece, Sept. 2017, IEEE.

[12] L. Gabrielli, C. E. Cella, F. Vesperini, D. Droghini, E. Principi, and S. Squartini, "Deep learning for timbre modification and transfer: An evaluation study," *Proc. of 144th AES*, Milan, Italy, 24-26 May 2018, Audio Engineering Society.

[13] L. Ambrosini, L. Gabrielli, F. Vesperini, S. Squartini, and L. Cattani, "Deep neural networks for road surface roughness classification from acoustic signals," *Proc. of 144th AES*, Milan, Italy, 24-26 May 2018, Audio Engineering Society.

[14] F. Vesperini, A. Galli, L. Gabrielli, E. Principi, and S. Squartini, "Snore sounds excitation localization by using scattering transform and deep neural networks," *Proc. of IJCNN*, Rio de Janeiro, Brasil, 8-13 Jul. 2018, IEEE.

[15] F. Vesperini, D. Droghini, E. Principi, L. Gabrielli, and S. Squartini, "Hierarchic ConvNets framework for rare sound event detection," *Proc. of EUSIPCO*. IEEE, Sept. 3-7 2018.

[16] F. Vesperini, L. Romeo, E. Principi, A. Monteriù, and S. Squartini, "Convolutional recurrent neural networks and acoustic data augmentation for snore detection," *Proc. of WIRN*, Vietri sul Mare, Italy, 13-15 Jun. 2018.

[17] D. Droghini, F. Vesperini, E. Principi, S. Squartini, and F. Piazza, "Few-shot siamese neural networks employing audio features for human-fall detection," *Proc. of The International Conference on Pattern Recognition and Artificial Intelligence*, Union, NJ, USA, Aug. 15-17 2018.

Others:

[1] F. Vesperini, D. Droghini, D. Ferretti, E. Principi, L. Gabrielli, S. Squartini, and F. Piazza, "A hierarchic multi-scaled approach for rare sound event detection," Ancona, Italy, 2017, DCASE Tech. Report. Copyright-free.

[2] F. Vesperini, L. Gabrielli, E. Principi, and S. Squartini, "A capsule neural networks based approach for bird audio detection," Ancona, Italy, 2018, DCASE Tech. Report. Copyright-free.

[3] L. Gabrielli, F. Vesperini, D. Droghini, and S. Squartini, "Rima Glottidis: Experimenting generative raw audio synthesis for a sound installation," *XXII Colloquium of Musical Informatics*, Udine, Italy, 20-23 Nov. 2018.

Bibliography

[1] T. Virtanen, M. D. Plumbley, and D. Ellis, *Computational analysis of sound scenes and events*, Springer, 2018.

[2] A. S. Bregman, *Auditory Scene Analysis: The Perceptual Organization of Sound*, MIT press, 1994.

[3] P. Divenyi, *Speech Separation by Humans and Machines*, Springer Science & Business Media, 2004.

[4] S. Chu, S. Narayanan, and C.-C. J. Kuo, "Environmental sound recognition with time-frequency audio features," *IEEE Trans. Audio, Speech, Language Process.*, pp. 1142–1158, 2009.

[5] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Audio context recognition using audio event histograms," in *Proc. of EUSIPCO*, 2010, pp. 1272–1276.

[6] M. Shah, B. Mears, C. Chakrabarti, and A. Spanias, "Lifelogging: Archival and retrieval of continuously recorded audio using wearable devices," in *Proc. of ESPA*. IEEE, 2012, pp. 99–102.

[7] G. Wichern, J. Xue, H. Thornburg, B. Mechtley, and A. Spanias, "Segmentation, indexing, and retrieval for environmental and natural sounds," *IEEE Trans. Audio, Speech, Language Process.*, pp. 688–707, 2010.

[8] M. Xu, C. Xu, L. Duan, J. S. Jin, and S. Luo, "Audio keywords generation for sports video analysis," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 4, no. 2, pp. 11, 2008.

[9] Y.-T. Peng, C.-Y. Lin, M.-T. Sun, and K.-C. Tsai, "Healthcare audio event classification using hidden Markov models and hierarchical hidden Markov models," in *Proc. of ICME*. IEEE, 2009.

[10] A. Harma, M. F. McKinney, and J. Skowronek, "Automatic surveillance of the acoustic activity in our living environment," in *Proc. of ICME*. IEEE, 2005.

[11] M. Crocco, M. Cristani, A. Trucco, and V. Murino, "Audio surveillance: a systematic review," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, pp. 52, 2016.

[12] E. Principi, D. Droghini, S. Squartini, P. Olivetti, and F. Piazza, "Acoustic cues from the floor: a new approach for fall classification," *Expert Systems with Applications*, pp. 51–61, 2016.

[13] A. Boudoukha, M. Hautekeete, S. Abdellaoui, W. Groux, and D. Garay, "Burnout and victimisation: impact of inmates' aggression towards prison guards," *L'encephale*, vol. 37, no. 4, pp. 284–292, 2011.

[14] J. F. Gemmeke, B. Ons, et al., "Self-taught assistive vocal interfaces: An overview of the ALADIN project," in *Proc. of Interspeech*, Lyon, France, 2013, pp. 2039–2043.

[15] M. Vacher, S. Caffiau, F. Portet, B. Meillon, C. Roux, E. Elias, B. Lecouteux, and P. Chahuara, "Evaluation of a context-aware voice interface for ambient assisted living: Qualitative user study vs. quantitative system evaluation," *ACM Trans. Access. Comput.*, vol. 7, no. 2, pp. 5:1–5:36, May 2015.

[16] E. Principi, S. Squartini, R. Bonfigli, G. Ferroni, and F. Piazza, "An integrated system for voice command recognition and emergency detection based on audio signals," *Expert Systems with Applications*, vol. 42, no. 13, pp. 5668–5683, 2015.

[17] P. C. Loizou, *Speech enhancement: theory and practice*, CRC Press, Boca Raton, FL, 2013.

[18] A. Hussain, M. Chetouani, S. Squartini, A. Bastari, and F. Piazza, *Nonlinear speech enhancement: An overview*, pp. 217–248, Springer, 2007.

[19] R. Rotili, E. Principi, S. Squartini, and B. Schuller, "A real-time speech enhancement framework in noisy and reverberated acoustic scenarios," *Cognitive Computation*, vol. 5, no. 4, pp. 504–516, Aug. 2013.

[20] I. Rodomagoulakis, A. Katsamanis, G. Potamianos, P. Giannoulis, A. Tsiami, and P. Maragos, "Room-localized spoken command recognition in multi-room, multi-microphone environments," *Computer Speech & Language*, vol. 46, pp. 419–443, 2017.

[21] G. Ferroni, R. Bonfigli, E. Principi, S. Squartini, and F. Piazza, "A Deep Neural Network Approach for Voice Activity Detection in Multi-Room Domestic Scenarios," in *Proc. of IJCNN*, Killarney, Ireland, Jul. 12-17 2015, IEEE, pp. 1574–1581.

[22] P. Giannoulis, A. Brutti, M. Matassoni, A. Abad, A. Katsamanis, M. Matos, G. Potamianos, and P. Maragos, "Multi-room speech activity detection using a distributed microphone network in domestic environments," in *Proc. of EUSIPCO*, Nice, France, 31 Aug. - 4 Sep. 2015, IEEE, pp. 1271–1275.

[23] A. Katsamanis, I. Rodomagoulakis, G. Potamianos, P. Maragos, and A. Tsiami, "Robust far-field spoken command recognition for home automation combining adaptation and multichannel processing," in *Proc. of ICASSP*, Florence, Italy, 2014, IEEE, pp. 5547–5551.

[24] T. Grill and J. Schlüter, "Two convolutional neural networks for bird detection in audio signals," in *Proc. of EUSIPCO*. IEEE, Aug 2017, pp. 1764–1768.

[25] S. Krstulović and al., "Audioanalytic – intelligent sound detection," `http://www.audioanalytic.com`, 2016.

[26] T. White and al., "Rainforest connection," `https://rfcx.org/home`, 2016.

[27] E. Wold, T. Blum, D. Keislar, and J. Wheaten, "Content-based classification, search, and retrieval of audio," *IEEE multimedia*, vol. 3, no. 3, pp. 27–36, 1996.

[28] K.-H. Woo, T.-Y. Yang, K.-J. Park, and C. Lee, "Robust voice activity detection algorithm for estimating noise spectrum," *Electronics Letters*, vol. 36, no. 2, pp. 180–181, 2000.

[29] N. Degara, M. E. Davies, A. Pena, and M. D. Plumbley, "Onset event decoding exploiting the rhythmic structure of polyphonic music," *IEEE J. Sel. T. in Signal Proc.*, vol. 5, no. 6, pp. 1228–1239, 2011.

[30] J. J. Carabias-Orti, T. Virtanen, P. Vera-Candeas, N. Ruiz-Reyes, and F. J. Canadas-Quesada, "Musical instrument sound multi-excitation model for non-negative spectrogram factorization," *IEEE J. Sel. T. in Signal Proc.*, vol. 5, no. 6, pp. 1144–1158, 2011.

[31] G. Guo and S. Z. Li, "Content-based audio classification and retrieval by support vector machines," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 209–215, 2003.

[32] I. McLoughlin, H. Zhang, Z. Xie, Y. Song, and W. Xiao, "Robust sound event classification using deep neural networks," *IEEE Trans. Audio, Speech, Language Process.*, vol. 23, no. 3, pp. 540–552, 2015.

[33] A.-r. Mohamed, G. E. Dahl, G. Hinton, et al., "Acoustic modeling using deep belief networks," *IEEE Trans. Audio, Speech, Language Process.*, vol. 20, no. 1, pp. 14–22, 2012.

[34] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *Proc. of MLSP*. IEEE, 2015, pp. 1–6.

[35] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. of ICASSP*. IEEE, 2013, pp. 6645–6649.

[36] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, B. Schuller, and S. Zafeiriou, "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network," in *Proc. of ICASSP*. IEEE, 2016, pp. 5200–5204.

[37] B. Wu, K. Li, F. Ge, Z. Huang, M. Yang, S. M. Siniscalchi, and C.-H. Lee, "An end-to-end deep learning approach to simultaneous speech dereverberation and acoustic modeling for robust speech recognition," *IEEE J. Sel. T. in Signal Proc.*, vol. 11, no. 8, pp. 1289–1300, 2017.

[38] S. Sigtia, A. M. Stark, S. Krstulović, and M. D. Plumbley, "Automatic environmental sound recognition: Performance versus computational cost," *IEEE Trans. Audio, Speech, Language Process.*, pp. 2096–2107, 2016.

[39] S. Hershey, S. Chaudhuri, et al., "CNN architectures for large-scale audio classification," in *Proc. of ICASSP*. IEEE, 2017, pp. 131–135.

[40] R. V. Sharan and T. J. Moir, "An overview of applications and advancements in automatic sound recognition," *Neurocomputing*, vol. 200, pp. 22–34, 2016.

[41] F. Eyben, F. Weninger, S. Squartini, and B. Schuller, "Real-life Voice Activity Detection with LSTM Recurrent Neural Networks and an Application to Hollywood Movies," in *Proc. of ICASSP*, Vancouver, BC, Canada, May 26-31 2013, IEEE, pp. 483–487.

[42] E. Marchi, F. Vesperini, S. Squartini, and B. Schuller, "Deep recurrent neural network-based autoencoders for acoustic novelty detection," *Computational intelligence and neuroscience*, vol. 2017, 2017.

[43] H. B. Sailor, D. M. Agrawal, and H. A. Patil, "Unsupervised filterbank learning using convolutional restricted boltzmann machine for environmental sound classification," *Proc. Interspeech*, pp. 3107–3111, 2017.

[44] M. Espi, M. Fujimoto, K. Kinoshita, and T. Nakatani, "Exploiting spectro-temporal locality in deep learning based acoustic event detection," *EURASIP Journal, on Audio, Speech, and Music Process.*, vol. 2015, no. 1, pp. 26, Sep 2015.

[45] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE Trans. Audio, Speech, Language Process.*, vol. 25, no. 6, pp. 1291–1303, 2017.

[46] T. Virtanen, A. Mesaros, T. Heittola, A. Diment, E. Vincent, E. Benetos, and B. M. Elizalde, *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop*, Tampere University of Technology. Laboratory of Signal Processing, 2017.

[47] H. Lim, J. Park, and Y. Han, "Rare sound event detection using 1D convolutional recurrent neural networks," in *Proc. of DCASE*. IEEE, 2017, pp. 80–84.

[48] E. Cakir and T. Virtanen, "Convolutional recurrent neural networks for rare sound event detection," in *Proc. of DCASE*. IEEE, 2017, pp. 27–31.

[49] H. Phan, M. Krawczyk-Becker, T. Gerkmann, and A. Mertins, "DNN and CNN with weighted and multi-task loss functions for audio event detection," in *Proc. of DCASE*. IEEE, 2017.

[50] F. Vesperini, L. Gabrielli, E. Principi, and S. Squartini, "A capsule neural networks based approach for bird audio detection," Ancona, Italy, 2018, DCASE Tech. Report. Copyright-free.

[51] T. Iqbal, Y. Xu, Q. Kong, and W. Wang, "Capsule routing for sound event detection," in *Proc. of EUSIPCO*. IEEE, Sept. 3-7 2018.

[52] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, 2017, pp. 3856–3866.

[53] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Proc. of ICANN*. Springer, 2011, pp. 44–51.

[54] D. Stowell and D. Clayton, "Acoustic event detection for multiple overlapping similar sources," in *Proc. of WASPAA*. IEEE, 2015, pp. 1–5.

[55] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. of ICASSP*. IEEE, 2017, pp. 776–780.

[56] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *Proc. of EUSIPCO*. IEEE, Aug 2016, pp. 1128–1132.

[57] D. Stowell, Y. Stylianou, M. Wood, H. Pamuła, and H. Glotin, "Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge," *Methods in Ecology and Evolution*, 2018.

[58] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[59] Y. Le Cun and Y. Bengio, "Word-level training of a handwritten word recognizer based on convolutional neural networks," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition, vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5)*. IEEE, 1994, vol. 2, pp. 88–92.

[60] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[61] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[62] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, 2005, vol. 1, pp. 539–546.

[63] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[64] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow, "Adversarial autoencoders," in *Proc. of ICLR*, 2015.

[65] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov 1997.

[66] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[67] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[68] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[69] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. of ICMC*, 2015, pp. 448–456.

[70] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, 1993.

[71] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proc. of. ISMIR*, 2011.

[72] O. Houix, G. Lemaitre, N. Misdariis, P. Susini, and I. Urdapilleta, "A perceptive categorization of sounds produced the interaction of solid objects.," *The Journal of the Acoustical Society of America*, vol. 127, no. 3, pp. 1899–1899, 2010.

[73] G. Lemaitre, O. Houix, N. Misdariis, and P. Susini, "Listener expertise and sound identification influence the categorization of environmental sounds.," *Journal of Experimental Psychology: Applied*, vol. 16, no. 1, pp. 16, 2010.

[74] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. of CVPR*. IEEE, 2009, pp. 248–255.

[75] D. J. Hermes, "Measurement of pitch by subharmonic summation," *The journal of the acoustical society of America*, vol. 83, no. 1, pp. 257–264, 1988.

[76] E. Marchi, G. Ferroni, F. Eyben, L. Gabrielli, S. Squartini, and B. Schuller, "Multi-resolution Linear Prediction Based Features for Audio Onset Detection with Bidirectional LSTM Neural Networks," in *Proc. of ICASSP*. IEEE, 2014, pp. 2183–2187.

[77] H. Hermansky and N. Morgan, "RASTA processing of speech," *IEEE Trans. Speech Audio Process.*, vol. 2, no. 4, pp. 578–589, 1994.

[78] S. Mallat, "Group invariant scattering," *Communications on Pure and Applied Mathematics*, vol. 65, no. 10, pp. 1331–1398, 2012.

[79] W. Li, L. Duan, D. Xu, and I. W. Tsang, "Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 6, pp. 1134–1148, 2014.

[80] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. pp. 321–357, 2002.

[81] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning," *Advances in intelligent computing*, pp. 878–887, 2005.

[82] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, pp. 162, 2016.

[83] C. Janott, W. Pirsig, and C. Heiser, "Acoustic analysis of snoring aids," *Somnologie-Schlafforschung und Schlafmedizin*, vol. 18, no. 2, pp. 87–95, 2014.

[84] S. Chokroverty, *Sleep Disorders Medicine E-Book: Basic Science, Technical Considerations, and Clinical Aspects*, Elsevier Health Sciences, 2009.

[85] T. Young, L. Finn, H. Kim, et al., "Nasal obstruction as a risk factor for sleep-disordered breathing," *Journal of Allergy and Clinical Immunology*, vol. 99, no. 2, pp. S757–S762, 1997.

[86] M. B. Blumen, M. A. Q. Salva, I. Vaugier, K. Leroux, M.-P. d'Ortho, F. Barbot, F. Chabolle, and F. Lofaso, "Is snoring intensity responsible for the sleep partner's poor quality of sleep?," *Sleep and Breathing*, vol. 16, no. 3, pp. 903–907, 2012.

[87] P. J. J. Strollo and R. M. Rogers, "Obstructive sleep apnea," *New England Journal of Medicine*, vol. 334, no. 2, pp. 99–104, 1996.

[88] K. Banno and M. H. Kryger, "Sleep apnea: Clinical investigations in humans," *Sleep Medicine*, vol. 8, pp. pp. 400–426, 2007.

[89] J. C. Lumeng and R. D. Chervin, "Epidemiology of pediatric obstructive sleep apnea," *Proceedings of the American Thoracic Society*, vol. 5, no. 2, pp. pp. 242–252, 2008.

[90] M. R. El Badawey, G. McKee, H. Marshall, N. Heggie, and J. A. Wilson, "Predictive value of sleep nasendoscopy in the management of habitual snorers," *Annals of Otology, Rhinology & Laryngology*, vol. 112, no. 1, pp. 40–44, 2003.

[91] D. Pevernagie, R. M. Aarts, and M. De Meyer, "The acoustics of snoring," *Sleep medicine reviews*, vol. 14, no. 2, pp. 131–144, 2010.

[92] M. Schmitt, C. Janott, V. Pandit, K. Qian, C. Heiser, W. Hemmert, and B. Schuller, "A bag-of-audio-words approach for snore sounds' excitation localisation," in *Proc. of ITG Speech Communication*, Paderborn, Germany, 2016, pp. 230–234.

[93] M. Cavusoglu, M. Kamasak, O. Erogul, T. Ciloglu, Y. Serinagaoglu, and T. Akcam, "An efficient method for snore/nonsnore classification of sleep sounds," *Physiological measurement*, vol. 28, no. 8, pp. 841, 2007.

[94] M. Shokrollahi, S. Saha, P. Hadi, F. Rudzicz, and A. Yadollahi, "Snoring sound classification from respiratory signal," in *Proc. of EMBC*, Orlando, FL, USA, Aug. 16-20 2016, pp. 3215–3218.

[95] C. Wang, J. Peng, L. Song, and X. Zhang, "Automatic snoring sounds detection from sleep sounds via multi-features analysis," *Australasian Physical & Engineering Sciences in Medicine*, pp. 1–9, 2016.

[96] K. Qian, Z. Xu, H. Xu, Y. Wu, and Z. Zhao, "Automatic detection, segmentation and classification of snore related signals from overnight audio recording," *IET Signal Processing*, vol. 9, no. 1, pp. 21–29, 2015.

[97] K. Qian, C. Janott, V. Pandit, Z. Zhang, C. Heiser, W. Hohenhorst, M. Herzog, W. Hemmert, and B. Schuller, "Classification of the excitation location of snore sounds in the upper airway by acoustic multi-feature analysis," *IEEE Trans. on Biomedical Engineering*, vol. PP, no. 99, pp. 1–1, 2016.

[98] B. Schuller, S. Steidl, et al., "The INTERSPEECH 2017 computational paralinguistics challenge: Addressee, cold & snoring," in *Proc. of Interspeech*, Stockholm, Sweden, Aug. 20-24 2017.

[99] S. Amiriparian, M. Gerczuk, S. Ottl, N. Cummins, M. Freitag, S. Pugachevskiy, A. Baird, and B. Schuller, "Snore sound classification using image-based deep spectrum features," in *Proc. of Interspeech*, Stockholm, SWE, 2017.

[100] M. Freitag, S. Amiriparian, N. Cummins, M. Gerczuk, and B. Schuller, "An end-to-evolution hybrid approach for snore sound classification," in *Proc. of Interspeech*, Stockholm, SWE, 2017.

[101] H. Kaya and A. A. Karpov, "Introducing weighted kernel classifiers for handling imbalanced paralinguistic corpora: Snoring, addressee and cold," in *Proc. Interspeech*, Stockholm, SWE, 2017, pp. 3527–3531.

[102] J. Andén and S. Mallat, "Deep scattering spectrum," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4114–4128, 2014.

[103] S. B. Davis and P. Mermelstein, "Comparison of parametric representation for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[104] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech Communication*, vol. 52, no. 1, pp. 12 – 40, 2010.

[105] L. Sifre and S. Mallat, "Rotation, scaling and deformation invariant scattering for texture discrimination," in *Proc. of CVPR*, 2013, pp. 1233–1240.

[106] J. Andén and S. Mallat, "Multiscale scattering for audio classification.," in *ISMIR*, 2011, pp. 657–662.

[107] L. Sifre, M. Kapoko, E. Oyallon, and V. Lostanlen, "Scatnet: a matlab toolbox for scattering networks," 2013.

[108] V. Nair and G. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. of ICML*, Haifa, Israel, Jun. 21-24 2010, pp. 807–814.

[109] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proc. of ICLR*, 2014.

[110] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016.

[111] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[112] C. Bishop, *Pattern Recognition and Machine Learning*, Springer Science+Business Media, LLC, New York, 2006.

[113] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27, 2011.

[114] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[115] G. King and L. Zeng, "Logistic regression in rare events data," *Political analysis*, vol. 9, no. 2, pp. 137–163, 2001.

[116] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[117] D. I. Hanson, R. S. James, and C. NeSmith, *Tire/pavement noise study*, the Center, 2004.

[118] U. Sandberg, "Tyre/road noise: myths and realities," in *International Congress and Exhibition on Noise Control Engineering, The Hague, Netherlands*, 2001.

[119] U. Sandberg, "The multi-coincidence peak around 1000 hz in tyre/road noise spectra," in *Proceedings of EURONOISE*, 2003, vol. 2003.

[120] E. Freitas, P. Pereira, L. de Picado-Santos, and A. Santos, "Traffic noise changes due to water on porous and dense asphalt surfaces," *Road Materials and Pavement Design*, vol. 10, no. 3, pp. 587–607, 2009.

[121] D. Doğan, "Road-types classification using audio signal processing and svm method," in *Proc. of SIU*, May 2017, pp. 1–4.

[122] J. Alonso, J. Lopez, I. Pavón, M. Recuero, C. Asensio, G. Arcas, and A. Bravo, "On-board wet road surface identification using tyre/road noise and support vector machines," *Applied Acoustics*, vol. 76, pp. 407–415, 2014.

[123] I. Abdić, L. Fridman, D. E. Brown, W. Angell, B. Reimer, E. Marchi, and B. Schuller, "Detecting road surface wetness from audio: A deep learning approach," in *Proc. of ICPR*. IEEE, 2016, pp. 3458–3463.

[124] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[125] Y. LeCun, Y. Bengio, et al., "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, pp. 1995, 1995.

[126] F. Eyben, S. Böck, B. Schuller, and A. Graves, "Universal onset detection with bidirectional long-short term memory neural networks," in *Proc. of ISMIR*, Utrecht, Netherlands, 2010, pp. 589–594.

[127] F. Eyben, F. Weninger, F. Gross, and B. Schuller, "Recent developments in openSMILE, the munich open-source multimedia feature extractor," in *Proc. of ACM Multimedia 2013*, Barcelona, Spain, 2013, pp. 835–838, ACM.

[128] Y. Ephraim and D. Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator," *IEEE Trans. Audio, Speech, Language Process.*, vol. 32, no. 6, pp. 1109–1121, 1984.

[129] P. K. Ghosh, A. Tsiartas, and S. Narayanan, "Robust voice activity detection using long-term signal variability," *IEEE Trans. Audio, Speech, Language Process.*, vol. 19, no. 3, pp. 600–613, 2011.

[130] P. Vecchiotti, F. Vesperini, E. Principi, S. Squartini, and F. Piazza, "Convolutional neural networks with 3-D kernels for voice activity detection in a multiroom environment," in *Multidisciplinary Approaches to Neural Computing*, pp. 161–170. Springer, 2018.

[131] F. Vesperini, P. Vecchiotti, E. Principi, S. Squartini, and F. Piazza, "Deep neural networks for multi-room voice activity detection: Advancements and comparative evaluation," in *Proc. of IJCNN*, Vancouver, Canada, 24-29 Jul. 2016, pp. 3391–3398.

[132] M. Tan and K. McDonald, "Bird Sounds:thousands of bird sounds visualized using machine learning.," 2018.

[133] D. Stowell, M. Wood, Y. Stylianou, and H. Glotin, "Bird detection in audio: A survey and a challenge," pp. 1–6, Sept 2016.

[134] I. McLoughlin and Y. Song, "Low frequency ultrasonic voice activity detection using convolutional neural networks," in *Proc. of Interspeech*, Dresden, Germany, Sep. 6-10 2015.

[135] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[136] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proc. of DCASE*. IEEE, 2017.

[137] M. Valenti, S. Squartini, A. Diment, G. Parascandolo, and T. Virtanen, "A convolutional neural network approach for acoustic scene classification," in *Proc. of IJCNN*. IEEE, May 2017, pp. 1547–1554.

[138] R. B. Payne, "Handbook of the birds of the world," *The Wilson Journal of Ornithology*, vol. 122, no. 3, pp. 627–629, 2010.

[139] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[140] M. D. Zeiler, "AdaDelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[141] F. Chollet et al., "Keras," `https://github.com/keras-team/keras`, 2015.

[142] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2016.

[143] G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," in *Proc. of ICLR*, Vancouver, BC, 2018.

[144] F. Vesperini, A. Galli, L. Gabrielli, E. Principi, and S. Squartini, "Snore sounds excitation localization by using scattering transform and deep neural networks," in *Proc. of IJCNN*, Rio the Janeiro, Brazil, July 8-13 2018, IEEE.

[145] M. Valenti, D. Tonelli, F. Vesperini, E. Principi, and S. Squartini, "A neural network approach for sound event detection in real life audio," in *Proc. of EUSIPCO*. IEEE, 2017, pp. 2754–2758.

[146] R. Scheibler, E. Bezzam, and I. Dokmanić, "Pyroomacoustics: A Python package for audio room simulations and array processing algorithms," in *Proc. ICASSP*, Calgary, Canada, Apr. 15-20 2018, IEEE.

[147] F. Vesperini, D. Droghini, D. Ferretti, E. Principi, L. Gabrielli, S. Squartini, and F. Piazza, "A hierarchic multi-scaled approach for rare sound event detection," Ancona, Italy, 2017, DCASE Tech. Report. Copyright-free.

[148] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[149] X.-L. Zhang and D. Wang, "Boosting contextual information for deep neural network based voice activity detection," *IEEE Trans. Audio, Speech, Language Process.*, vol. 24, no. 2, pp. 252–264, Feb 2016.

[150] P. Giannoulis, A. Tsiami, I. Rodomagoulakis, A. Katsamanis, G. Potamianos, and P. Maragos, "The Athena-RC system for speech activity detection and speaker localization in the DIRHA smart home," in *Proc. of HSCMA, 2014*, Florence, Italy, May 12-14 2014, pp. 167–171.

[151] A. Abad, M. Matos, H. Meinedo, R. F. Astudillo, and I. Trancoso, "The L2F system for the EVALITA-2014 speech activity detection challenge in domestic environments," in *Proc. of EVALITA*, 2014, pp. 147–152.

[152] J. A. Morales-Cordovilla, M. Hagmuller, H. Pessentheiner, and G. Kubin, "Distant speech recognition in reverberant noisy conditions employing a microphone array," in *Proc. of EUSIPCO*, Lisbona, Portugal, Sep 1-5 2014, pp. 2380–2384.

[153] S. Thomas, S. Ganapathy, G. Saon, and H. Soltau, "Analyzing convolutional neural networks for speech activity detection in mismatched acoustic conditions," in *Proc. of ICASSP*, Florence, Italy, May 4-9 2014, pp. 2519–2523.

[154] R. Price, K.-I. Iso, and K. Shinoda, "Wise teachers train better DNN acoustic models," *Eurasip Journal on Audio, Speech, and Music Processing*, vol. 2016, no. 1, 2016.

[155] L. Cristoforetti, M. Ravanelli, M. Omologo, A. Sosi, A. Abad, M. Hagmüller, and P. Maragos, "The DIRHA simulated corpus," in *Proc. of LREC*, Reykjavik, Iceland, May 26-31 2014, vol. 5.

[156] N. Lopes and B. Ribeiro, "Towards adaptive learning with improved convergence of deep belief networks on graphics processing units," *Pattern Recogn.*, vol. 47, no. 1, pp. 114–127, Jan. 2014.

[157] S. Adavanne, G. Parascandolo, T. Heittola, and T. Virtanen, "Sound event detection in multichannel audio using spatial and harmonic features," in *Proc. of DCASE*. IEEE, 2016.

[158] R. Rojas, *Neural networks: a systematic introduction*, chapter 7, Springer Science & Business Media, 2013.

[159] J. Sohn, N. S. Kim, and W. Sung, "A statistical model-based voice activity detection," *IEEE signal processing letters*, pp. 1–3, 1999.

[160] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[161] S. Adavanne and T. Virtanen, "A report on sound event detection with different binaural features," *arXiv preprint arXiv:1710.02997*, 2017.

[162] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. of AISTATS*, 2010, pp. 249–256.

[163] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proc. of SciPy*, 2015, pp. 18–25.

[164] F. Vesperini, D. Droghini, E. Principi, L. Gabrielli, and S. Squartini, "Hierarchic ConvNets framework for rare sound event detection," in *Proc. of EUSIPCO*. IEEE, Sept. 3-7 2018.

[165] F. Deng, S. Pu, X. Chen, Y. Shi, T. Yuan, and S. Pu, "Hyperspectral image classification with capsule network using limited training samples," *Sensors*, vol. 18, no. 9, 2018.

[166] Y. Shen and M. Gao, "Dynamic routing on deep neural network for thoracic disease classification and sensitive area localization," in *Machine Learning in Medical Imaging*, Y. Shi, H.-I. Suk, and M. Liu, Eds. 2018, pp. 389–397, Springer International Publishing.

[167] M. A. Jalal, R. Chen, R. K. Moore, and L. Mihaylova, "American sign language posture understanding with deep neural networks," in *Proc. of FUSION*. IEEE, 2018, pp. 573–579.

[168] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio.," in *SSW*, 2016, p. 125.

[169] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "SampleRNN: An unconditional end-to-end neural audio generation model," in *Proc. of ICLR*, 2016.

[170] C. Donahue, J. McAuley, and M. Puckette, "Synthesizing audio with GANs," in *Proc. of ICLR*, Vancouver, Canada, 2018.

[171] Z. Jin, A. Finkelstein, G. J. Mysore, and J. Lu, "FFTnet: A real-time speaker-dependent neural vocoder," in *Proc. of ICASSP*. IEEE, 2018, pp. 2251–2255.

[172] L. Cabello Piqueras, "Autoregressive model based on a deep convolutional neural network for audio generation," 2017, MSc. Thesis.