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

Ambient Intelligence: Computational Audio Processing For Human Fall Detection

Ph.D. Dissertation of:
Diego Droghini

Advisor:
Prof. Francesco Piazza

Coadvisor:
Prof. Stefano Squartini

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

Ambient Intelligence: Computational Audio Processing For Human Fall Detection

Ph.D. Dissertation of:
Diego Droghini

Advisor:
Prof. Francesco Piazza

Coadvisor:
Prof. Stefano Squartini

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

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Contents

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The origins of Ambient Intelligence are rooted in the birth of domotics and Home Automation. In fact, the concept of home automation design was already being started in the 80s for residential buildings. Information revolution, microprocessors, and telecommunication networks are the main elements that have allowed the development of domotics. At the same time, increasingly sophisticated automation processes in the manufacturing field are at the base of building automation and domotics applications, transferring the control and automation systems present in the factories, with appropriate measures, to the building and its plants. During the decade '70 '80, the foundations for the transformation of household appliances were laid. In this phase, we witness the first transition from the electric house to the electronic house; subsequently, the progress of the sector will lead to the current concepts of integrated home system, and therefore bringing with it the concept of services that put man at the center, and therefore of *ambient intelligence*. The vision of the Ambient Intelligence (AmI) [1] has its foundations in the article by M. Weiser [2]. He states that "the deepest technologies are those that disappear" and states that computer technology at the time of its maturity should be invisible. Today the efforts in this direction are aimed at creating a "non-deterministic and open" cyberspace within which autonomous and intelligent entities will interact in order to put man at the center of a design that will see the realization of the fully integrated home of the future. This space will be able to self-organize and self-adapt to the user and anticipating his needs. The systems that conform to this vision will offer technological solutions that will be integrated into the environment, context-aware, tailored to the user need, able to adapt actions in new scenarios and able to anticipate the needs and wishes of users, all with minimal user intervention. The objects and the environment will interact with each other in order to support users in carrying out their daily activities in a natural way, using the information and the intelligence that is hidden in the technological infrastructure that connects the devices (the technological complexity will become invisible for the user). In an AmI system, many heterogeneous devices work in cooperation to support users in everyday activities. As a result, the

intelligence has been introduced in the domestic environment to provide more comfortable spaces for the inhabitants and allow the automatic implementation of different functions, such as ensuring greater independence and autonomy of the inhabitants acting in the areas of security/safety and issues of Ambient Assisted Living (AAL). A particular application in the field of AAL is precisely the human fall detection. The decreasing birth rate [3] and the simultaneous increase of the life expectancy at birth [4] in the majority of industrialized countries have been generating new challenges in the assistance of the elderly. The scientific community, companies, and governments are trying to face them by investing in the development of efficient healthcare systems and solutions. The direction taken goes towards the development of smart home capable of taking care of the inhabitants by supporting and monitoring them in their daily actions [5, 6]. Since falls are one of the leading cause of death for the elderly [7], several efforts have been devoted to the development of algorithms for automatically detecting this kind of events. This work is aimed at the presentation of different computational audio processing systems for fall detection. The approach described in each chapter follows a data availability perspective, that means the systems that presented will approach the problem by considering increasingly complex and realistic scenarios starting from different knowledge conditions. The main contribution is to demonstrate that the audio human fall detection is a reliable solution and not only the mainstream systems based on vision or wearable sensors can be used for this kind of problem. In particular, a particular acoustic sensor specially designed for the fall detection task is proposed and evaluated. Moreover, the dataset used to assess all the proposed methods has been created by the same authors and made available by the scientific community. The authors aim to provide a complete dataset to the scientific community, that other researchers can use in order to compare the performances of their proposed systems in the audio field concerning the detection of human falls.

The outline of the dissertation is the following. Section 1.1introduce the human fall classification task with an updated state-of-the-art mainly focused on the audio based approaches. Chapter 2 gives an overview of the theoretical background of the data-driven techniques used during the development of the presented systems. The Chapter 3 described the dataset made by the authors, used for the assessment of the systems. Chapter 4 presents the supervised approaches aimed to evaluate both the created dataset and the innovative proposed sensor. The unsupervised systems are described in Chapter 5 where no human fall data has been used to train the algorithms. In Chapter 6 approaches that operate in more realistic conditions are described, where additional information is provided by the user or available for the target class

1.1 Fall Detection Systems

The continuous and unprecedented growth rate of the elderly world population is one of the primary aspects of concern for society and governments. Nowadays about 8.5% of people in the world are more than 65 years old [8, 4]. Although the average life of the world population is getting longer, older adults may not necessarily live a healthier life. It is enough to say that 37.5 million falls require medical interventions and more than 600 thousand are the cause of death every year worldwide. In particular, the population segment most affected by this problem is composed of elderly over 65 years that, with the growing mobility of the population, are more frequently left alone in their homes without aid in the case of need. Moreover, since falls are the leading cause of death and hospitalizations for older adults, this phenomenon leads to a substantial increase in the cost of healthcare [9, 7]. It is not surprising, thus, that the research community is encouraged, even by governments, to find reliable and performing solutions to minimize the damage caused by the human falls problem. The above is also confirmed by the presence in the literature of several reviews dedicated to this specific topic [7, 10, 11, 12, 13, 14]. In fact, in the past few years, a variety of systems have been presented. One way to divide the methodologies for approaching the falls detection problem is based on the placement of the sensing devices [7]. The main categories are wearable, vision and environmental, with each category presenting advantages and disadvantages. Wearable systems do not suffer from ambient condition, but people may forget to wear them, and they are not operational during the charging time; thus, some people may consider them annoying. Furthermore, a device must be installed on each person to be monitored. An environmental sensor may be used to avoid this kind of problems, but with other limitations. Vision systems, although they are actually environmental sensors, deserve a dedicated category because of many systems proposed in the literature based on this type of sensors [7]. This category includes several types of sensors like, e.g., cameras for which the major limitations are field-of-view constraints, lighting condition, the positioning of multiple cameras and lack of privacy. The ambient category includes several types of sensors. For example, radar doppler based systems used in [15] raise fewer privacy concerns, but they suffer from reflection and blind spots. In particular, for a data-driven system, another aspect that should not be underestimated is the need for a re-training when changing the environment to be monitored or even just some of its components such as the arrangement of furniture as happens in [16]. All this implies that there is no optimal choice, which is instead, a compromise that depends on the type of environment that is monitored as well as on personal sensitivity of the subjects under monitoring. Going into more detail, another significant distinction between falls detection

systems can be made based on the type and amount of data used for the algorithm development [10]. In fact, the problem can be approached either as supervised or unsupervised based on the availability of data in the hands of the researchers as well as their goals. Most state-of-the-art works tackle the problem under fully supervised conditions assuming they have enough data for falls. Almost all of these falls are simulated with professional mannequins [17, 18] or by people with adequate protections [19, 20] that however may not correctly emulate an actual fall. Although this approach leads to more accurate results, there is no guarantee that it will generalize well in real situations. Other researchers opt for approaches based on outlier/anomaly detection [21, 22, 23] because of the plentiful availability of data that can represent normal activity. However, it is challenging to define what "normal activities" are for such approaches, and the risk is to raise several false alarms. Perhaps the situation that most closely approximates reality is a hybrid between the previous ones, in which a large amount of data representing the normality are easily available, with just a few samples of real human fall and eventually some related synthetic or simulated data. In these situations, supervised approaches that suffer from strong data imbalance have to apply subsampling [24] or weighting [10] techniques to mitigate this effect. Thus, the need to find an effective way to exploit the few available falls data is evident.

1.2 State-Of-The-Art

As aforementioned, fall detection approaches can be divided based on their sensing technology, in particular if they employ wearable or ambient sensors. Regarding the first ones, the most common choice is to employ accelerometers. The algorithms proposed in [25] and [26] detect a fall by verifying if the acceleration signals exceed a certain threshold. In contrast, in [27] the authors implemented several machine learning techniques and studied their classification performance. For the experiments, a fall events dataset has been developed using six sensor units with three-axis accelerometers and worn by 14 persons who simulated falls from different angles. The best performing classifier resulted the k-Nearest Neighbour classifier. [28] employed radio tags worn on the user's chest, waist, and ankles, and an optional three-axial accelerometer worn on the chest. The algorithms performs a basic activity recognition, distinguishing from walking, standing, sitting, sitting on the ground, lying down, the process of sitting or lying down, the process of standing up, and falling. The actual fall is detected combining the results of two classifiers, an SVM and a decision tree, and hand-crafted rules. The authors performed the experiments on a laboratory scenario and reported accuracies of 100% combining radio tags and the accelerometer.

Differently from wearable sensors, the physical quantities captured by ambient sensors are more heterogeneous. Generally, fall detectors are based on vibration, video or acoustic sensors, sometimes in combination with presence detectors. In [29] the fall detector is based on a floor vibration sensor and the algorithm detects a fall when the vibration pattern matches the one of a human fall. The authors do not give further details on the algorithm and report 100% sensitivity and specificity on tests conducted on a dummy falls dataset. Yazar and colleagues [30] employ both passive infrared (PIR) sensors and floor vibration sensors. PIRs are employed to reduce false alarms, i.e., by detecting if a person is present in the region of interest. Single-tree complex wavelet transform features are extracted from the vibration signal and classified as fall or non-fall. In their dataset, the non-fall classes are represented by human (walking or running and sitting) or non-human activities (door slamming and a book falling). Three different classifiers have been compared: Euclidean distance, Mahalanobis distance and SVM, with the latter resulting in the most performing one, since it is able to classify human falls without errors regardless the employment of PIR sensors.

Regarding approaches based on audio signals, a common solution is to install several microphones in the building, usually on the ceiling or near the walls. Indeed, also single-microphone approaches exist but they are much less robust to environmental noise, thus resulting in poor performance. For example, in [31], the authors employ a single far field microphone and they model audio segments by means of perceptual linear predictive (PLP) coefficients and GMM supervectors. An SVM with a kernel based on the Kullback-Leibler divergence, then, classifies the segment as being a fall or noise. For this purpose, nine classes of noise have been considered. In the experiments, the algorithm achieves an F_1-Measure of 67% in the classification task, and an accuracy equal to 64% in the detection task.

The difficulty in using a single microphone drove the scientific community to employ multi-channel algorithms. In [21] the authors present an unsupervised algorithm based on two microphones. The algorithm comprises a source separation and localization block to reduce the impact of background noise. Then, a one class Support Vector Machine is trained on MFCCs of non-fall events only. The SVM is then applied to distinguish normal sound events (i.e., sounds originating from normal activities) from abnormal ones (i.e., falls sounds). The authors validated the algorithm using simulated falls of persons only in presence of a television that produced the interfering sound. The results in terms of Area Under Curve are 0.9928 without interference and 0.9738 with 75% interference. The work by Li and colleagues [19] employs a circular microphone array to firstly determine the position of the sound source, and then to enhance the signal by applying a beamformer. The height of the sound source is

used as first filter to discriminate falls from non falls. If the sound originates from a source positioned on the ground, MFCC features are extracted and a k-Nearest Neighbour classifier is employed to detect persons' falls. The algorithm has been tested on a dataset composed of 120 simulated fall sounds and 120 non-fall sounds recorded in different acoustic conditions. In presence of background noise and TV interference, the resulting AUC was equal to 0.989 (accuracy 95%) on clean conditions and 0.932 at 10 dB SNR (accuracy 89%).

An approach to improve the performance of fall detection systems is to combine the information coming from different sensors. The approach proposed by Zigel *et al.* [18] is based on a combination of sound and vibration sensors attached to the floor with a adhesive tape. The algorithm employs energy features extracted from the vibration signal to detect the fall event. Then, the event is classified as fall or non-fall with naive Bayes classifier employing features from both the vibration and sound signals. The experiments were conducted on a dataset containing falls of the "Rescue Randy" human mimicking doll and four objects, and the resulting sensitivity and specificity were respectively 97.5% and 98.6%. [32] fuse the information coming from a Doppler sensor and motion sensors and classify falls with an SVM. The authors report an AUC equal to 0.98 with Doppler sensor only, and a further reduction of false alarms by 63% employing motion sensors information. Motion, sound and video signals are employed in [33]. Signals are captured both from environment sensors and from body sensors. A fall is detected by analysing sounds and motion information, while visual and motion behaviour indicates the severity of the fall. The work by Toreyin and colleagues [34] combines PIRs, microphones and vibration sensors. Signals are processed to extract features in the wavelet domain and and HMM classifier is then employed to detect falls. The authors showed the using PIR signals 100% accuracy can be obtained.

1.3 Case studies

Generally speaking, the "analytical methods" distinguish between fall and non-fall events by applying a threshold directly on the acquired signals or on the features sequences extracted from them [35]. These methods are generally built exploiting some a-priori knowledge to operate in a specific scenario and needs manual tuning of the hyperparameters of the algorithm. For these reasons, the "analytical methods" can hardly perform when the operating conditions and the subjects are variable. In "machine learning" methods, the algorithm learn from the data how to discriminate falls from non-falls [35]. Between them can be distinguished "supervised" and "unsupervised" approaches. The fist require a labelled dataset for training the classifier, while the latter build a normality model considering only the non-fall events. Regardless of the used

approach, machine learning tasks require that the inputs are mathematically and computationally convenient to process, so researchers have traditionally relied on a two-stage strategy: some features are extracted from the raw signals of dataset and are then used as input for the successive tasks. The choice and design of the appropriate features requires considerable expertise about the problem and constitutes a significant engineering effort.

In this work, different application of computational audio processing based on machine learning techniques for ambient intelligence are analyzed. Particular attention was given to the knowledge condition each proposed approach. In fact, the presented works start from a total knowledge of the data describing first the supervised methods dedicated to the falls detection. Because of the difficulty in recovering examples of human fall for algorithm training, this is primarily just a case study. Subsequently, methods that operate in the opposite condition are described, that is, without the a priori knowledge of signals related to the human fall. This would be the ideal condition, but that does not present very high reliability in terms of false alarms rate. Finally, the problem is dealt with from a more realistic point of view, in which only a small portion of data related to the human fall is available, while a vast knowledge of what is not human fall can be accessed.

Chapter 2

Background

In recent years, the IoT revolution has led to the creation of enormous amounts of data. The use of intelligent devices that can interface with cloud computing systems or perform complex calculations directly on board, in homes and cities, has allowed the affirmation of data-driven algorithms compared to other methodologies used so far. In fact, these approaches try to emulate the functioning of the human mind, enabling computers to perform tasks that are unthinkable until now. In this chapter are resumed the data-driven algorithms used for developing the proposed methodologies for fall classification systems.

2.1 Support Vector Machines

Support Vector Machines (SVM) [36] are one of the most popular classification algorithms and are well known for their strong theoretical foundations, generalization performance and ability to handle high dimensional data. This section presents an overview of support vector machine, starting with linear SVMs, followed by their extension to the nonlinear case and finally the One-Class SVM for novelty detection.

Linear Support Vector Machines In the binary classification setting, let $((x_1, y_1) \dots (x_n, y_n))$ be the training dataset where $x_i \in \Re^n$ are the n-dimensional feature vectors representing the instances (i.e. observations) and $y_i \in \{-1, +1\}$ be the labels of the instances. Support vector learning is the problem of finding a separating hyperplane that separates the positive examples (labeled +1) from the negative examples (labeled -1) with the largest margin:

$$f(\vec{w}) = \text{sign}(\vec{w}^T \cdot \vec{x} + b), \tag{2.1}$$

where a value of -1 indicates one class, and a value of $+1$ the other class. In the simpler linearly separable problem, the margin of the hyperplane is defined as the shortest distance between the positive and negative instances that are closest to the hyperplane. The intuition behind searching for the hyperplane

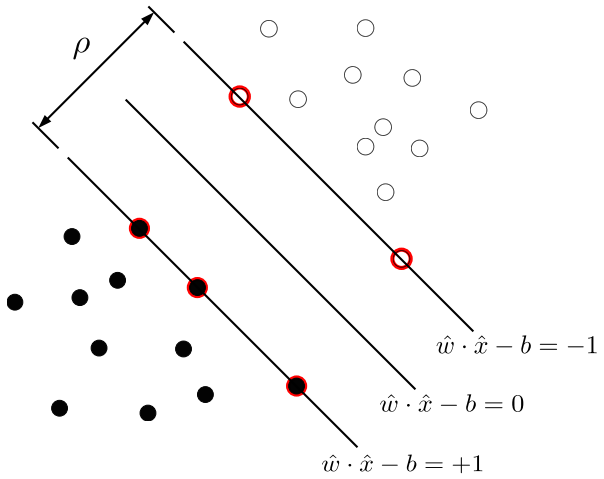


Figure 2.1: A hyperplane separating two classes with the maximum margin. The red highlighted points are the support vectors.

with a large margin is that a hyperplane with the largest margin should be more resistant to noise than a hyperplane with a smaller margin.

Formally, suppose that all the data satisfy the constraints

$$\vec{w} \cdot \vec{x}_i + b \geq +1 \text{ for } y_i = +1, \tag{2.2}$$

$$\vec{w} \cdot \vec{x}_i + b \leq +1 \text{ for } y_i = -1, \tag{2.3}$$

where $\vec{w}$ is the normal to the hyperplane, $\frac{|b|}{\|\vec{w}\|}$ is the perpendicular distance from the hyperplane to the origin, and $\|\vec{w}\|$ is the Euclidean norm of $\vec{w}$. These two constraints can be expressed in compact form as:

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1. \tag{2.4}$$

The *canonical hyperplane* is the hyperplane that separates the data and has maximal margin. The margin ρ can be computed as the distance between the two canonical hyperplanes:

$$\rho = \frac{1-b}{\|\vec{w}\|} - \frac{-1-b}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|} \tag{2.5}$$

Thus, we need to solve an optimisation problem, finding the hyperplane that maximises the margin and ensures the classes are separable

$$\min_{\vec{w}_i, b} \frac{1}{2}\|\vec{w}\|^2 \text{ subject to } y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1. \tag{2.6}$$

The problem can be expressed in the Lagrangian formulation:

$$\mathcal{L}(\vec{w}, b, \lambda) = \frac{1}{2}\|\vec{w}\|^2 + \sum_{i=1}^{m} \lambda_i (1 - y_i(\vec{w} \cdot \vec{x}_i + b)) \tag{2.7}$$

with Lagrange multipliers $\lambda_i \geq 0$ for each constraint in 2.6. The objective is then to minimize 2.7 with respect to $\vec{w}$ and b and simultaneously require that the derivatives of $\mathcal{L}(\vec{w}, b, \lambda)$ with respect to all the λ vanish. The advantage is twofold: the training vectors only appear as a scalar product among the vectors, and the constraints are easier to manage.

With the formulation presented above, the SVM fails in some situation. In fact, there is no solution if samples can not be separated by a hyperplane. Moreover, although data are linearly separable the SVM may overfit to some outlier compromising system performance. For dealing with this type of problem, has been developed the soft margin SVM [36] which allows data points to lie within the margins. Introducing *slack variables* ξ_i into the constraints and penalize them in objective, the new problem becomes

$$\min_{\vec{w}_i, b, \vec{\xi}} \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{m} \xi_i \tag{2.8}$$

$$\text{subject to } y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \text{ for } i = 1 \cdots m.$$

The cost coefficient C > 0 is a hyper-parameter that specifies the misclassification penalty and is tuned by the user based on the classification task and dataset characteristics.

Non-Linear Support Vector Machines A way to solve the problem when data are not linearly separable, is to map the data on to a higher dimensional space and then to use a linear classifier in the higher dimensional space. This methods is referred to as "the kernel trick " that exploit the fact that the training data appears as a dot product between vectors in the Lagrangian formulation to from non-linear decision boundaries. Suppose to use a transformation $\Phi : \vec{x} \rightarrow \phi(\vec{x})$ to map every data sample into higher dimensional space, the dot product becomes $\phi(\vec{x}_i)^T \phi(\vec{x}_j)$. By the use of a kernel function

$$K(\vec{x}_i, \vec{x}_j) = \langle \phi(\vec{x}_i), \phi(\vec{x}_j) \rangle, \tag{2.9}$$

it is possible to compute the separating hyperplane without explicitly carrying out the mapping into feature space. The classifier become:

$$f(\vec{x}) = \text{sign}(\sum_i \lambda_i y_i K(\vec{x}_i, \vec{x}_j) + b) \tag{2.10}$$

The most popular kernel functions are:

- Linear Kernel:
$$K(\vec{x}_i, \vec{x}_j) = \langle \vec{x}_i, \vec{x}_j \rangle \tag{2.11}$$
- Polynomial Kernel:
$$K(\vec{x}_i, \vec{x}_j) = (\langle \vec{x}_i, \vec{x}_j \rangle)^d \tag{2.12}$$
- Sigmoid Kernel:
$$K(\vec{x}_i, \vec{x}_j) = tanh(\gamma \langle \vec{x}_i, \vec{x}_j \rangle - \theta) \tag{2.13}$$
- RBF Kernel:
$$K(\vec{x}_i, \vec{x}_j) = \exp(-\frac{\|\vec{x}_i - \vec{x}_j\|}{2\sigma^2}) \tag{2.14}$$

Up to now the SVM algorithm for binary classification has been described. This algorithm can be extended to the multi-class case using the "one vs all" technique [37].

2.1.1 One-Class Support Vector Machines

One-Class SVM (OCSVM) proposed by Schölkopf et al. [38] is the extension of the support vector machine to the case of unlabeled data that makes them useful for novelty detection problems. In the OCSVM, a new parameter ν that controls the trade-off between maximizing the distance of the hyperplane from the origin and the number of data points contained by the hyperplane has been introduced. To separate the data from the origin, the following quadratic program has to be solved:

$$\min_{\vec{w}_i, \vec{\xi}, \rho} \frac{1}{2}\|\vec{w}\|^2 + \frac{1}{\nu l}\sum_{i=1}^{m} \xi_i - \rho \tag{2.15}$$

$$\text{subject to} \quad (\vec{w} \cdot \phi(\vec{x}_i)) \geq \rho - \xi_i \text{ and } \xi_i \geq 0 \text{ for } i = 1 \cdots m.$$

In fact, One-Class SVM consists in a discriminant function that takes the value $+1$ in a small region that captures the majority of the data points of a set and -1 outside that region [39]. The discriminant function has the following expression:

$$f(\mathbf{x}) = \text{sgn}\left(\sum_i \alpha_i \cdot k(\mathbf{x}_i, \mathbf{x}) - \rho\right), \tag{2.16}$$

where $\vec{x}_i$ denotes the i-th support vector. The position of the hyperplane, thus, defines the region that represents normal data points. For each point $\mathbf{x}$ that lies outside this region, the function $f(\vec{x})$ takes the value -1, whereas for point

inside the region, it takes the value +1. The terms λ_i can be found by solving the solution to the dual problem:

The terms λ_i can be found by solving the solution to the dual problem:

$$\min_{\lambda} \frac{1}{2} \sum_{ij} K(\vec{x_i}, \vec{x_j}) \tag{2.17}$$

$$\text{subject to} \quad 0 \leq \lambda_i \leq \frac{1}{\nu l} \quad \text{and} \quad \sum_i \lambda_i = 1,$$

where λ_i is a Lagrange multiplier and l is the number of points in the training dataset. The term $\nu \in (0, 1]$ is an hyperparameter of the algorithm that is determined on a validation set.

The offset ρ can be obtained from the Karush-Kuhn-Tucker (KKT) condition with the expression [40]:

$$\rho = \sum_j \lambda_i k(\vec{x}_j, \vec{x}_i), \tag{2.18}$$

which is satisfied for any λ_i that is not at the upper or lower bound.

2.2 Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities. Generally, GMMs are used as a parametric model of the probability distribution of some features and then are used to extract higher level features such Gaussian Mean Supervectors (GMS). GMSs are higher level features composed of the means of a Gaussian mixture model (GMM) adapted with maximum a posteriori (MAP) algorithm [41, 42]. The GMM models a Universal Background Model (UBM) and is trained on a large set of audio data by using Expectation Maximization (EM) algorithm [43]. Then, a GMS is calculated by adapting the GMM with the MAP algorithm [44] and concatenating the adapted GMM mean values. More in details, consider a sequence of L feature vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2 \ldots, \mathbf{x}_L\}$, where each $\mathbf{x}_l$ has size $D \times 1$. The GMM representing an UBM is given by

$$p(\mathbf{x}_l|\lambda) = \sum_{j=1}^{J} w_j p(\mathbf{x}_l|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j), \tag{2.19}$$

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 $D \times 1$ mean vector $\boldsymbol{\mu}_j$ and $D \times D$ diagonal covariance matrix $\boldsymbol{\Sigma}_j$.

The GMS $\mathbf{M}$ of the sequence $\mathbf{X}$ is obtained by adapting the means of the

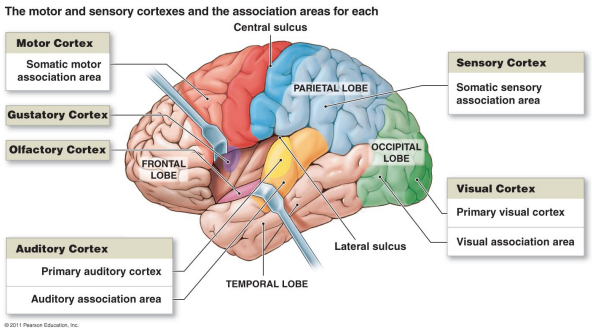


Figure 2.2: The human brain.

UBM model with maximum a posteriori (MAP) algorithm and then concatenating the mean vectors:

$$\mathbf{M} = [\boldsymbol{\mu}_1^T, \boldsymbol{\mu}_2^T, \cdots, \boldsymbol{\mu}_J^T]^T, \tag{2.20}$$

where T denotes the transpose operator. Regardless the number of vectors in the sequence $\mathbf{X}$, $\mathbf{M}$ is a $DJ \times 1$ vector.

The number of Gaussians J are generally determined on a validation set.

2.3 The Artificial Deep 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 2.2. 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.

2.3.1 The Human Nervous System

A *biological Neural Networks* is a big set of specialized cells (*neurons*) connected among them, which memorize and process information, thus controlling the body activities they belong to.

The *neuron* model is composed of:

- *Soma*, which is the calculation unit
- *Axon*, that acts as a transmission line output

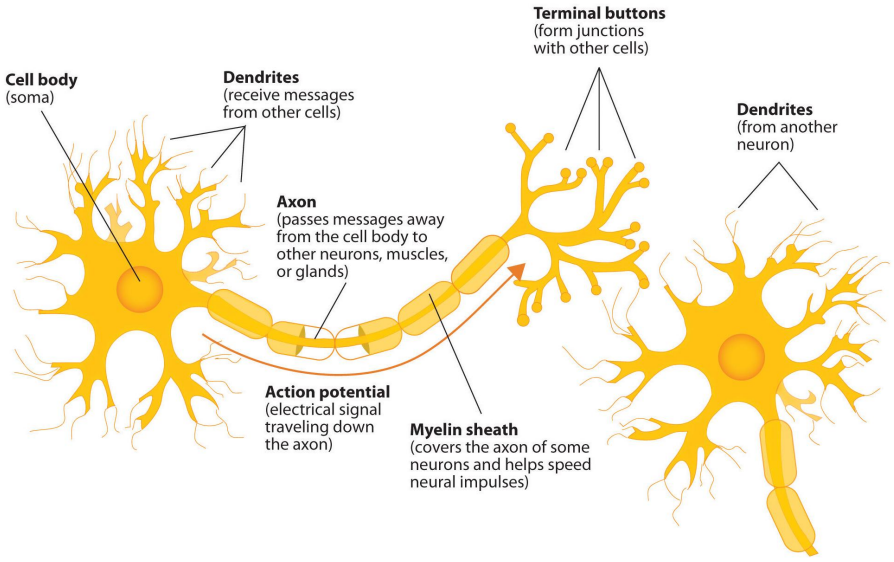


Figure 2.3: The neuron model.

- *Dendrite* is the input terminal ot the cell that receive messages from other connected cells
- *Synapses*: permits a neuron to pass an electrical or chemical signal to another neuron or to the target effector cell.

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.

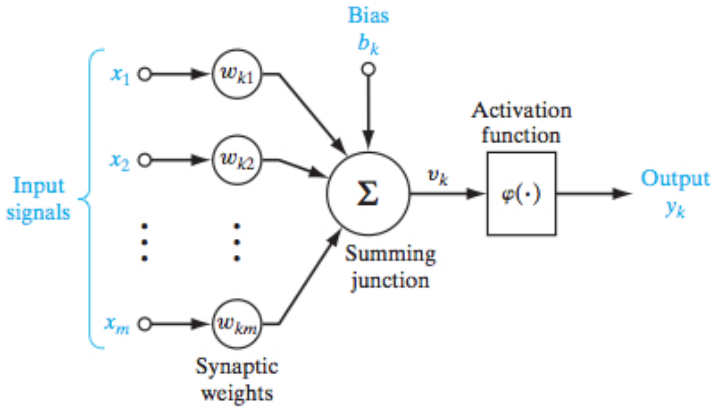


Figure 2.4: The artificial neuron model.

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

The *Artificial Neural Networks* (ANNs) are defined as *Massively parallel distributed processors made up of simple processing units having a natural propensity for storing experiential knowledge and making it available for use* (Haykin, 2008).

An ANN resembles the brain in two aspects:

1. Knowledge is acquired by the network from its environment through a learning process;

2. Synaptic weights are used to store the acquired knowledge.

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 modeling 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.

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 2.4:

- 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 neural model also includes an externally applied *bias*, denoted by b_k.

Therefore, the mathematical description of neuron activity can be defined as:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.21}$$

$$y_k = \varphi\left(u_k + b_k\right) \tag{2.22}$$

where:

- $x_1, x_2, \cdots, x_m$ are the input signals;
- $w_{k1}, w_{k2}, \cdots, w_{km}$ are the respective synaptic weights of neuron k;
- u_k is the linear combiner output due to the input signals;
- b_k is the bias;
- $\varphi(\cdot)$ is the activation function;
- y_k is the output signal of the neuron.

The types of *activation non-linear functions* $\varphi(x)$ are:

- the *threshold function*: in engineering, this form of a threshold function is commonly referred to as a Heaviside function;

$$\varphi\left(v\right) = 1 \quad if \quad v \geq 0 \tag{2.23}$$

$$\varphi\left(v\right) = 0 \quad if \quad v < 0 \tag{2.24}$$

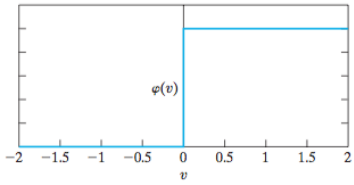


Figure 2.5: The threshold non-linear function.

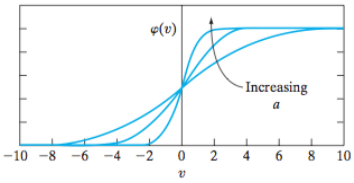


Figure 2.6: The sigmoid non-linear function.

- 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:

$$\varphi(v) = \frac{1}{1 + exp(-av)} \tag{2.25}$$

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

$$\varphi(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.26}$$

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

$$\varphi(x) = \max(0, x) \tag{2.27}$$

- 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.

$$\varphi(x_k) = \frac{e^{x_k}}{\sum_{j=1}^{N} e^{x_j}} \text{ for } k = 1, \ldots, K \tag{2.28}$$

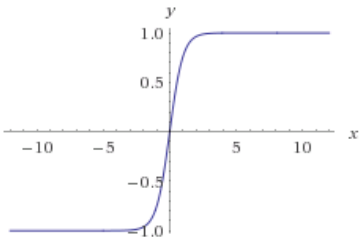


Figure 2.7: The *tanh* non-linear function.

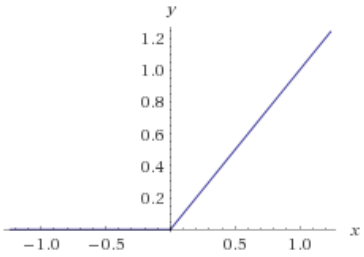


Figure 2.8: The *ReLU* non-linear function.

In past years, many models of neurons and neural architectures have been proposed in the literature, each with its peculiarities. A brief description of the leading models used in the works described in the following chapters will now be given:

1. *Multilayer Feedforward Networks* - (FFNN):

   it is characterized by the presence of one or more hidden layers, whose computation nodes are correspondingly called *hidden neurons* (or hidden units); the term *hidden* refers to the fact that this part of the neural network is not seen directly from either the input or output of the network. The function of hidden neurons is to intervene between the external input and the network output in some useful manner. By adding one or more hidden layers, the network is enabled to extract higher-order statistics from its input.

   The MLP is a well known kind of artificial neural network introduced in 1986 [45]. 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 stochastic gradient descent with error back-propagation algorithm is used for the supervised learning of the network. 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

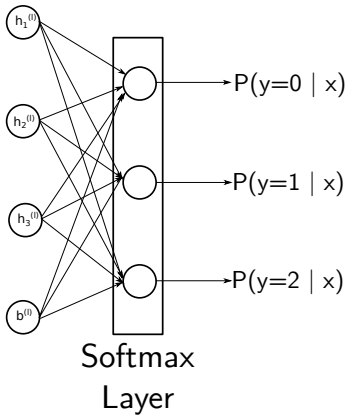


Figure 2.9: The *softmax* layer in a neural network classifier.

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:

$$g(\mathbf{u}[n]) = \varphi\left(\sum_{j=1}^{D} w_j u_j[n] + b\right), \tag{2.29}$$

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 $\mathbf{f} : \mathbb{R}^D \to \mathbb{R}^{D'}$, where D' is the size of the output vector, so:

$$\mathbf{f}(\mathbf{u}[n]) = \varphi\left(\mathbf{b}_2 + \mathbf{W}_2\left(\varphi\left(\mathbf{b}_1 + \mathbf{W}_2 \cdot \mathbf{u}[n]\right)\right)\right), \tag{2.30}$$

where $\mathbf{W}_i$ and $\mathbf{b}_i$ are the respective synaptic weights matrix and the bias vector of the i-th layer. The behaviour of this architecture is parametrized by the connection weights, which are adapted during the supervised network training.

2. *Convolutional Neural Networks*(CNN)

   Convolutional neural networks are feedforward neural networks similar to multilayer perceptron, with some special layers.

   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. Each hidden neuron is thus

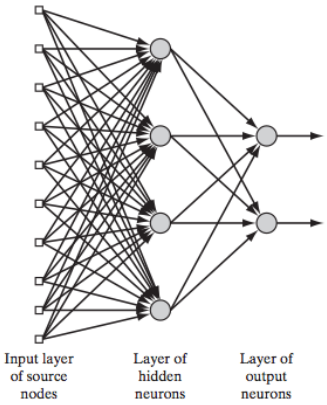


Figure 2.10: The Multilayer Feedforward Network.

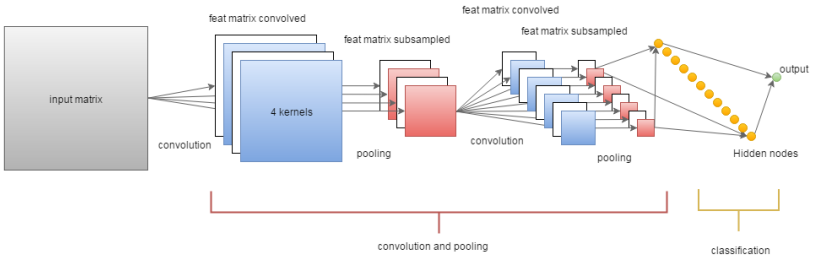


Figure 2.11: The Convolutional Neural Network.

connected to a local receptive field, and all the neurons form a matrix called *feature map*. 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 main advantages of this network is the robust pattern recognition system characterized by a strong immunity to pattern shifts.

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.

CNN is a feed-forward neural network [46] usually composed of three types of layers: convolutional layers, pooling layers and layers of neurons. The convolutional layer performs the mathematical operation of convolution between a multi-dimensional input and a fixed-size kernel. Successively, a non-linearity is applied element-wise. The kernels are generally small compared to the input, allowing CNNs to process large inputs with few trainable parameters. Successively, a pooling layer is usually applied, in order to reduce the feature map dimensions. One of the most used is the *max-pooling* whose aim is to introduce robustness against translations of the input patterns. 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. For clarity, the units contained in this layer will be referred as *Hidden Nodes* (HN).

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:

$$\mathbf{h}_m = \varphi\left(\sum_{d=1}^{D_3} \mathbf{W}_m * \mathbf{u}_d + \mathbf{b}_m\right), \tag{2.31}$$

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., $\mathbf{h}_m \in \mathbb{R}^{D_1 \times D_2}$. Please note that for the sake of simplicity, the time frame index n has been omitted. Commonly, (2.31) is followed by a pooling layer in order to be more robust against patterns shifts

in the processed data, e.g. a max-pooling operator that calculates the maximum over a $P_1 \times P_2$ matrix is employed.

A *Deep Learning* definition: *A class of machine learning techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification.* Artificial Neural Networks are often referred as deep when they have more than 1 or 2 hidden layers.

2.3.2 Stochastic gradient descent (SGD)

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 $f(x)$:

$$f(x + \epsilon) \approx f(x) + \epsilon\, f'(x). \tag{2.32}$$

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 infinitesimal 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.

$$\mathbf{x'} = \mathbf{x} - \epsilon\, \nabla_{\mathbf{x}} f(\mathbf{x}), \tag{2.33}$$

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:

$$J(\theta) = \mathbb{E}(L(\mathbf{x}, y, \theta)) = \frac{1}{m} \sum_{i=1}^{m} L(\mathbf{x}^{(i)}, y^{(i)}, \theta), \tag{2.34}$$

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

$$\nabla_\theta J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(\mathbf{x}^{(i)}, y^{(i)}, \theta). \tag{2.35}$$

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

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:

$$\theta \leftarrow \theta - \epsilon\, \mathbf{g} \tag{2.36}$$

where ϵ is the learning rate.

2.3.3 Autoencoder

An Autoencoder is a kind of neural network typically consisting of only one hidden layer, trained to set the target values to be equal to the inputs.

$$\tilde{x} = f(W_2 h(x) + b_2) \tag{2.37}$$

Given an input set of examples $\mathcal{X}$, autoencoder training consists in finding parameters $\theta = \{W_1, W_2, b_1, b_2\}$ that minimize the Reconstruction Error:

$$\mathcal{J}(\theta) = \sum_{x \in \mathcal{X}} \|x - \tilde{x}\|^2 \tag{2.38}$$

Defining M the number of hidden units, and N the number of input units, output units, features size:

- (a): $M = N \rightarrow$ Basic Autoencoder (AE);

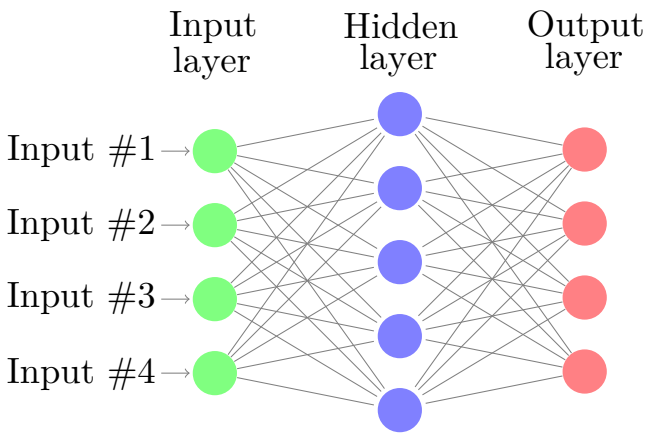


Figure 2.12: The different types of Autoencoders.

- (b): $M < N \rightarrow$ Compression Autoencoder (CAE);
- (c): $M > N$ and Gaussian Noise $\rightarrow$ Denoising Autoencoder (DAE);

An AE – a kind of neural network typically consisting of only one hidden layer –, sets the target values to be equal to the input. It is used to find common data representation from the input [47, 48]. Formally, in response to an input example $x \in \mathbf{R}^n$, the hidden representation $h(x) \in \mathbf{R}^m$ is

$$h(x) = f(W_1 x + b_1), \tag{2.39}$$

where $f(z)$ is a non-linear activation function, typically a logistic sigmoid function $f(z) = 1/(1+\exp(-z))$ applied component-wisely, $W_1 \in \mathbf{R}^{m \times n}$ is a weight matrix, and $b_1 \in \mathbf{R}^m$ is a bias vector.

The network output maps the hidden representation h back to a reconstruction $\tilde{x} \in \mathbf{R}^n$:

$$\tilde{x} = f(W_2 h(x) + b_2), \tag{2.40}$$

where $W_2 \in \mathbf{R}^{n \times m}$ is a weight matrix, and $b_2 \in \mathbf{R}^n$ is a bias vector.

Given an input set of examples $\mathcal{X}$, AE training consists in finding parameters $\theta = \{W_1, W_2, b_1, b_2\}$ that minimise the reconstruction error, which corresponds to minimising the following objective function:

$$\mathcal{J}(\theta) = \sum_{x \in \mathcal{X}} \|x - \tilde{x}\|^2 . \tag{2.41}$$

The minimisation is usually realised by stochastic gradient descent as in the training of neural networks. The structure of the AE is given in Figure 2.12a.

Chapter 3

Dataset

The importance of using public data sets for algorithm evaluation is very important. Only in this way can a direct comparison be made between the different approaches to determine which of these is actually the best. There are publicly available datasets for the fall detection task, the majority of them are all related to wearable or vision sensors and often include both types [49, 50, 51, 52, 53]. Since in this work, we face the problem of human fall detection from an audio perspective, a preliminary search for datasets containing audio signals has been performed. Only one dataset containing audio recording has been found [54]. However, the audio files available in the dataset are suitable for speech recognition related works rather than sound event detection. In fact, only the utterance of short sentences or interjections of the actors involved during the human falls recordings have been annotated. Although the principal limitation in that the human falls were made by volunteers by using various protections, which do not allow the correct acquisition of the acoustic pattern produced by the sound event. As in this work, several data-driven approaches for pattern recognition produced by the sound generated by the human fall are presented, the dataset [54] result useless. Given the lack of available audio datasets, we have created a suitable one in order to assess the proposed approaches. This choice was also forced by the fact that in these works an innovative acoustic sensor explicitly developed for the fall detection and described in Section 3.1 has been used. In this chapter, the instrumentation, the procedure for recording the audio corpus and its composition are described.

3.1 The floor acoustic sensor

The floor acoustic sensor (FAS) is composed of a resonant enclosure and a microphone located inside it (Figure 3.1) [55]. At the bottom of the enclosure, a membrane is in direct contact with the floor and guarantees the acoustic coupling with the surface. The inner container accommodates the microphone and is where the acoustic resonance phenomenon takes place. It can be covered by a layer of acoustic isolation material and it is enclosed by the outer container

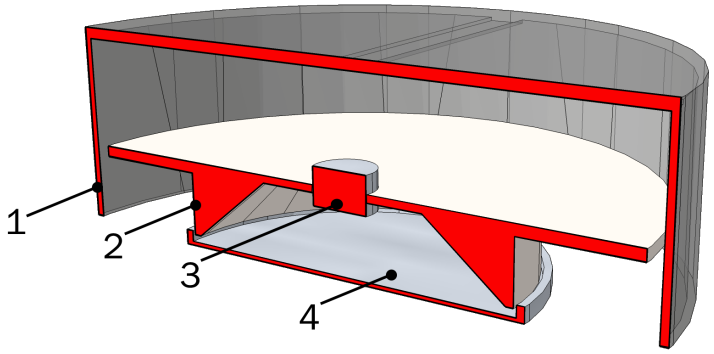


Figure 3.1: The floor acoustic sensor: conceptual scheme. 1 - The outer container. 2 - The inner container. 3 - The microphone slot. 4 - The membrane touching the floor.

that further reduces the intensity of the acoustic waves that propagate through air. The enclosure has been manufactured in Polylactic Acid with a 3D printer, its diameter is 16.5 cm and its height 5.5 cm.

Regarding the microphone, an AKG C 400 BL[1] has been inserted in the enclosure. The outer case of the microphone has been removed to extract the capsule that has then been inserted in the sensor enclosure. The AKG C 400 BL is characterized by an hypercardiod directivity pattern, thus it has been oriented so that the maximum gain is towards floor.

3.2 The fall events dataset: A3Fall

The performance of the floor acoustic sensor has been evaluated on a corpus of audio events corresponding to falls of several objects recorded in different conditions[2]. The dataset has been specifically created by the authors and it will be presented in this section.

3.2.1 The recording setup

Fall events have been recorded in 3 different rooms with the following characteristics:

- The first, is a rectangular room, hereafter named R0, measuring about $7\,\text{m} \times 2\,\text{m}$ (Figure 3.5). The room is particularly suitable for the propagation of acoustic waves through the floor since it is obtained from a

[1] http://www.akg.com/pro/p/c400-bl

[2] The dataset is available at the following URL: `http://www.a3lab.dii.univpm.it/research/fasdataset`

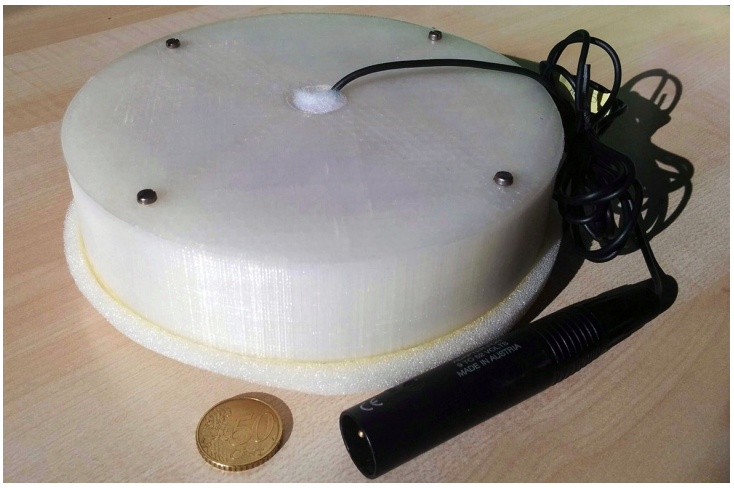


Figure 3.2: A picture of the floor acoustic sensor used during the recordings.

cantilever beam. In addition, the considerable distance of the supporting pillars facilitates the transmission of a fall vibrations through the floor.

- the second location for the recording was the university auditorium room (R1) in which the flooring is composed of fitted carpet. This makes it particularly suitable for evaluating system performance on surfaces with acoustical behavior that can mitigate the impact sound transmitted through the floor and in the air; all the recordings were performed near the auditorium stage in an area of 8×3 m.

- a recording studio (R2) was selected as the third location for its particular characteristics. Here, it was possible to make the acquisitions by placing the sensors in the live room while the audio events were performed in the control room. In particular, the sensors were positioned immediately behind the soundproof wall with the window overlooking the live room. The size of the live room is 5×7 m, while the size of the control room is 3×8 m.

The recording equipment comprises the floor sensor, a linear array of three aerial microphones (the same AKG 400 BL included in the floor sensor) and a Presonus AudioBox 44VSL sound card connected to a laptop. The microphones of the array are separated by 4 cm and positioned on a table 80 cm high. Signals were sampled at 44.1 kHz with a resolution of 32 bits. Levels were calibrated to assure the maximum dynamic range at the smallest distance.

Table 3.1: Composition of the A3Fall-v2.0 dataset.

Class	**R0**	**R1**	**R2**
	Nr. of occurrences		
Basket	64	40	40
Fork	64	40	40
Ball	64	40	40
Book	64	40	40
Bag	64	30	40
Chair	96	40	40
Table	0	40	40
Guitar Slide	0	40	40
Nipper	0	40	40
Keys	0	40	40
Hook	0	40	40
Coat Hook	0	40	40
Manikin Doll	44	0	0
Human Fall	0	40	40
	Total length (s)		
Background	2530	9055	5550

3.2.2 Description

In Table 3.1 the composition of the dataset is summarized. For the R0, the dataset comprises recordings of fall events related to everyday objects and to a human mimicking doll. The objects were chosen according to the recent literature on the topic [29] and are the following: a ball, a metal basket, a book, a metal fork, a plastic chair, and a bag (Figure 3.3). Objects have been dropped at four distances from the sensors, i.e., 1 m, 2 m, 4 m, and 6 m, and with various angles in order to reproduce realistically different fall patterns. With the exception of the chair and the basket, which have been overturned from their natural position, half of the falls has been performed at a height of 0.5 m and the other half at 1 m. For each object and for each distance, 16 fall events have been performed for a total of 64 events per object. Instead, the chair has been overturned 8 times for each side of fall (back, front, side) and for each distance, thus obtaining a total of 96 events.

Human falls have been simulated by employing the "Rescue Randy" doll[3], a professional equipment employed in water rescues. It weights 75 kg, it is 1.85 m high, and it is equipped with articulated joints. The doll is made of vinyl and its weight is distributed according to the human weight distribution chart. The doll has been dropped from upright position and from a chair, both forward and

[3]http://www.simulaids.com/1475.htm

Figure 3.3: Objects employed for creating the fall events dataset.

backward, for a total of 44 events (Figure 3.4). Differently from the everyday objects, the distribution of the fall events with the distance is not uniform: 10 events have been performed from 2 m, 18 from 4 m (7 of which from the chair), and 16 from 6 m (6 of which from the chair).

Moreover, several backgrounds sounds has been added to the dataset. Normal activities sounds have been recorded while persons were performing common actions, such as walking, talking, and dragging chairs. Three musical tracks have been played from a loudspeaker and acquired back with the FAS. The first track contained classical music[4], while the second[5] and the third[6] rock music. Musical tracks and normal activities sounds have been divided in segments whose lengths have mean and standard deviation estimated from instances of fall events. In addition, they have been employed alone and to create noisy versions of human and object falls occurrences in order to assess the algorithm in presence of interferences.

In R2 and R1 other every-days objects,in addition to those used in R0, have been recorded for a total of 12 different object fall classes and 1420 instances. While the manikin doll has been used only in R0, in R1 and R2 a total 80 human falls have been performed by 4 people. These falls were performed in different ways: forward, backward and on the side, trying to use the arms to cushion the fall and without any protections. As in R0, also in R2 and R2 all events were performed from 1, 2, 4 and 6 m away from the FAS.

As shown in Table 3.1 background noises have been recorded also in R1 and R2 rooms rooms, which include: human activities noise as, i.e., footsteps, human and phone conversation, dragging objects and so on; classic, rock and

[4]W. A. Mozart, "Piano trio in C major"
[5]Led Zeppelin, "Dazed and confused"
[6]Led Zeppelin, "When the levee breaks"

(a) Fall from upright position. (b) Fall from the chair.

Figure 3.4: Falls of the "Rescue Randy" doll from upright position (a) and from the chair (b).

pop music played from loudspeakers; TV shows like newscast and satiric.

Since the data relating to rooms R1 and R2 have been collected at different times to those of room R0, in the following chapters, for each proposed approach, it will be specified which subset of the total dataset has been used as well as the usage of the noisy version of falls events.

3.2.3 Signal analysis

The signal related to the same fall event acquired with the floor sensor and with the aerial microphone exhibits different spectral characteristics. In this section and in depth analysis of the audio signals acquired in the R0 room is presented. Figure 3.6 shows the spectrograms of a doll fall acquired with the floor sensor (above) and with the aerial microphone (below) in the clean acoustic condition. Observing the figures, it can be noticed that the aerial microphone is more sensitive to high frequencies, in particular to the ones above 1.5 kHz. On the contrary, the majority of the energy of the signal acquired with the floor sensor concentrates below 1 kHz.

This is even more evident by plotting the values of the mel coefficients (Figure 3.7): the first and second mel channels of the FAS, corresponding to the frequency bands 0–128.10 Hz and 61.30-200.60 Hz, are higher respect to the aerial microphone. Channels 3 to 7, respectively corresponding to bands 128.10–279.50 Hz and 458.70–670.70 Hz, are almost equivalent, while from channel 8 (560.30–790.80 Hz) to 29 (6654.60–8000.00 kHz) the aerial microphone mels are

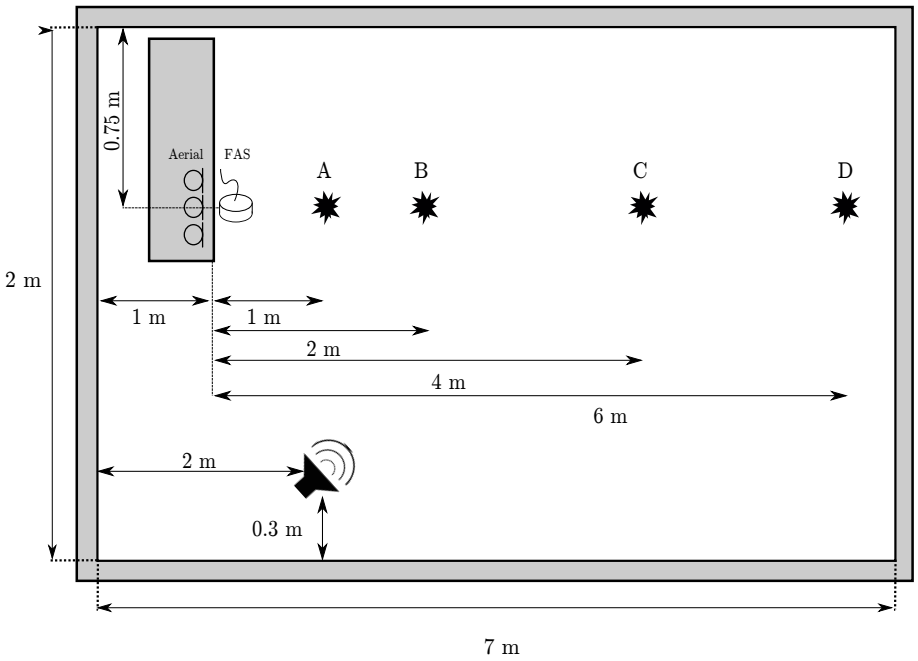
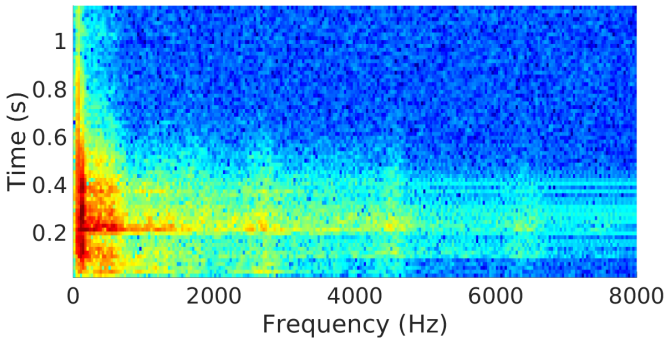


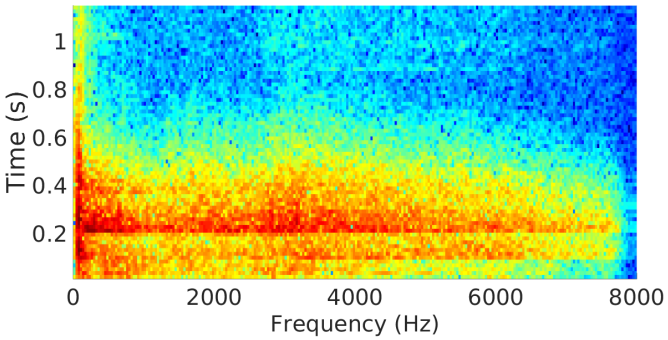
Figure 3.5: The recording room: the letters A, B, C and D indicate the positions of fall events.

greater.

The analysis of noisy signals highlights the different behaviour of the floor sensor respect to the aerial microphone in presence of external interferences. In fact, taking into account the backgrounds tracks as interferences, the floor sensor has a global signal-to-noise ratio (SNR) equal to 20.94 dB and a segmental SNR equal to 7.28 dB. The global SNR of the central aerial microphone is 8.92 dB and the segmental SNR is -1.47 dB. The values of the aerial microphone SNRs are thus considerably lower than the ones of the floor sensor.The global SNR of the floor sensor noisy dataset is 13.66 dB higher than the one of the aerial microphone, highlighting the superior ability of the former to isolate fall signals from external interferences. However, it is worth investigating how the SNR distributes over the frequency range of the acquired signals in order to have a better insight of the physical phenomenon. Figure 3.8 shows the SNR calculated for each mel channel and averaged across the noisy datasets. It can be noticed that the SNR of the floor sensor exceeds the one of the aerial microphone for channels below the fourth. Then, the opposite occurs and the SNR of the aerial microphone assumes greater values. The "valley" in the curves are due to the pitch of the music signal.



(a) Spectrogram of the signal acquired with the floor sensor.



(b) Spectrogram of the signal acquired with the aerial microphone.

Figure 3.6: Frequency content of the same fall event (file "rndy_d2st_bar_0.wav") acquired with the FAS (a) and with the aerial microphone (b).

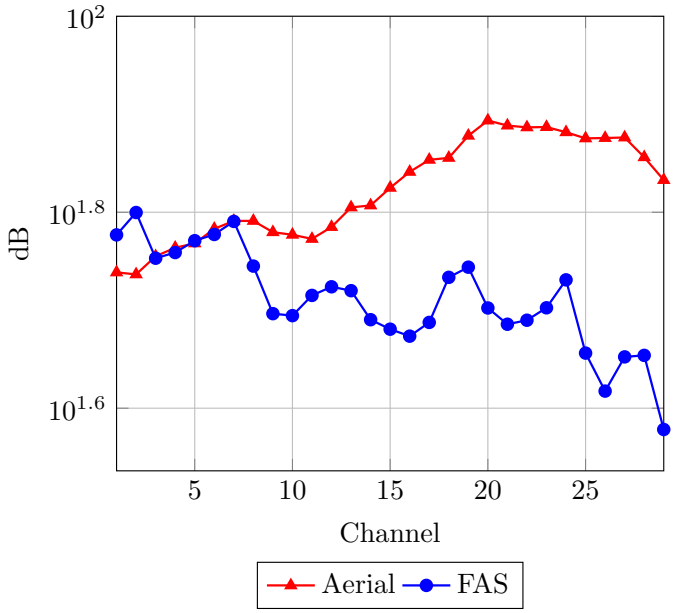


Figure 3.7: Average value of the mel channels.

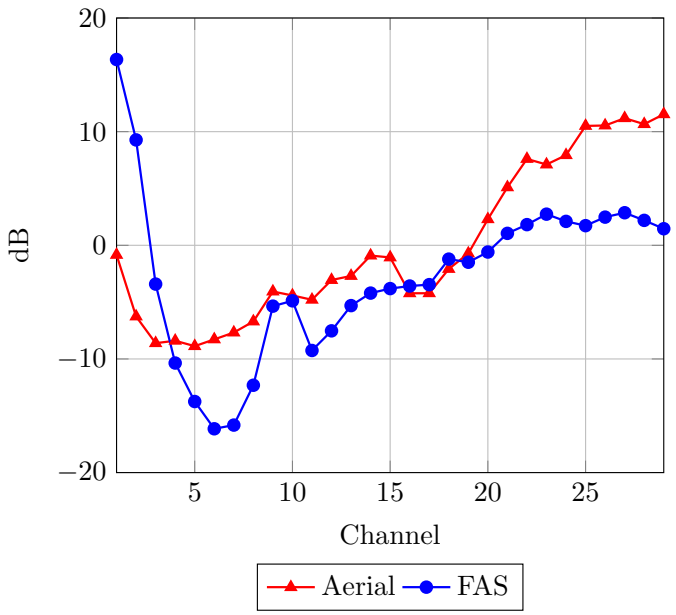


Figure 3.8: Average value of the SNR for each mel channel.

Chapter 4

Supervised Approaches

The supervised learning is the most used strategy in machine learning. It basically consists in inferring a function that maps an input to an output from a dataset composed of examples with an associated label each. In machine learning, this is an optimal scenario to work with, in which the examples for each class to classify and the exact labels are available. The supervised approaches are generally able to achieve better performance with respect to unsupervised or weakly-supervised systems because of the more precise knowledge with which they are trained. In this chapter, the supervised approach is used to work in favourable conditions in order to evaluate the quality of the created dataset as well as the performance of the FAS compared to standard aerial microphones. Below, a GMM-SVM based multi-class classifier able to discriminate which type of object has fallen to the ground in both clean and noisy sound conditions is presented first. Then the binary classifier counterpart of the previous one is tailored to discriminate a human fall with respect to other sounds

4.1 SVM based algorithm for fall classification

The fall detection task consists in recognize which object produced the pattern of a signal and it mainly consists of two subtasks: the location of the time boundaries of the fall event and the classification of that event. In this section, we concentrate on the second subtask, since the main objective is determining the performance of the FAS as compared to common aerial microphones. The entire falls detection activity will be addressed in the works presented below.

4.1.1 Porposed approach

The fall classification algorithm is composed of a feature extraction stage and a classification stage. The first extracts MFCCs from the input audio signal, while the second classifies the audio event by means of Gaussian means supervectors and SVM. Following is a detailed description of the two stages.

Feature extraction

Despite MFCCs were originally developed for speech and speaker recognition tasks, they have been successfully applied also for acoustic event classification [56] and fall detection [18].

The block-scheme of the MFCC feature extraction pipeline is shown in Figure 4.1. The first processing step is the pre-emphasis of the input signal, which consists in applying a filter whose transfer function is:

$$G(z) = 1 - \alpha z^{-1}. \tag{4.1}$$

Usually $0.9 < \alpha \leq 1.0$ and here it has been set to 0.97. The objective of pre-emphasis is to remove the DC components and to raise the high-frequency part of the spectrum.

The signal is then segmented in frames 16 ms long overlapped by 8 ms and multiplied with a Hamming window. For each frame, the Discrete Fourier Transform (DFT) is calculated and filtered with a filterbank composed of 29 triangular filters uniformly spaced on the mel scale.

Denoting with $S(i)$ the DFT of a frame and i the frequency bin, the output of the "Mel Filterbank & Frequency Integration" block in Figure 4.1 is

$$mel(k) = \sum_{i=ini(k)}^{end(k)} |S(i)|^2 W_k(i), \qquad k = 1, 2, \ldots, N \tag{4.2}$$

where $mel(k)$ is the energy of the k-th subband, $W_k(i)$ is the frequency response of the k-th filter, $ini(k)$ and $end(k)$ are starting and ending frequency indices of that filter and N is the number of filters in the bank, which in this case is 29. The terms $mel(k)$ are often named "mel coefficients".

The final steps for the calculation of the j-th MFCC $c(j)$ is the logarithm

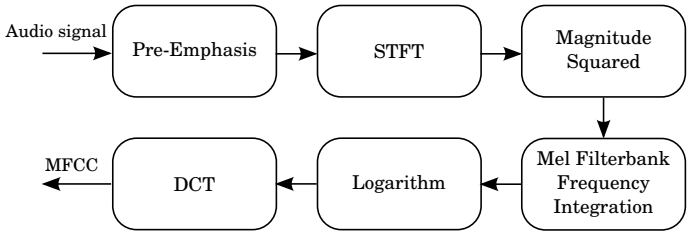


Figure 4.1: The MFCC feature extraction pipeline.

and the Discrete Cosine Transform (DCT):

$$c(j) = \sum_{k=1}^{N} \log\left[H(k)\right] \cos\left[\frac{\pi j}{N}(k-0.5)\right], \quad j = 0, 1, \ldots, M-1 \leq N \tag{4.3}$$

The set $\{c(0), c(1), \ldots, c(M-1)\}$ forms the static coefficients elements of the feature vector. Here M has been set to 13. The final feature vector is composed of 39 coefficients, i.e., the 13 static coefficients plus their first and second derivatives.

Classification stage

The approach proposed in this section is based on a One-Class Support Vector Machine (OCSVM) [39]. The general idea is that a human fall produces a sound considerably different from the ones commonly occurring in a home (e.g., voices, sounds from electronic devices, footsteps, etc.). The OCSVM is trained on a large set of "normal" sounds to detect acoustic events that deviate from normality. However, it is expected that certain acoustic events are as abnormal as a human fall (e.g., the fall of book, a chair, etc.), thus they could raise false alarms.

The classification stage employs Gaussian Mean Supervectors (GMS) and a Support Vector Machine classifier as in speaker recognition systems [41]. The algorithms consists in modelling the entire acoustic space with a Universal Background Model (UBM) represented by mixture of gaussians (Gaussian Mixture Model, GMM). The GMM is trained using the Expectation Maximization (EM) algorithm [43] on a large corpus of acoustic events. Then, for each acoustic event class in the training corpus a GMS is calculated by adapting the UBM with the Maximum A Posteriori (MAP) algorithm [44] and concatenating the adapted GMM mean values. The block diagram of the approach is shown in Figure 4.2. The final step of the training phase is the estimation of the SVM parameters. In this work, an SVM with a radial basis function has been used. The complete diagram of the training phase is shown in Figure 4.3.

Classification is performed by extracting the supervector from an input audio signal as in the training phase, and then determining the acoustic event class evaluating the SVM discriminant function (Figure 4.4). Since the number of classes is greater than two and SVMs are binary classifiers, the "one versus all" technique has been adopted [37]. LIBSVM [57] has been employed both in the training and testing phases of the SVM.

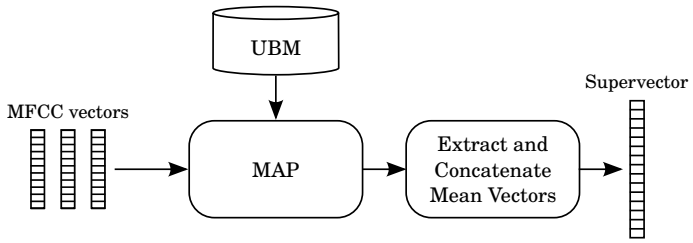


Figure 4.2: Block scheme for extracting a gaussian mean supervector from MFCCs and a trained UBM.

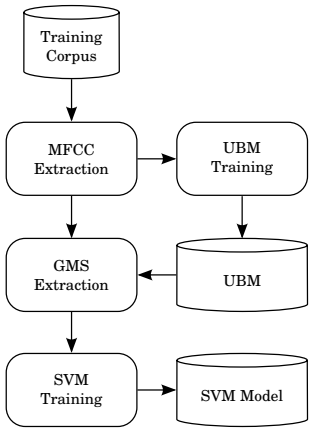


Figure 4.3: Block scheme for training the UBM and SVM models.

4.1.2 Dataset

All the experiments have been conducted data related to R0 room only. In particular, in order to compare the performance of the FAS with a standard aerial microphone, not all data concerning all the aerial microphones array have been used but only the one of the microphone closest to the wall (see Figure 3.5). Moreover, signals have been downsampled to 16 kHz and the resolution has been reduced to 16 bit. In order to obtain a balanced dataset, for each object and for each distance, 11 fall events have been randomly picked for a total of 44 events per object, while all the 44 simulated human fall with the manikin has been selected. In addition, the three musical track recored in the R0 room have been employed to create a noisy version of the dataset by digitally adding a random segment of the recorded musical tracks to the "clean" datasets as was done for the analysis of the signals in Section 3.2.3. In particular, the analysis presented in Section 3.2.3 suggested the authors that the standard MFCC extraction pipeline can be modified in order to better exploit the properties the floor acoustic sensor. In particular, the analysis led to the following considerations:

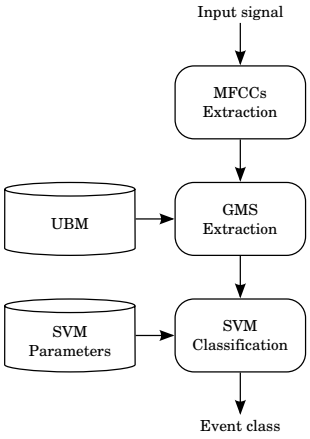


Figure 4.4: Block-scheme of the fall classification phase.

Table 4.1: Data related to R0 room used in this work.

Classes	**Nr. of occurrences**
Basket	44
Fork	44
Ball	44
Book	44
Bag	44
Chair	44
Manikin Doll	44
Backgrounds	**Total length (s)**
Classic Music	882
Rock Music	616

- The pre-emphasis filter is an inheritance of automatic speech recognition systems, where the input signal is the human voice. Since the effect of the filter is to enhance the high frequency components of the signal and the floor sensor is more sensitive to low frequencies, removing the pre-emphasis could improve the classification performance.

- The majority of the energy of the signals acquired with the floor acoustic sensor is concentrated at frequencies below 1.5 kHz. This suggest that introducing a low-pass filter that removes low SNR frequencies may improve the classification performance.

In Table 4.1 the data used for this approach are summarized.

4.1.3 Experimental setup

The MFCC extraction pipeline has been firstly parametrised as described in Section 4.1.1, then, based on the study presented in the previous section, two additional pipelines have been tested:

- no pre-emphasis (NOPRE): the pre-emphasis stage has been removed from the feature extraction pipeline;

- no pre-emphasis plus low-pass filtering (NOPRELP): the pre-emphasis stage has been removed and the maximum frequency of the mel filterbank has been set to 4 kHz.

The standard MFCC extraction pipeline described in Section 4.1.1 will be denoted with "STD" in the results.

Regarding the UBM, it has been trained until convergence with the EM algorithm setting the threshold value to 10^{-3}. The same value has been employed in the MAP adaptation algorithm, but the maximum number of iterations was set to 5.

Due to the limited amount of data, the experiments have been conducted with the leave-one-label-out method: fall events recorded at all distances but one were employed for training and validation, while the remaining for testing. The validation phase consisted in searching for the number of components of the UBM and for the SVM hyperparameters which yielded the best results. In particular, the number of components of the UBM assumed the values $\{1, 2, \ldots, 64\}$, while the SVM hyperparameters C and γ the values $\{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ and $\{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$ respectively.

The performance was evaluated both on clean data and on data corrupted with the music interference. In particular, the system was evaluated in matched condition, where the acoustic scenario of the training, validation and test data is the same, in mismatched condition, where training data is clean and testing data is corrupted with music, and in multi-condition, where training data and testing data contain both clean signals and corrupted signals. In the latter case, the training, validation and test sets have been divided so that they contain 1/3 of clean data and 2/3 of noisy data.

The performance has been evaluated in terms of precision (P), recall (R) and F_1-Measure (F) per class and the values have then been averaged. The

metrics definitions for class i is:

$$P_i = \frac{C(i,i)}{\sum_j C(j,i)}, \tag{4.4}$$

$$R_i = \frac{C(i,i)}{\sum_j C(i,j)}, \tag{4.5}$$

$$F_i = \frac{2P_iR_i}{P_i + R_i}, \tag{4.6}$$

where $C = [C(i,j)]$ is the confusion matrix.

4.1.4 Results

Results in matched condition

Figure 4.5 shows the results obtained in matched condition. As shown in the figure, the FAS exceeds the F_1-Measure of the aerial microphone both with clean and noisy signals regardless the feature setup employed. In particular, in clean conditions the aerial microphone achieves the highest F_1-Measure with standard MFCCs, while the FAS with the NOPRELP ones, which results in 6.50% absolute improvement. In noisy conditions, the aerial microphones best performance is again achieved with STD features, while the FAS one is achieved with the NOPRE features, with the NOPRELP setup performing almost the same. The absolute improvement of the FAS respect to the aerial microphone is 5.36%.

Focusing on the FAS performance with the various feature extraction pipelines, the highest F_1-Measure is obtained by removing the pre-emphasis and low-pass filtering the signals (NOPRELP) in clean conditions. Notice, however, the standard pipeline performs almost the same, thus in this condition the influence is minimal. The sole removal of the pre-emphasis, on the other hand, is detrimental for the classification performance. The opposite occurs in noisy condition, where the removal of the pre-emphasis improves the results by 3.99%. Low-pass filtering, on the other hand, does not give further improvements. Overall, the feature pipeline that gives the highest F_1-Measure is the NOPRELP, as argued in Section 3.2.3.

Further insights on the results are given by the confusion matrices of the aerial microphone (Table 4.2) and of the FAS (Table 4.3) for the feature pipeline giving the highest F_1-Measure. With the only exception of the basket falls, the FAS is able to achieve superior performance for all the objects in the dataset. In particular, the doll F_1-Measure improves by 9.89% respect to the aerial microphone. The object exhibiting the lowest performance regardless the sensor is the book: this can denote a limit in the algorithm suggesting that further improvements could be obtained by properly modifying the features or the

Table 4.2: Aerial microphone confusion matrix related to the STD configuration in clean condition. The average precision is 91.48%, the average recall 91.23%, and the average F_1-Measure 91.04%.

	Doll	**Bag**	**Ball**	**Basket**	**Book**	**Chair**	**Fork**	**F_1**
Doll	93.18	0	0	0	2.27	4.55	0	90.11
Bag	0	86.36	0	0	13.64	0	0	82.61
Ball	0	0	100.00	0	0	0	0	100.00
Basket	2.27	0	0	97.73	0	0	0	98.85
Book	0	22.73	0	0	77.27	0	0	78.16
Chair	11.36	0	0	0	4.55	84.09	0	89.16
Fork	0	0	0	0	0	0	100	100.00
Precision	87.27	79.17	100.00	100.00	79.07	94.87	100.00	

Table 4.3: FAS confusion matrix related to the NPRELP configuration in clean condition. The average precision is 98.13%, the average recall 98.05%, and the average F_1-Measure 98.06%.

	Doll	**Bag**	**Ball**	**Basket**	**Book**	**Chair**	**Fork**	**F_1**
Doll	100.00	0	0	0	0	0	0	100.00
Bag	0	97.73	0	0	2.27	0	0	97.73
Ball	0	0	100.00	0	0	0	0	100.00
Basket	0	0	0	95.45	2.27	2.27	0	97.67
Book	0	2.27	0	0	97.73	0	0	94.51
Chair	0	0	0	0	4.55	95.45	0	96.55
Fork	0	0	0	0	0	0	100.00	100.00
Precision	100.00	97.73	100.00	100.00	91.49	97.67	100.00	

classifier.

Results in mismatched condition

Figure 4.6 shows the results obtained in mismatched condition, i.e., with training and validation performed on clean data and testing on noisy data. As expected, both the performance of the aerial microphone and of the floor sensor decrease. Notice, however, that regardless the features employed, the floor sensor is able to achieve considerably higher F_1-Measures respect to the aerial microphone. In particular, the floor sensor highest F_1-Measure, obtained with the NPRE features, exceeds the one of the aerial microphone obtained with the standard MFCC pipeline by 8.76%. This value confirms that the higher SNR of the floor sensor signals actually results in better classification performance.

Regarding the features, the hypothesis that the pre-emphasis could be detrimental for the performance of the sensor is confirmed: indeed, the highest F_1-Measure has been obtained with the NOPRE configuration and exceeds the one of the standard MFCC pipeline by 1.32%. Differently, the introduction of the low-pass filter does not result in better performance respect to the NOPRE case.

Results in multicondition

Figure 4.7 shows the results obtained in multicondition, i.e., when training and test sets both contain clean and noisy data. The results further confirm the superiority of the FAS respect to the aerial microphone, with the first reaching 90.82% using the NOPRE features and the latter reaching 85.28% using the standard MFCC pipeline. Regarding the features, Figure 4.7 confirms that removing the pre-emphasis indeed improves classification results, while introducing the low-pass filtering does not give further benefits.

4.1.5 Remarks

The experimental results taken as a whole provide important insights on the system behavior and allow us to express possible future directions for its improvement. In particular, the experiments demonstrated the superiority of the floor acoustic sensor compared to the aerial microphone. The results in matched condition are notable, since the F_1-Measure is greater than 98% in clean condition and close to 90% in noisy condition. The experiment in matched condition is important to compare the performance of the two sensors, but it puts the classifier in a favourable scenario. The mismatched condition is closer to a real scenario, where the classifier is trained on a dataset whose characteristics differ from testing ones. The floor sensor still achieves higher results compared to the aerial microphone, but respect to the matched condition they are considerably lower. This suggests that there is room for improvement both on the sensor side and on the algorithmic side.

Regarding the sensor, the insertion of isolating material in the enclosure would further reduce the impact of external interferences. On the algorithmic side, different low-level features could be employed instead of MFCCs. In addition, the SVM classifier here employed has not been designed to recognize a specific class. In the latter case, e.g., focusing on the human fall class, the problem can be tackled in two ways: as a two class problem, where one class is the class of interest and the other comprises "the rest of the world" that will be addressed in Section 4.2 or as a novelty detection problem, where the "novel" events are represented by the specific class occurrence that will be addressed in Section 5.1. In both cases, the classification task is simpler, since the number of classes to discriminate is reduced and the overall performance is expected to increase [37].

4.2 Binary SVM based classifier for human fall detection

The classification algorithm proposed here is based on the previous work (Section 4). It employs both low-level features, i.e., MFCCs [58], and high-level features, i.e., Gaussian Mean Supervectors which are then used by a Support Vector Machine to distinguish falls from no-falls. Despite MFCCs were originally developed for speech and speaker recognition tasks, they have been successfully applied also for acoustic event classification [59] and fall detection [18]. Differently from the algorithm presented in the previous section, where the algorithm discriminated falls of general objects, here the focus is specifically on the classification of human falls. The classifier, thus, has been designed to discriminate between two classes: human falls and generic sound events. In order to assess the performance of the approach, the dataset described in Section 4.1.2 has been augmented with instances of everyday sounds (speech, footsteps, etc.), thus making the task more challenging. As a reference, results employing the original dataset (Section 4.1.2) are also reported.

4.2.1 Proposed approach

The classification algorithm is composed of two main parts. The first is the features extraction phase where we have extract the Mel Frequency Cepstral Coefficients from all audio files that comprise the dataset. For doing this, the feature extraction pipeline is the same used in Section 4.1.1. In particular, the signal is segmented in frames 16 ms long overlapped by 8 ms, the parameter α of the pre-emphasis filter has been set to 0.97 and the number of filters which compose the filterbank has been set to 29. At the end, after the Discrete Cosine Transform, 13 statics coefficients are extracted that, together with their first and second derivatives, form the final feature vector of a signal. The classification phase is similar to that one adopted in Section 4.1.1: first it uses a mixture of gaussians (GMM), trained on a large corpus of audio events with the Expectation Maximization algorithm to model the acoustic space (Universal Background Model, UBM). Then, for each audio segment, the Maximum a Posteriori (MAP) algorithm is used to calculate the Gaussian Mean Supervector (GMS) from the MFCCs. In contrast with the previous work Section 4.1.1, where we used a multi-class approach, here we employ a binary SVM to discriminate the class "fall" from "rest" which allows to distinguish human falls from the other types of sounds. In addition, the class decision is usually performed by evaluating the sign of the SVM discriminative function, which ultimately consists in deciding whether in example belongs to a class by setting a threshold equal to zero. However, in a human fall classification task, it is important to

Table 4.4: Data related to R0 room used in this work.

Classes	**Nr. of occurrences**
Basket	44
Fork	44
Ball	44
Book	44
Bag	44
Chair	44
Manikin Doll	44
Backgrounds	**Total length (s)**
Classic Music	882
Rock Music	616
human activities	665

minimize the probability of missing a fall event, i.e., false negatives. In order to push the system towards this direction, we decided to consider the entire value of the SVM discriminative function, and then to set an appropriate threshold in order to minimize the occurrence of false negatives.

4.2.2 Dataset

As can be seen in the Table 4.4, the dataset used for training and assess the proposed method is an enlarged version of the one used in Section 4.1. Moreover, in order to further stress the system, a 20 minutes long recording session in which 2 persons have produced everyday noises such as talking, walking, dragging chairs, and playing with the ball was used. Then this background recording has been divided in 665 sub-tracks. The lengths of these sub-tracks have been randomly generated with gaussian distribution. Mean value and standard deviation of this distribution have been calculated based on the lengths of the other files that form the dataset. In addition to the clean dataset, a noisy version has been created to assess the performance in noisy condition. The noisy dataset consists in a musical background recorded with both sensors and digitally added to the clean events.

4.2.3 Experiments

In this section, the experimental procedure is firstly discussed and then the algorithm performance is presented. In the experiments, the signals of the dataset described above have been downsampled to 8 kHz and the bit depth has been reduced to 16 bit. Both the choice of the sampling frequency and the

choice of MFCCs are justified by the analysis performed in the Section 3.2.3, where it was shown that the signals recorded with the FAS have the majority of the energy concentrated at low frequencies (below 1 kHz). Indeed, the mel scale used in the feature extraction pipeline has a higher resolution at low frequencies, that allows to better describe the portion of the spectrum where the majority of the energy resides. In addition, the algorithm has been evaluated using two features extraction pipelines:

- the first is the same described in the Section 4.1.1 and will be denoted as STD;
- the second pipeline does not include the pre-emphasis filter and will be denoted with NOPRE.

The experiments have been conducted with a 4-fold cross-validation strategy and a three-way data split in three different operating conditions:

- matched, where the training, validation and test sets share the same acoustic condition, i.e., clean or noisy;
- mismatched, where the training set is composed of clean signals while the validation and test sets are composed of noisy signals;
- multicondition, where the training, validation and test sets contain both clean and noisy data. In this case the sets have been divided so that they contain 1/3 of clean data and 2/3 of noisy data.

The tests have been conducted also without using the signals of everyday noises, i.e., using the same dataset employed in Section 4.1.2. The performance is evaluated in terms of F_1-Measure per class and the values have then been averaged to obtain a single performance metric. To better describe the algorithm behavior, we have used the Detection Error Trade-off (DET) curve, as defined in [60]. This graph allows evaluating the false alarm probability when miss probability is equal to 0 (henceforward named as FPM0), which is particularly relevant in a fall classification task.

4.2.4 Results

Results in matched condition

Figure 4.8 shows the results obtained in the matched condition case study using the enlarged version of the dataset. In Figure 4.8a, the FAS achieves higher F_1-Measures both in clean and noisy conditions. In the first case, the floor sensor exceeds the F_1-Measure of the aerial microphone obtained with the standard MFCC pipeline by 0.11%. In the noisy condition case study, the FAS with STD

features achieves an F_1-Measure greater than 0.3% with respect to the aerial microphone with NOPRE features.

The DET curves are shown in Figure 4.8b. Note that the lines relative to the FAS in clean condition are both absent. This is because independently from the threshold, either the miss probability or the false alarm probability is 0 and the DET curves assume values towards minus infinite (in logarithmic scale). This means that the FPM0 are 0 for both these configurations, while the lower FPM0 for the aerial microphone in clean condition is 1.3%. Regarding the tests with noisy signals, the performance difference between the two sensors increases, since the FPM0 is equal to 2.2% with the FAS (NOPRE) and is equal to 12.5% with the aerial sensor (STD).

In order to facilitate the analysis, Table 4.5 summarises the results obtained with the dataset version used in Section 4.1.2, which does not comprise everyday noises. As it can be observed, with respect to the previous results, the performance decreases in all tests and for both the F_1-Measure and the FPM0, although the gap between the FAS and the aerial microphone increases.

Results in mismatched condition

The mismatched condition is the most difficult case for the classifier. As expected, the performance decrease for both sensors. Focusing on Figure 4.9a, the best F_1-Measure is obtained with the NOPRE pipeline for both the FAS and aerial microphone, and is equal to 99.14% and 98.43% respectively. A consistent performance difference between the two sensors can be observed in Figure 4.9b, where the smallest FPM0 for the FAS, obtained with NOPRE, is around 13% while for the aerial one is 95%, this time obtained with STD.

The results of the mismatched experiments obtained with the dataset without everyday noises are reported in Table 4.7a. Regarding the F_1-Measure, a performance decrease can be observed compared to the results in Figure 4.9a. Differently, the aerial microphone FPM0 improves, but it is again below the FPM0 achieved by the FAS with NOPRE MFCCs (12.5%).

Results in multicondition

Figure 4.10 shows the results in the multicondition case. The FAS superiority is confirmed, since it achieves an F_1-Measure equal to 99.78% regardless the feature extraction pipeline, while the aerial sensor obtains an F_1-Measure equal to 99.19% with the STD pipeline.

Regarding the DET plot (Figure 4.10b), an FPM0 equal to 2.9% is achieved by the FAS, but differently from the previous cases, with the STD feature. The aerial microphone achieves an FPM0 equal to 9.8% with the NOPRE pipeline.

In the last table (Table 4.7b) are shown the result of the multicondition tests

Table 4.5: Fall classification performance in matched condition with the dataset excluding everyday noises.

		STD		NOPRE	
		F_1-Measure	FPM0	F_1-Measure	FPM0
Clean	Aerial	96.29	9.85	93.11	4.95
	FAS	98.81	0.00	100.00	0.00
Noisy	Aerial	97.22	43.00	96.92	73.00
	FAS	99.63	6.05	99.62	9.85

Table 4.6: Fall classification performance in mismatched condition (a) and multicondition (b) with the dataset excluding everyday noises. F_1 denotes the F_1-Measure.

(a)

%	STD		NOPRE	
	F_1	FPM0	F_1	FPM0
Aerial	96.20	87.50	95.44	76.00
FAS	97.80	34.50	98.11	12.50

(b)

%	STD		NOPRE	
	F_1	FPM0	F_1	FPM0
Aerial	96.80	16.50	96.78	31.4
FAS	99.62	0.00	99.43	0.00

obtained by using the smaller dataset. Again there is an overall decrease for the F_1-Measure, greater for the aerial microphone, while the FAS exhibiting a FPM0 equal to 0% contrary to the increasing FPM0 for the aerial sensor.

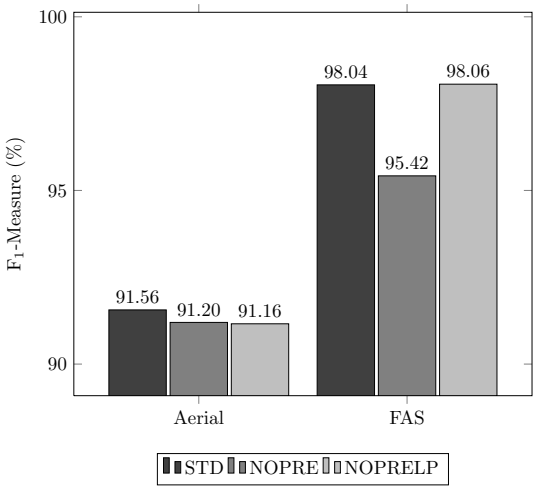
4.2.5 Remarks

In this section, a human fall classification system based on the previous work discussed in Section 4.1 has been described. The main sensor used in the FAS the sensor operates similarly to stethoscopes, with a microphone embedded in a resonant enclosure and a membrane in contact with the floor that captures the acoustic waves resulting from a fall. The performance of that sensor has been compared with the one obtained by a standard aerial microphone. The classification algorithm extracts MFCC features from the signal acquired with the FAS, and then discriminates a fall from a generic event by using GMM supervectors and an SVM classifier. Differently from the works discussed in Section 4.1, here we specifically addressed the human fall classification task by designing the classifier to discriminate human falls from other events.

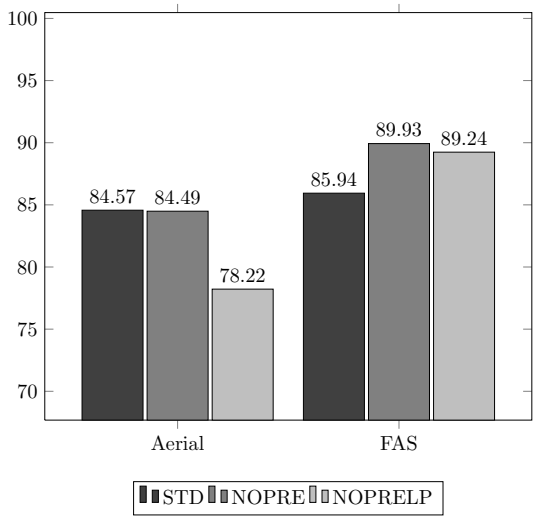
The performance of the system has been evaluated on a corpus containing recordings of several events: falls of a human mimicking doll, falls of common objects and everyday noises (speech, footsteps, etc.). In order to assess the performance of the solution in adverse acoustic conditions, a noisy version of the dataset has been created. The experiments have been performed in three operating conditions: matched, mismatched and multicondition, and the performance has been evaluated in terms of average F_1-Measure and false alarm probability when the miss probability is equal to 0. The superiority of the FAS resulted evident in all the addressed conditions, in particular with an F_1-Measure equal to 100% and an FP0 equal to 0% in clean matched conditions, and an F_1-Measure equal to 99.14% and an FP0 equal to 13% in noisy mismatched conditions.

Moreover, by looking at the results obtained with the two different features pipeline, the FAS based solution always show robust performance and it is difficult to determine which represents the best choice. The increasing performance (F_1-Measure) obtained with the largest version of dataset, in truth, are due to the fact that signals corresponding to the everyday noises are extremely easy to classify being very different from a fall.

Given the nature of the floor acoustic sensor employed, it is questionable how it can perform in case the falls occurs in a different room respect to the one where the FAS is placed or in different scenario as falls occurs in presence of furniture or with a different paving. Both this aspect will be addressed in Section 6.6.



(a)



(b)

Figure 4.5: Fall classification performance in matched condition with clean (a) and noisy signals (b).

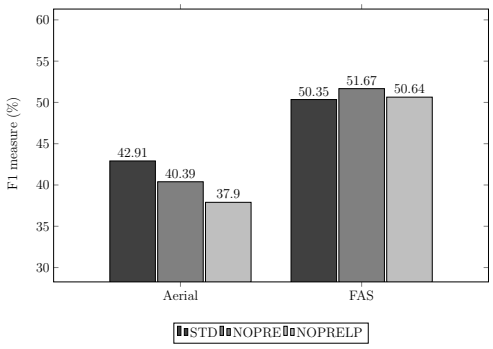


Figure 4.6: Fall classification performance in mismatched condition.

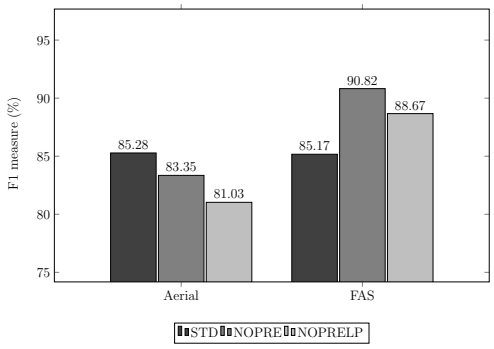
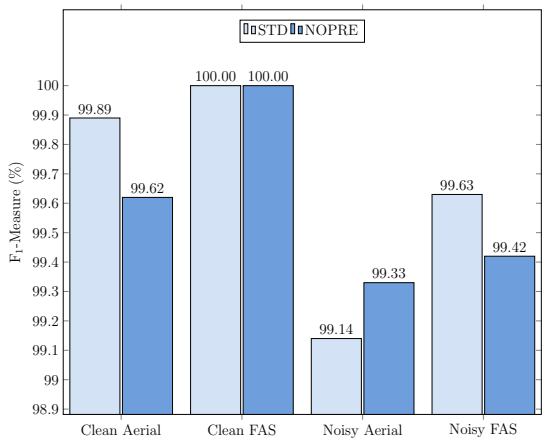
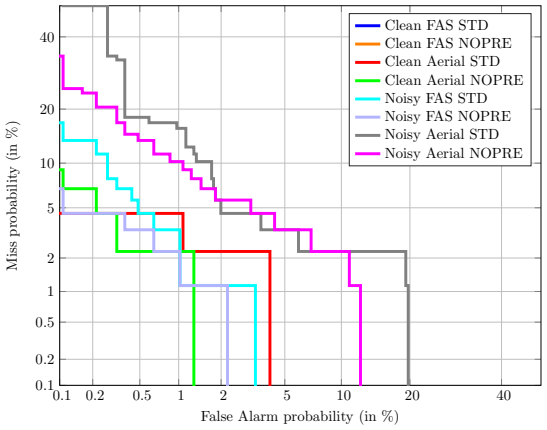


Figure 4.7: Fall classification performance in multicondition.

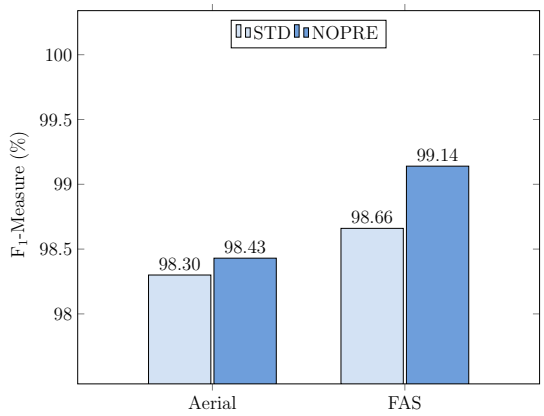


(a) F_1-Measure histogram plot.

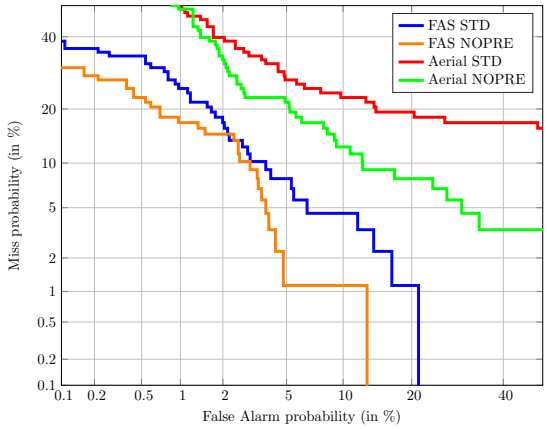


(b) Comparison of DET graphs.

Figure 4.8: Fall classification performance in matched condition with the dataset comprising everyday noises.

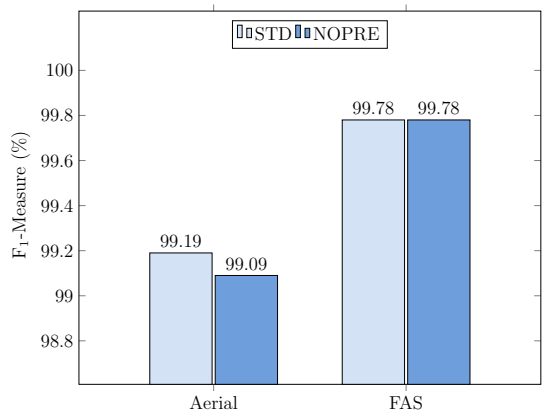


(a) F_1-Measure histogram plot.

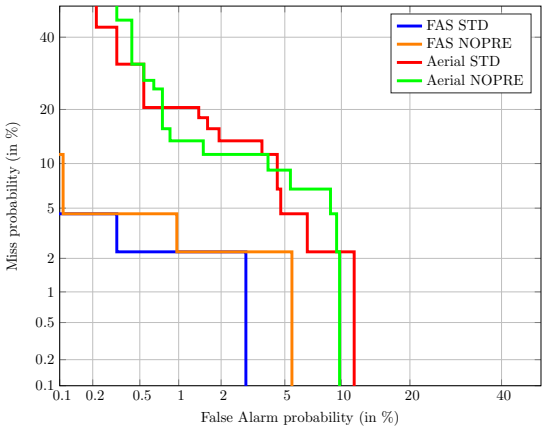


(b) Comparison of DET curves.

Figure 4.9: Fall classification performance in mismatched condition with the dataset comprising everyday noises.



(a) F_1-Measure histogram plot.



(b) Comparison of DET curves.

Figure 4.10: Fall classification performance in multicondition with the dataset comprising everyday noises.

Chapter 5

Unsupervised Approach

The problem with supervised approaches is that they require that each class of interest is well represented in the training dataset. However, in some case the variability of the environmental conditions or the subjects makes difficult or impossible to collect a sufficient number of examples that allow the algorithm to generalize well on unseen conditions [61]. This is the case of human falls. Unsupervised approaches tackle the problem as a novelty detection task [62, 63], i.e., by learning a normality model from data not related to human falls. In this chapter two approaches are proposed. The fist (Section 5.1) in based on One-Class SVM Section 2.1.1 trained with only background noises. The second Section 5.2 instead, approaches the problem from an end-to-end learning prospective by means of neural network autoencoder Section 2.3.3 for novelty detection.

Since the previous results have shown the superiority of the FAS with respect to the standard microphone, in the next sections only the samples recorded with the FAS has been used, unless otherwise specified.

5.1 Human fall detection algorithm based on One-Class Support Vector Machine

The approach proposed in this section is based on a One-Class Support Vector Machine (OCSVM) [39] to obtain an unsupervised framework for Fall Detection. The acoustic signals are captured by means the Floor Acoustic Sensor and then MFCCs and Gaussian Mean Supervectors (GMSs) are extracted by using the same methods described in Section 4.1.1: GMSs are higher level features computed by adapting the means of a Gaussian mixture model (GMM) with maximum a posteriori algorithm (MAP). In the training phase, a large set of audio data is used to model an Universal Background Model (UBM) composed of the GMM extracted by using Expectation Maximization (EM) algorithm [43]. Then, the GMS of each event is calculated by adapting the GMM with the MAP algorithm and concatenating the resulting GMM mean

Table 5.1: Composition of the dataset.

Class	Nr. of occurrences	Total length (s)
Basket	64	86
Fork	64	82
Ball	64	129
Book	64	63
Bag	64	57
Chair	96	157
Human Falls	44	76
Human Activity	665	1218
Music	776	1498

values. Abnormal acoustic events are discriminate from normal ones employing the OCSVM classifier.

5.1.1 Dataset

The performance of the algorithm has been evaluated on a corpus containing sounds of human falls, falling objects, human activities, and music. In particular, from the dataset presented in Section 3.2, have been used the samples reported in Table 5.1.

Musical tracks and normal activities sounds have been divided in segments whose lengths have mean and standard deviation estimated from instances of fall events. In addition, they have been employed alone and to create noisy versions of human and object falls occurrences in order to assess the algorithm in presence of interferences.

In the experiments, signals have been downsampled to 8 kHz and the resolution has been reduced to 16 bit. As in the approach presented previously, the choice of the sampling frequency is motivated by the analysis performed in a previous work by the authors Section 3.2.3, where it was shown that the signals recorded with the FAS have the majority of the energy concentrated at frequencies below 1 kHz.

5.1.2 Experimental setup

The dataset described previously has been divided in one set for training the UBM and the OCSVM and three sets for evaluating the performance.

Training has been performed on the set shown in Table 5.2 composed of 947 occurrences (1773 s) of human activities, classical music and rock music. The assessment of the algorithm has been performed on the following datasets:

Table 5.2: Composition of the training-set.

Class	**Nr. of occurrences**	**Total length (s)**
Human Activity	320	593
Music	627	1180
Total	947	1773

Table 5.3: Composition of "Set 1".

Class	**Nr. of occurrences**
Human Falls	44
Human Activity	15
Music	29

- Set 1 (Human fall and background sounds): this set comprises 44 examples of human fall sounds and 44 examples of human activity and music sounds (Table 5.3).
- Set 2 (Human fall and object fall sounds): this set comprises 44 examples of human fall sounds and 44 examples of object fall sounds (Table 5.4).
- Set 3 (Human fall, object fall and background sounds): this set comprises 44 examples of human fall sounds, 22 examples of background sounds and 22 examples of object fall sounds (Table 5.5).

For each set, the data have been divided in four folds, each composed of 11 human falls and 11 non-falls. Then, one fold has been used for estimating the hyperparameters of the algorithm and three for calculating the performance. The final performance is calculated by using the cumulative true positives, false positives, and false negatives obtained by varying the test folds. The validation phase consisted in searching for the number of components of the UBM, the values of ν and γ of the OCSVM. The values assumed by these variables are summarised in Table 5.6.

Comparative method The proposed approach has been compared to the algorithm presented in [64] based on OCSVM. The same algorithm has also been employed in [21] with a multi-microphone acquisition setup and a source separation stage. As in [64], the audio signals are divided in windows of the same lengths, and the related MFCCs are used for training the OCSVM and for classification. In [64], 7 MFCCs were extracted from audio signals sampled at

Table 5.4: Composition of "Set 2".

Class	**Nr. of occurrences**
Human Falls	44
Basket	7
Fork	7
Ball	8
Book	7
Bag	8
Chair	7

Table 5.5: Composition of "Set 3".

Class	**Nr. of occurrences**
Human Falls	44
Basket	3
Fork	4
Ball	4
Book	3
Bag	4
Chair	4
Human Activity	8
Music	14

20 kHz and the length of the window was set to 1 s. Here, the feature vectors are the same of the proposed approach, i.e., they are composed of the first 13 MFCCs and their first and second derivatives. The same window length of [64] cannot be employed here, since the dataset used in this paper comprises signals with lengths less than 1 s. Thus, the length of the window corresponds to the duration of the shortest event in the dataset, and it is equal to 576 ms (71 frames). Windows are overlapped by 50%, and, as in [64], an event is classified as fall if at least two consecutive frames are classified as novelty by the OCSVM. The same grid search procedure of the proposed approach has been adopted to search for the optimal values of ν and γ of the OCSVM.

The performance has been evaluated in terms of F_1-Measure calculated as:

$$F_1\text{-Measure} = \frac{2 \cdot tp}{2 \cdot tp + fn + fp}, \tag{5.1}$$

where tp is the number of correctly classified falls, fn is the number of falls

Table 5.6: Hyperparameters of the algorithm and search space explored in the validation phase.

Stage	Hyperparameter	Range
UBM	J	$1, 2, 4, \ldots, 64$
OCSVM	ν	$0.1, 02, \ldots, 1.0$
	γ	$2^{-15}, 2^{-13}, \ldots, 2^{3}$

misclassified as non-falls, and fp is the number of non-falls misclassified as falls.

5.1.3 Results

Figure 5.1 shows the results in clean conditions obtained with the proposed method named "OCSVM" and the comparative method proposed in [64] denoted as "Popescu (2009)". Observing the figure, it is evident that in all the three cases the OCSVM approach is able to improve the performance with respect to "Popescu (2009)" [64]. In particular, in "Set 1", that comprises human falls, human activities and music, the performance improves by 16.73% with respect to "Popescu (2009)". This case can be considered as the least challenging of the three, since non-falls events are considerably different from falls ones. Conversely, "Set 2" comprises both human falls and object falls, thus it includes abnormal events whose pattern is similar to the one of human falls. Indeed, the performance with respect to "Set 1" is 17.91% lower, mostly due the increased false positives rate that goes from 13.64% to 50.76%. Regarding "Popescu (2009)" [64], the F_1-Measure is below both OCSVM and the proposed approach, however it is less affected by the presence of object falls, since the F_1-Measure decreases only by 0.64% . "Set 3" comprises human falls, human activities, music and object falls and represents the most realistic test condition of the three. The results obtained by using the OCSVM classifier alone is 82.25%. As expected, this result is lower than "Set 1", since object falls are also present, and higher than "Set 2", since human activities and music segments are easier to discriminate. Differently, the approach by Popescu and Mahnot [64] degrades by 5.25% with respect to "Set 1", and by 4.61% with respect to "Set 2", demonstrating that it is less robust to the concurrent presence of object falls and daily human activities sounds.

Figure 5.2 shows the results obtained for the three cases in noisy conditions. As expected, the performance decreases in all the two evaluated methods. In "Set 1", the performance decrease is modest (2.32% for the OCSVM and 1.44% for "Popescu (2009)"), demonstrating that the OCSVM is able to effectively reject non-fall events corrupted by music interference. In "Set 2", the presence object falls corrupted by music significantly decreases the performance of the OCSVM, reducing the F_1-Measure by 12.74% with respect to the clean "Set 2".

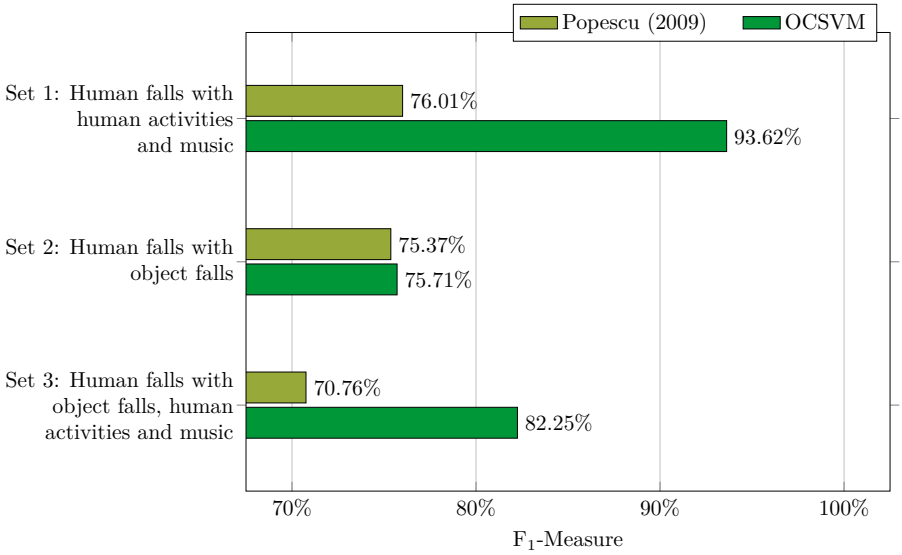


Figure 5.1: Results in *clean* conditions for the three test cases. "Set 1" comprises human falls, human activities and music. "Set 2" comprises human falls and object falls. "Set 3" comprises human falls, object falls, human activities, and music.

The method by Popescu and Mahnot [64] achieves the highest F_1-Measure in this case, confirming the good capabilities of rejecting dropping objects sound events observed in clean conditions. In "Set 3", the proposed approach improves the performance by 3.91% with respect to "Popescu (2009)", confirming that it is able to achieve the highest performance in the most realistic scenario of the three.

5.2 End-To-End Unsupervised Approach employing Convolutional Neural Network Autoencoders for Human Fall Detection

In recent years, thanks to the success of deep learning methods have become increasingly popular the feature learning approaches that independently transform the raw data inputs to a representation that can be exploited in machine learning tasks, minimizing the need of prior knowledge of application domain. Furthermore, such approaches are often able to generalize well real-world data compared to traditional hand-crafted features [65], resulting in an increase in performance of classification or regression tasks. The end-to-end learning is a particular example of feature learning, where the entire stack, connecting the

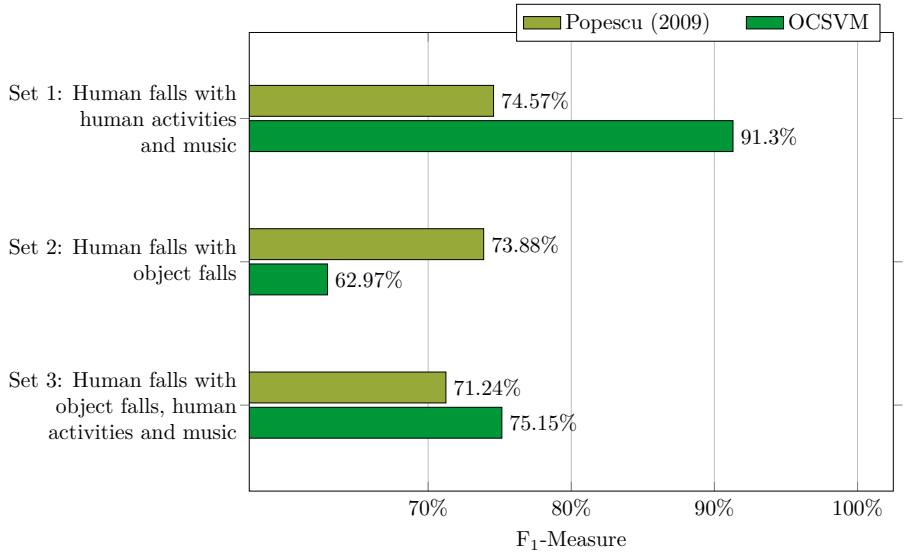


Figure 5.2: Results in noisy conditions for the three test cases. "Set 1" comprises human falls, human activities and music. "Set 2" comprises human falls and object falls. "Set 3" comprises human falls, object falls, human activities, and music.

input to the desired output, is learned from data [66]. As in feature learning, only the tuning of the model hyperparameters requires some expertise, but even that process can be automated [67].

In this section, an end-to-end acoustic fall detection approach is presented. A deep convolutional neural network autoencoder is trained with the signals, gathered by the Floor Acoustic Sensor (Section 3.1), corresponding to sounds that commonly occurring in a home (e.g., voices, footsteps, music, etc.). Since the sound produced by a human fall should be considerably different from the ones used for the training, it will be recognized as "novelty" by the network and classify as Fall. The performance of the algorithm has been evaluated on a subset of the dataset described in Section 3.2, which contains human fall events simulated by employing the "Rescue Randy" human mimicking doll [68, 18, 29] and sounds related to common human activities. Dieleman *et al.* [69] address the content-based music information retrieval tasks, investigating whether it is possible to apply end-to-end feature learning directly to raw audio signals instead that on a spectrogram representation of data. Up to the authors' knowledge, the end-to-end strategy has never been applied to a unsupervised fall detection approach with acoustic sensors. However, many works in other research fields can be founds in literature. Their convolutional neural networks trained on raw audio signals do not outperform a spectrogram-based approach

in the automatic tagging task but are able to autonomously discover frequency decompositions from raw audio, as well as phase and translation-invariant feature representations. The contribution is a novel fall detection method which exploits the end-to-end approach, resulting in a reduction of the engineering effort needed to design the system, with respect to those method that used features extracted from input signals. Is also shown that it can achieve good performance, higher than two state-of-the-art systems chosen for a comparison.

5.2.1 Proposed Approach

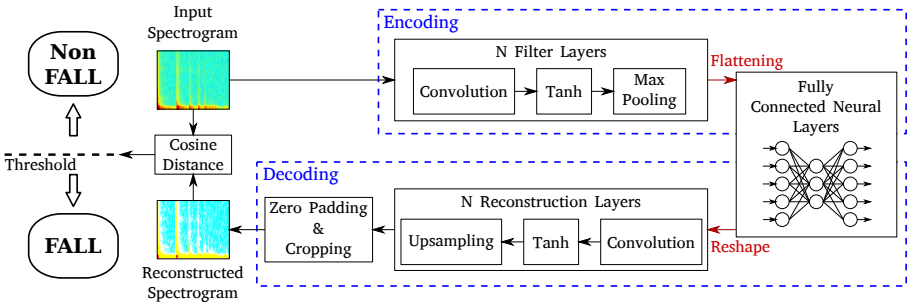


Figure 5.3: The proposed approach scheme.

In the state of the art the majority of fall detection systems are logically designed as a cascade of elements which perform some sub-task (e.g. feature extraction, modeling, classification, ecc.). Each task is independently developed and generally requires the tuning of a number of hyperparameters using an experimental procedure and some a prior knowledge about the domain of the problem.

The proposed approach, showed in Figure 5.3, is designed according to the end-to-end paradigm and then, the entire stack, connecting the input to the desired output, is learned from data. End-to-End is a feature learning strategy that, in presence of sufficient training data, may result in better performance than systems based on handcrafted features, since the training procedure automatically select the salient information. Therefore, if were possible to analyze the feature learned by the network with end-2-end strategy, there should be clues about what kind of information is important for a specific task.

The system core is a deep convolutional neural network autoencoder. Some exhaustive discussion about this type of network can be easily found in literature [70, 71, 72]. The network input consists of the normalized log-power spectrograms of the signals calculated with a STFT on windows of 32 ms and overlapped by 50%. Due to the presence of fully-connect neural layers, the input dimensions must be fixed. After having identified the widest spectro-

gram extracted from the dataset, the other ones have been extended with some AWGN frames added at the end. Each input consist in a $f \times t$ matrix, where f are the positive points of discrete Fourier transform and t are the number of windows considered in time. The output of the autoencoder are the reconstructed spectrograms. To classify an event, a distance measurement between input and output must be made with some heuristic. If the distance exceeds a certain threshold, automatically defined by the algorithm during the training phase, the system label the output as "Fall" or as "Non Fall" otherwise. In this work the cosine distance has been used:

$$D_C(v,u) = 1 - \frac{u \cdot v}{\|u\| \, \|v\|} = 1 - \frac{\sum_{k=1}^{n} u(k)v(k)}{\sqrt{\sum_{k=1}^{n} u(k)^2}\sqrt{\sum_{k=1}^{n} v(k)^2}} \tag{5.2}$$

where u and v are the the vectors obtained flattening the input and the output spectrograms and n are the length of this vectors. According to the cosine definition, the value of the distance always has a value between -1 and $+1$, where $+1$ indicates two equal vectors while -1 indicates two opposite vectors. The added AWGN part of the spectrums was not considered to calculate the distance. The choice of this heuristic allowed to make distance measurements independents of the size of the initial spectrum. The structure of the autoencoder is not defined a priori, but it is chosen through a phase of cross-validation during which the network parameters are varied with a random search strategy.

5.2.2 Experiments

In this section are described the composition of the dataset used in this work and the experimental set-up.

Dataset

The instances used in this approach are summarized in table Table 5.7. It is composed of two type of sounds: the first, namely novelty, comprises several human fall sounds that have been simulated by means of a human-mimicking doll employed in water rescues. The doll has been dropped from upright position and from a chair, both forward and backward. The drops have been then repeated at three distances from the FAS, i.e., 2, 4 and 6 m, for a total of 44 events, all included in the "Human fall" class. The second type, i.e, the background, comprises sounds of normal activities (voices, footsteps, etc.) for a total of 1218 s, and three musical tracks[1][2][3], for a total of 1498 s, played from a loudspeaker and acquired back with the FAS. In addition, the signals of the

[1] W. A. Mozart, "Piano trio in C major"
[2] Led Zeppelin, "Dazed and confused"
[3] Led Zeppelin, "When the levee breaks"

Table 5.7: Composition of the dataset.

Class	Nr. of occurrences	Total length (s)
Human Falls	44	76
Human Activity	665	1218
Music	776	1498

second type have been employed alone and to create noisy versions of human falls occurrences in order to assess the algorithm in presence of interferences. The signals have been acquired with a sampling rate equal to 44.1 kHz and 32 bit depth. Based on previous experience in using the FAS, in order to exploit the acoustic characteristics of the sensor, signals have been downsampled to 8 kHz and the resolution has been reduced to 16 bit.

Experimental Setup

Since in this work a novelty approach is presented, the dataset has been divided in two groups: the former composed only of background sounds (i.e human activity sounds and musical background) used for the training; the latter composed of both background sound and novelty sounds, i.e, the human falls, used in development and test phase. In order to assess the classification accuracy in noisy conditions, a second version of human fall sounds were crated in which a musical background was recorded and then digitally added to the fall events.

The input spectrograms of the audio signals has been calculated with a fft point number of 256 and a windows size of 256 samples (32 ms at sample rate of 8000 kHz). The longest spectrogram present in the dataset is composed of 197 frame. Therefore the resulting input matrix dimension $f \times t$ is 129×197. The optimization of the experiment hyper-parameters has been carried out using the random-search technique. Table 5.8 shows the parameters used in the random-search, and their ranges. The parameters of the network architecture are related only to the encoding part of autoencoder since the decoding part is its mirrored version. Instead other parameters, described below, have been set to the same value for all experiments. The activation function for each layer, whether they are convolutional or fully connected, have been set to *tanh*. "Adam" [73] has been used as optimization algorithm for the traing phase. The loss function used was *mlse*. The initialization algorithm for the weight of the autoencoder was Glorot Uniform [74]. The number of epoch has been set to 1000, while the patience, that is the number of epoch without an Auc improvement on a devset to wait before stopping the training phase, has been set to 40.

Table 5.8: Hyper-parameters optimized in the random-search phase, and their range.

Parameter	Range	Distribution	Parameter	Range	Distribution
Cnn layer Nr.	[1-3]	uniform	Batch size	[10%-25%]	log-uniform
Kernel shape	[3x3-8x8]	uniform	Max pool shape	[1x1-5x5]	uniform
Kernel Nr.	[4-64]	log-uniform	Max Pool	All[4]-Only end[5]	uniform
MLP layers Nr.	[1-3]	uniform	Dropout	[Yes-No]	uniform
MLP layers dim.	[128-4096]	log-unifom	Drop rate	[0.5-0.6]	normal
Stride	[1x1-3x3]	uniform	Learning rate	$[10^{-4}$-$10^{-2}]$	log-unifom

Table 5.9: Best hyper-parameters find in random-search phase for *clean* and *noisy* condition

	Clean				Noisy			
Parameter	Fold1	Fold2	Fold3	Fold4	Fold1	Fold2	Fold3	Fold4
Cnn layer Nr.	3	3	3	2	3	3	3	3
Kernel shape	8x8	7x7	5x5	8x8	8x8	7x7	8x8	8x8
Kernel Nr.	[16,16,8]	[32,16,16]	[8,8,8]	[32,16]	[32,32,8]	[32,32,8]	[32,32,32]	[8,8,8]
Max Pool Position	only end	only end	all	all	only end	only end	all	only end
Max pool shape	5x5	3x3	5x5	4x4	3x3	5x5	5x5	3x3
Stride	3x3	3x3	1x1	3x3	3x3	3x3	1x1	3x3
MLP layers Nr.	2	1	1	1	2	2	1	3
MLP layers dim.	[16,231]	96	32	32	[48,153]	[16,2084]	128	[48,1952,1952]
Learning rate ($\times 10^{-4}$)	4.89	4.08	15.09	15.44	1.56	4.46	1.01	1.00
Batch size	11.26%	10.81%	13.59%	21.10%	20.06%	12.55%	13.51%	13.13%
Drop rate	0.64	0.57	0.53	0.55	0.58	0.53	0.55	0.59

In order to implements a 4 fold cross-validation, the signals not being part of training-set have been divided in four folds, each composed of 11 human falls and 11 non-falls signals. Then, one fold has been used as validation-set and the remaining three for calculating the performance in test phase. In cross-validation phase the scores have been evaluated in term of AUC. Here also the optimal thresholds have been infer by searching points on ROC curves closest to the $(0, 1)$: $d_{min} = \sqrt{(1 - fpr)^2 + (1 - tpr)^2}$. At the end the final performance has been evaluated in term of $F_1 - Measure$ by mediating the results obtained on individual folds.

Since this approach share the same train-set (Table 5.2) and test-set (Section 5.3) that has been used in Section 5.1 for both the clean and noisy conditions, the results scored here were compared with those obtained showed in Figure 5.1 and related to Set 1.

5.2.3 Results

The results for both clean and noisy are reported in Figure 5.4. The comparative algorithms are denoted with "Popescu (2009)" and "OCSVM" respectively, while the proposed approach is named "Autoencoder". It is immediately clear that the proposed approach outperforms the other in both conditions. In fact, in clean condition, it gains about 1% compared to "OCSVM" and about 18.6%

[4] After each Conv. layer
[5] At the end of cnn part

compared to "Popescu (2009)". Moreover the proposed algorithm results to be very robust in noisy condition, whereas both other cases get worse a bit. In particular the performance improves by 3.72% with respect to "OCSVM" and by 20.45% with respect to "Popescu (2009)". Clearly the end-to-end method seems insensitive with respect to the corrupted human fall signals when the novelty sounds are dissimilar respect to normality model learned from the background sounds. In Table 5.9 are reported the hyperparameters that have led to the results discussed above.

Furthermore other experiments were made up with a manual tuning of parameters. Particularly have been investigated deeper architectures composed up to 5 convolutional layer. We found that increasing the depth on cnn part (5 layers) with a different kernel number for each layers of [32,32,16,16,8] and a kernel dimension for all layers of 4x4, a max pooling after only the first three convolutional layer of 2x2 and two MLP layer of 1024 and 512, leads to considerable improvements. In effect the final $F_1 - Measure$ rise up to 95,42% in both clean and noisy condition.

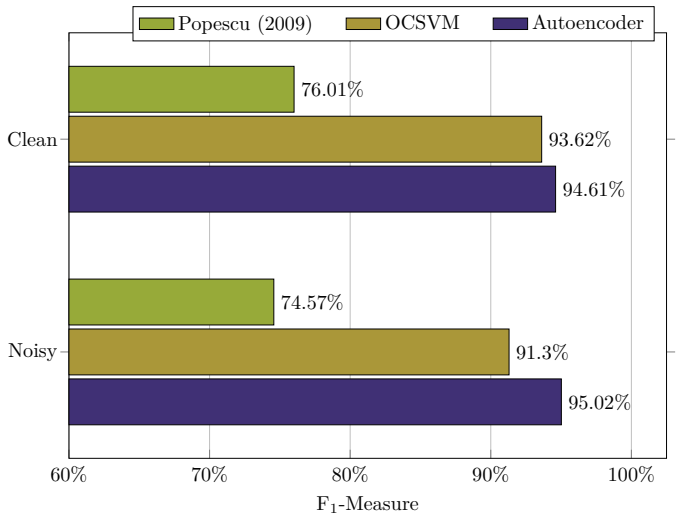


Figure 5.4: Results in *clean* and *noisy* conditions for the three test cases.

5.2.4 Remarks

The methods presented is an end-to-end approach composed by a deep-convolutional-autoencoder with a downstream threshold classifier, that is a purely unsupervised approach to acoustic fall detection. Our method exploits the reconstruction error of the autoencoder. When a sound that the network has never seen in training phase occurs, the reconstruction error raise up allowing the recognition

of novelty. The algorithm has been trained with a large corpus of background signals, i.e, human activity noise and music, and evaluated with human-fall sound and other instances of background sounds. It has been evaluated in two different condition: the first with a clean version of human fall sounds and the second with corrupted version of the same. The results showed that the proposed solution leads to an average improvement about 20% with respect to the [64] and about 2.3% towards the OCSVM based approach.

Chapter 6

Weakly-supervised Approach

Unsupervised methods, consider a human fall as an event that deviate from normality and they are based on one-class classifiers. The main advantage of an unsupervised methods for fall detection is that it can works without knowing any examples related the the class of interest, i.e. the human fall. As aforementioned, this should be the perfect way to go in an application like this, where what we are interested in is very difficult to retrieve or we only have very few data available, but the principal weakness of an unsupervised system is that certain events deviate from normality as the human fall (e.g., the fall of an object), thus they may produce false alarms. Moreover, in some applications, like the fall detection task, it can be difficult to obtain strong supervision information due to the high cost of the data-labeling process. Thus, it is desirable for machine-learning techniques to be able to work with weak supervision. The weak-supervision is a generic term that include different kind of techniques [75]:

- "incomplete supervision" where the labels are available only for a small subset of training;
- "inexact supervision" where only coarse labels with respect to the samples are provided
- "inaccurate supervision" where the labels are not always correct.

In this chapter, methods belonging to the first an second category are described: in Section 6.1 a semi-supervised OCSVM method, that is a subcategory of the "incomplete supervision" problem [75], is exposed in which the user provides some additional labels to the system.. Instead, in Section 6.2 present an approach based on Siamese neural network for one-shot learning in a "inexact supervision" context, where only information on the difference in belonging to the same fall class is provided. Both of them, has been assessed with samples related to R0 room of the employed dataset Section 3.2. An extension of the Siamese approach is presented in Section 6.2 where the entire dataset described in Section 3.2 has been used, proving the effectiveness of the Siamese approach in a complex scenario.

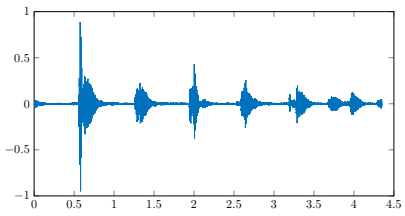
6.1 A Combined One-Class SVM and Template Matching Approach for User-Aided Human Fall Detection

The approach proposed here, is the extension of the one presented in Section 5.1 thus, consists of a combined One-Class Support Vector Machine (OCSVM) based method and template-matching classifier that operate in cascade. The template-matching classifier operates in a user-aided supervised manner and it is employed to reduce such errors by using a set of templates that represent these events. Templates are identified by the user that marks the occurrence of a false positive instead of a true human fall event. As shown in the previous section, "unsupervised methods" are able to overcome the need of manual tuning of "analytical methods" and the necessity of a large labelled dataset of "supervised methods". In "unsupervised methods", falls are discriminated from non-falls based on a model of "normality" constructed from a large amount of non-fall events. However, certain events differ from the "normality" as human falls, and they may induce the classifier to produce false alarms. As an example, Figure 6.1a and Figure 6.1b show respectively the waveform and the spectrogram of a segment of "normal" human activity (footsteps and speech) Figure 6.1c and Figure 6.1d show the waveform and the spectrogram of a segment of human fall, and Figure 6.1e and Figure 6.1f the waveform and the spectrogram of a book fall. The figures show clearly that both falls signals differ significantly from the human activity one, thus a classifier may be induced to consider the fall of a book as the fall of a person.

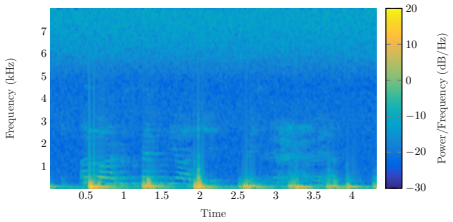
The algorithm proposed in this work reduces the problem by employing a multi-stage classification approach that combines a one-class classifier based on OCSVM with a template-matching stage. The OCSVM is trained unsupervisedly on a large corpus containing sounds that represent the "normality". On the contrary, the template-matching stage employs a set of templates represented by a small number of feature vectors marked as false alarm by the user. Thus, robustness against possible false alarms is achieved by using only few examples of false positive classes without the need of multiple sensors. An additional advantage with respect to the state of the art is that the proposed approach is able to evolve and improve after its initial training, since the template set can be augmented as non-falls events are detected.

6.1.1 Proposed approach

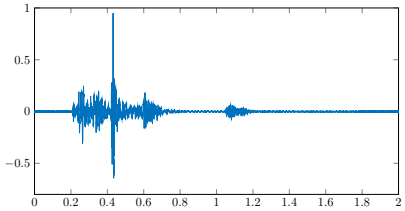
The proposed approach is composed of three stages Figure 6.2: the first ("Feature Extraction") extracts MFCCs from the input audio signal and then GMSs to describe the entire audio segment. The second stage ("Abnormal Event



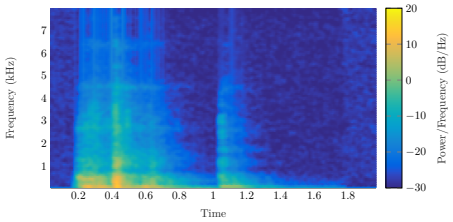
(a) Normal human activity signal in the time domain.



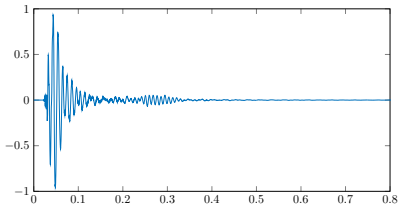
(b) Normal human activity signal in the frequency domain.



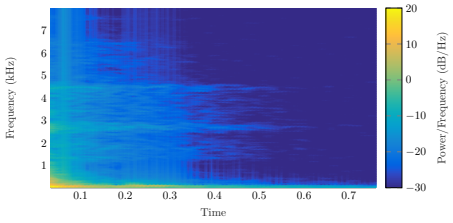
(c) Human fall signal in the time domain.



(d) Human fall signal in the frequency domain.



(e) Book fall signal in the time domain.



(f) Book fall signal in the frequency domain.

Figure 6.1: Time domain (on the left) and frequency domain (on the right) representation of a normal human activity signal (a-b), human fall signal (c-d), and book fall signal (e-f).

Detection") consists of a One-Class SVM classifier that discriminates between normal and abnormal sounds. Up to the authors' knowledge, OCSVM together with GMSs have never been jointly used for acoustic fall detection. The third stage represents the innovative contribution of this work for reducing false alarms in unsupervised approaches: it consists of a "Template-Matching" block that refines the output of the OCSVM and classifies the input data as fall or non-fall. The OCSVM is trained unsupervisedly on a large dataset of everyday sounds with the objective of discriminating normal from abnormal sounds. As aforementioned, the basic assumption is that the acoustic events related to human falls are "rare" respect to sounds normally occurring inside a home. The template-matching stage, on the other side, requires a set of "template" instances that represent rare events that can be confused with a fall. Referring to Figure 6.2, the "Template-Matching" stage is composed of a set of "Templates", a block that calculates the distance between the input GMS and the templates ("Euclidean Distance Calculation"), and a "Decision" block the decides whether the event is a fall or a non-fall by evaluating the magnitude of the distance. The rationale here is that certain acoustic events are as abnormal as falls and confuse the OCSVM: the template-matching stage reduces false positives by using a set of examples related to the most confusing classes. In this work, the algorithm is "user-aided", i.e., templates are indicated by the user each time the OCSVM produces a false positive. This is shown in Figure 6.2 with the person silhouette near the block that decides whether a detected fall is a false positive or not ("False Positive?"). In general, however, it is possible to create the templates set a-priori by recording several instances of possible false alarms events. Although rare, false alarm events (e.g., falls of objects) are certainly easier to reproduce in laboratory respect to human falls.

Template Matching

The template-matching classifier operates on a set of templates, i.e., supervectors, that can be defined a-priori or selected by the user when the OCSVM detects an abnormal sound that is not a human fall. Denoting with $\mathbf{x}$ the supervector of the input signal and with $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ the set of templates, the algorithm operates by calculating the Euclidean distance $D^{(i)} = \|\mathbf{x} - \mathbf{y}_i\|$ between the supervector to be classified and all the templates in the set. Indicating with $D_{min} = \min_i D^{(i)}$, the supervector $\mathbf{x}$ is classified as a fall if $D_{min} > \beta$ and as non-fall otherwise. The threshold β is a hyperparameter of the algorithm that can be determined on a validation set.

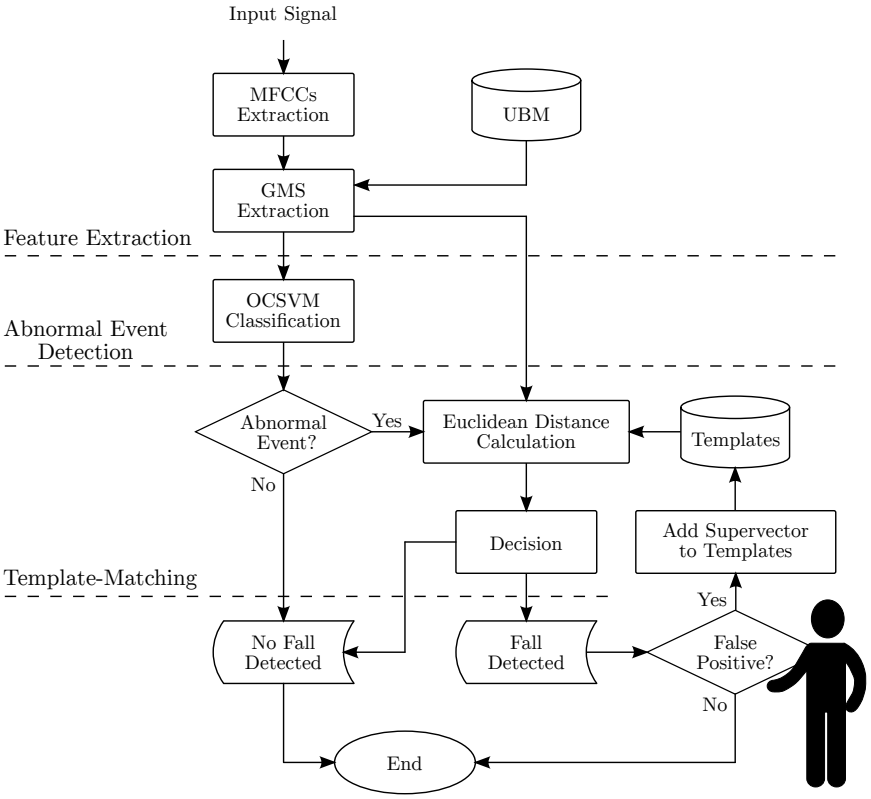


Figure 6.2: The block scheme of the proposed approach.

6.1.2 Dataset

The dataset employed in this method is the same used for the approach proposed in Section 5.1 and reported in Table 5.1. Please refer to Section 5.1.1 for the details.

6.1.3 Experimental setup

As in the system presented in Section 5.1, the dataset has been divided in one set for training the UBM and the OCSVM and three sets for evaluating the performance. Training has been performed on the same set used for the OCSVM base algorithm presented in Section 5.1 and shown in Table 5.2. The same three sets described in Section 5.1.2 has been used for the assessment and summarized below as a reminder:

- Set 1: Human fall and background sounds (Table 6.2a).
- Set 2: Human fall and object fall sounds (Table 6.4a).

- Set 3: Human fall, object fall and background sounds (Table 6.6a).

Table 6.1: Data used in "Set 1".

(a) Composition of "Set 1".

Class	**Nr. of occurrences**
Human Falls	44
Human Activity	15
Music	29

(b) Templates of "Set 1".

Class	**Nr. of templates**	
	Clean	**Noisy**
Human Activity	13	11
Music	8	16
Total	21	27

Table 6.3: Data used in "Set 2".

(a) Composition of "Set 2".

Class	**Nr. of occurrences**
Human Falls	44
Basket	7
Fork	7
Ball	8
Book	7
Bag	8
Chair	7

(b) Templates of "Set 2".

Class	**Nr. of templates**	
	Clean	**Noisy**
Basket	55	57
Fork	39	55
Ball	11	52
Book	26	57
Bag	26	56
Chair	86	89
Total	243	366

The validation phase has been set following the same procedure described in section Section 5.1.2: a cross-validation composed of four fold has been used for estimating the hyperparameter and the final performance is calculated by using the cumulative true positives, false positives, and false negatives obtained by varying the test folds. Differently from the previous method, the validation phase consisted not only in searching for the number of components of the UBM and the parameters (ν and γ) of the OCSVM, but also the value of the threshold β in the template-matching stage. The values assumed by these variables are summarized in Table 6.7. The method employed for the template-matching decision threshold is explained in Section 6.1.4.

All the aforementioned datasets require a set of templates for the template-matching stage of the algorithm. In the case of object falls, the set of templates has been created by classifying a set of 372 object falls with the OCSVM and selecting the occurrences misclassified as human falls. In the case of background sounds, the set of templates has been created by calculating the Euclidean distance between each occurrence of the development-set and each occurrence of a set of 470 background signals and then selecting the segment whose distance

Table 6.5: Data used in "Set 3".

(a) Composition of "Set 3".

Class	Nr. of occurrences
Human Falls	44
Basket	3
Fork	4
Ball	4
Book	3
Bag	4
Chair	4
Human Activity	8
Music	14

(b) Templates of "Set 3".

Class	Nr. of templates	
	Clean	Noisy
Basket	52	57
Fork	57	57
Ball	19	55
Book	53	57
Bag	50	56
Chair	89	89
Human Activity	11	4
Music	4	11
Total	335	386

Table 6.7: Hyperparameters of the algorithm and search space explored in the validation phase. The search space of the template-matching threshold β is not reported, since is determined with the procedure described in Section 6.1.4.

Stage	Hyperparameter	Range
UBM	J	$1, 2, 4, \ldots, 64$
OCSVM	ν	$0.1, 02, \ldots, 1.0$
	γ	$2^{-15}, 2^{-13}, \ldots, 2^{3}$
Template-matching	β	See Section 6.1.4

is minimum. Details on the templates sets are shown in Table 6.2b, Table 6.4b, and Table 6.6b respectively for "Set 1", "Set 2", and "Set 3".

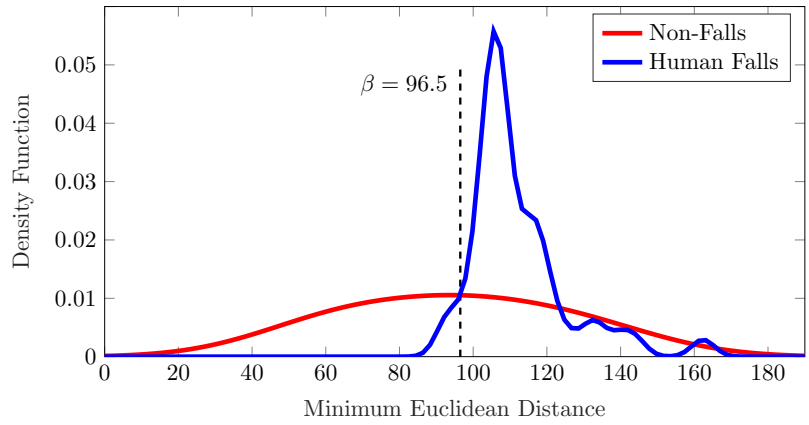
The proposed approach has been compared with the method from which it derives (Section 5.1) and the algorithm presented in [64] based on OCSVM (please revert to Paragraph 5.1.2 for the details)

The performance has been evaluated in terms of F_1-Measure (5.1)

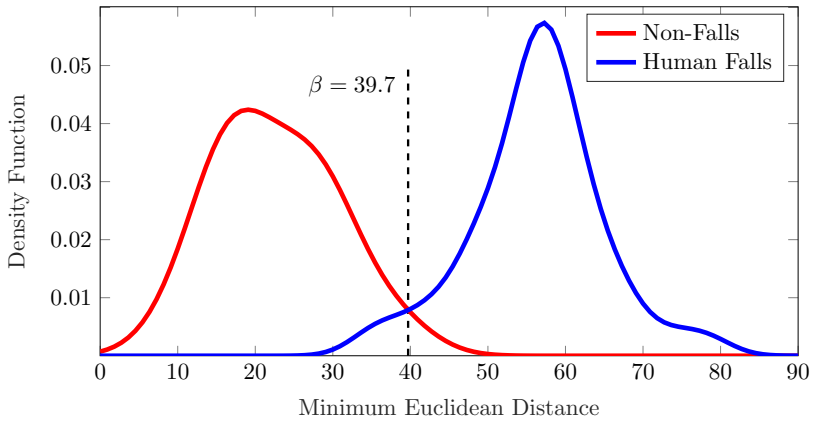
6.1.4 Choice of the template-matching decision threshold

A key point of the proposed approach is the decision threshold β in the template-matching stage. Choosing a too low value would result in a low number of false negatives and a high number of false positives. On the contrary, a too high value would result in a high number of false negatives and a low number of false positives. The choice of β has been performed by calculating the minimum Euclidean distance between each fall and non-fall event in the validation set and the set of templates. Figure 6.3 and Figure 6.4 show respectively the probability distributions for the three sets in clean and noisy conditions. The

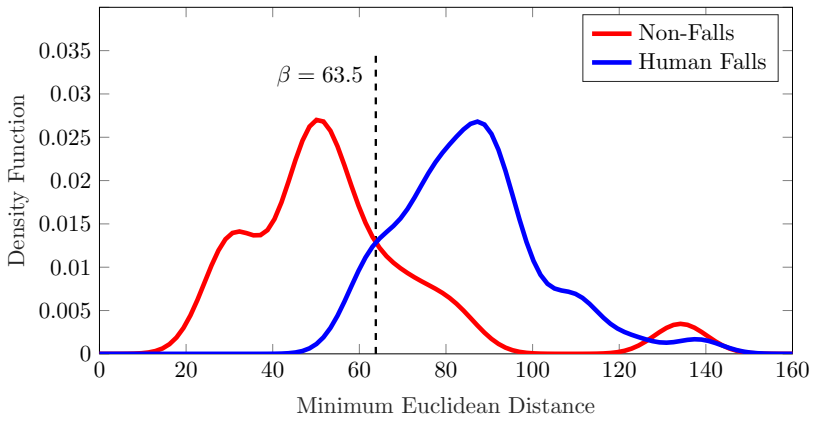
decision threshold β has been chosen at the intersection point between the distribution of fall and non-fall distances. This choice represents a compromise that balances false positives and false negatives. Observing clean condition distributions, in "Set 1" the two density are considerably overlapped, while in "Set 2" the overlap is modest. It is expected that the possible improvement of the template-matching stage will be more consistent for "Set 2" respect to "Set 1". "Set 3" contains human activity and music occurrences as "Set 1" and object falls as "Set 2": indeed, the probability distributions (Figure 6.3c) are more distinct respect to the ones of "Set 1", but not so much as the ones of "Set 2". Noisy condition distributions, shown in Figure 6.4, are in general less distinct compared to clean condition ones. The effect of noisy is to flatten the distances of the fall and non-fall classes, thus resulting in a less discriminative capabilities of the classifier. Thus, it is expected that the performance improvement in noisy conditions will be more modest respect to the one obtained in clean condition.



(a) Probability distributions related to "Set 1".

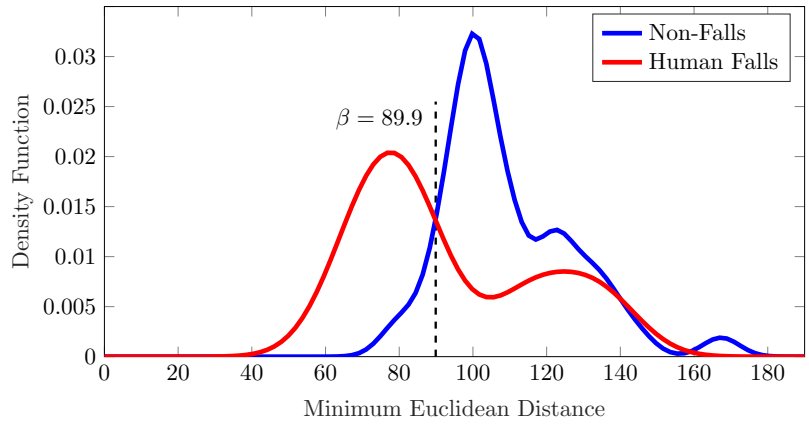


(b) Probability distributions related to "Set 2".

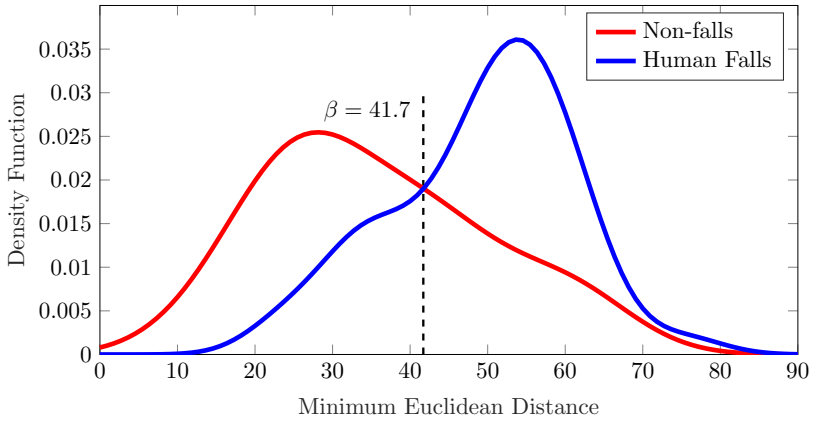


(c) Probability distributions related to "Set 3".

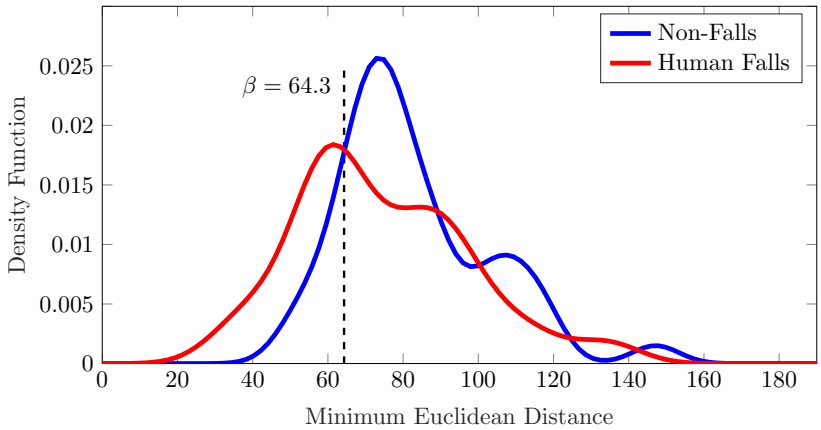
Figure 6.3: Probability distributions of the minimum Euclidean distances among the template sets, and human falls and non-falls in *clean* acoustic condition.



(a) Probability distributions related to "Set 1".



(b) Probability distributions related to "Set 2".



(c) Probability distributions related to "Set 3".

Figure 6.4: Probability distributions of the minimum Euclidean distances among the template sets, and human falls and non-falls in *noisy* acoustic condition.

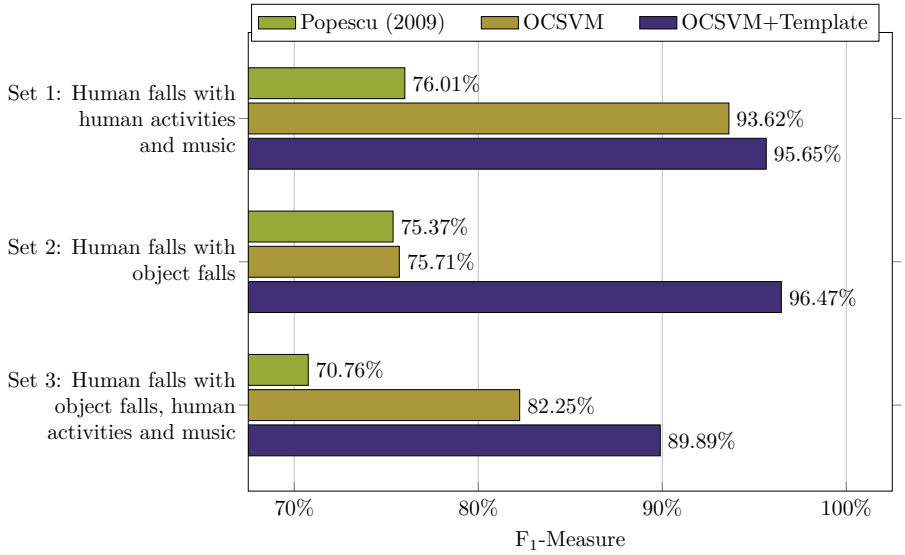


Figure 6.5: Results in *clean* conditions for the three test cases. "Set 1" comprises human falls, human activities and music. "Set 2" comprises human falls and object falls. "Set 3" comprises human falls, object falls, human activities, and music.

6.1.5 Results

Figure 6.5 shows the results in clean conditions obtained with and without the template-matching stage, respectively denoted as "OCSVM+Template-Matching" and "OCSVM". The results obtained with the method proposed in [64] are denoted with "Popescu (2009)". Observing the figure, it is evident that in all the three cases the template-matching approach is able to improve the performance with respect to "Popescu (2009)" [64] and the OCSVM only approach. In particular, in "Set 1", that comprises human falls, human activities and music, the performance improves by 2.03% with respect to OCSVM and by 19.64% with respect to "Popescu (2009)". This case can be considered as the least challenging of the three, since non-falls events are considerably different from falls ones. Conversely, "Set 2" comprises both human falls and object falls, thus it includes abnormal events whose pattern is similar to the one of human falls. The introduction of the template-matching stage considerably reduces the number of false positives, leading to an overall performance improvement of 20.76%. "Set 3" comprises human falls, human activities, music and object falls and represents the most realistic test condition of the three. Introducing the template-matching stage, the performance improves by 7.64%, leading to an F_1-Measure equal to 89.89%.

Figure 6.6 shows the results obtained for the three cases in noisy conditions.

As expected, the performance decreases in all the three evaluated methods. In "Set 1", the performance decrease is modest (2.32% for the OCSVM, 2.63% for the proposed approach, and 1.44% for "Popescu (2009)"), demonstrating that the OCSVM is able to effectively reject non-fall events corrupted by music interference. The use of the template-matching stage increases the performance by 1.72%, thus providing a significant improvement also in noisy conditions. In "Set 2", Template-matching provides a performance improvement of 8.02% with respect to the OCSVM, leading to an F_1-Measure higher than 70%. The improvement is lower with respect to the clean "Set 2", since the variability of the music interference makes the Euclidean distances of fall and non-fall classes more similar and is not sufficient to overcome the "Popescu (2009)" [64]. In "Set 3", the proposed approach improves the performance by 4.77% with respect to OCSVM and by 8.68% with respect to "Popescu (2009)". In summary, the results demonstrated that the introduction of a template-matching stage significantly improves the performance both of the OCSVM only approach and of the method by Popescu and Mahnot [64]: averaging the results over "Set 1", "Set 2", and "Set 3", the absolute improvement with respect to the former is 10.14% in clean conditions and 4.84% in noisy conditions. With respect to the latter [64] the improvement is 19.96% in clean conditions and 8.08% in noisy conditions. As shown in Figure 6.5 and Figure 6.6, both in clean and noisy conditions the F_1-Measure of the method by Popescu and Mahnot [64] is close to 75% in "Set 1" and "Set 2", and close to 71% in "Set 3". The different behaviour compared to the OCSVM only approach can be attributed firstly to the different feature representation of the audio signal (MFCCs instead of supervectors). Secondly, to the strategy adopted for classification: in [64], signals are divided in windows and a fall is detected if at least two consecutive windows are classified as fall. Differently, in the proposed algorithm, the overall signal is represented by a single supervector and classified as fall or non fall. Comparing the results in clean (Figure 6.5) and noisy (Figure 6.6) conditions, it is evident that techniques for reducing the impact of additive noise are needed. Additionally, the proposed solution requires the intervention of the user for selecting the templates after the first classification stage performed by the OCSVM. This aspect will be addressed in next sections in order to make the algorithm completely independent of the user and using only few examples related to human fall.

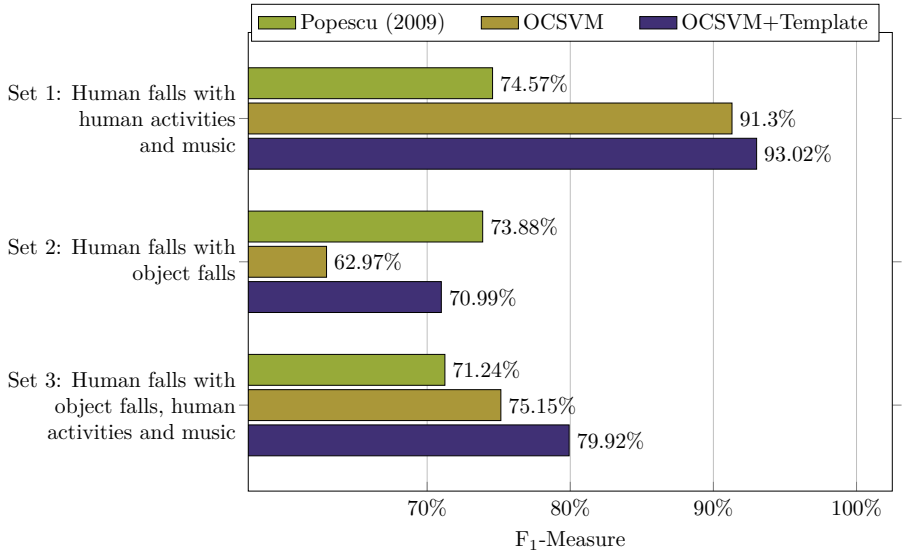


Figure 6.6: Results in noisy conditions for the three test cases. "Set 1" comprises human falls, human activities and music. "Set 2" comprises human falls and object falls. "Set 3" comprises human falls, object falls, human activities, and music.

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

As already mentioned, the main challenge for falls detector systems is the lack of human falls examples in the dataset, that are rare and hardly recoverable. Hence the need to develop systems that are able to work with no or few such data. In fact, Khan et al. [10] analyze Fall Detection techniques and divides them based on data availability prospective in two groups. The group is composed of algorithms that can draw form dataset with sufficient number of human falls that can be used in training. Mostly based on supervised machine learning, thresholding and one-class techniques, this methods attempt to detect a fall directly given their training data. As we show here, although these supervised techniques can achieve a high reliability, they needs a large labelled dataset including many human fall, that is not easy to retrieve for this specific application. Moreover, applying this techniques when there are no training data for human falls leads to an unacceptable miss rate. As a comparative model for this category, has been evaluated an algorithm based on SVM.. The second group include algorithms that can draw form dataset with insufficient or no training data for falls mostly based on over/under-sampling, semi-supervised learning, novelty detection and one-class classification. Those

techniques differ from the previous ones because uses the available data as a description of normal activities. Once it is trained on these data, can be able to classify a test sample as a human fall or non-human fall. The major drawback here is that they need a good description of what "normal activities" are. In fact to give a good representation of this concept, a big data set comprising all the normal activities is needed [76]. In real scenario, this is very difficult to obtain and this may induce the classifier to produce a high number of false alarms. As a comparative models for this category, 2 approaches have been evaluated: a OCSVM that is a totally unsupervised method and a extremely unbalanced SVM that use only one human fall for the training phase. The proposed algorithm belongs to the second category, and as a first step, we demonstrate that the SNN can achieve better results than both unsupervised and supervised methods when they are tested under similar conditions.

One-shot or few-shot methods have been recently revived in other fields of application. The Siamese approach was introduced by Bromley et al. [77] for signature verification and later also used in [78] for face verification, both of them in a supervised framework. Regarding the one-shot learning approach, the Siamese framework was first employed by Koch et al.[79] for image recognition. In [80] an attention mechanism over a learned metrics is used. In that work, the authors propose so-called Matching Networks trained by showing only a few examples per class for each minibatch in order to mimic the few-shot task by subsampling classes in a meta-learning perspective. In the audio field, one-shot approaches have been rarely used up to now. Lake et al. [81] proposed a hierarchical Bayesian acoustic-based approach to model the way a person learns a word of a new language from a few examples. They use a Hierarchical Hidden Markov model that induces the set of phone-like acoustic units directly from the raw unsegmented speech data in a completely unsupervised manner, identifying segments that should be clustered together and learning a set of phone-like acoustic units for the language. Manocha et al.[82] proposed a method based on Siamese networks for audio Content-based Representations.

6.3 Proposed Approach

In this work the authors 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 work euclidean distance was used. In Figure 6.7, are reported two example 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.

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:

$$E_w = \|S_e(X_1) - S_e(X_2)\|. \tag{6.1}$$

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 [78]:

$$Loss = (1 - Y)\frac{1}{2}(E_w)^2 + (Y)\frac{1}{2}\{(max(0, m - E_w)\}^2. \tag{6.2}$$

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. In this way the system should be able to learn embedded features allowing classification even of the unseen rare sound event such as human fall. 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 as explained in section 6.4. 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.

6.3.1 Feature Extraction

The example provided to the system are preprocessed in a features extraction stage that computes the log mel-energies. These features have been chosen as they are very popular for computational audio analysis [83, 84, 85]. In particular, for this work the log mel-energies have been computed as follows. First the signals were padded with some AWGN samples to the length of the longest audio event present in the dataset in order to work with the designed neural network. Then the audio has been downsampled to 16 kHz and normalized. After that signal has been segmented in frames 40 ms long overlapped by 20 ms

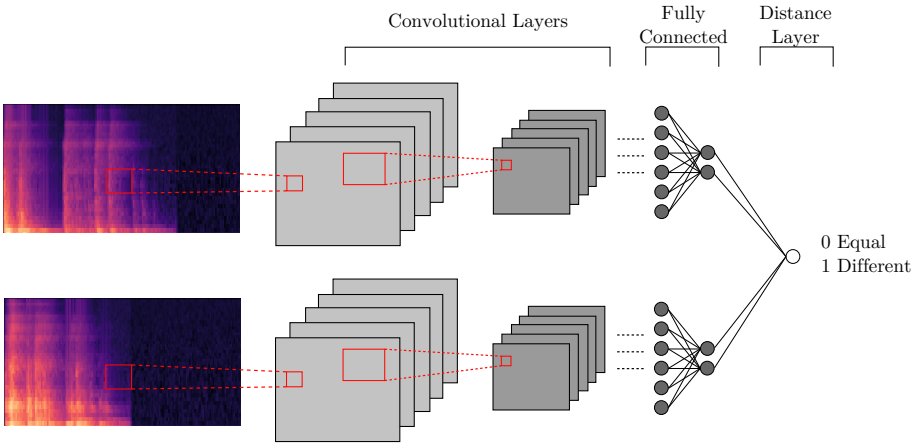


Figure 6.7: Proposed approach.

and multiplied with a Hamming window. Discrete Fourier Transform (DFT) is calculated and the absolute value is filtered with a filterbank composed of 40 triangular filters uniformly spaced on the mel scale. At the end each audio event is represented as a matrix X of shape 40×159.

6.3.2 Network Architecture

As shown in Figure 6.7, the architecture explored here for the twin networks consists of L_c convolutional layers followed by a max pooling. After the convolutional part, there are L_f fully connected layers. After each convolutional and fully-connected layers a *batch normalization* on features map is applied followed by a *Leaky ReLU* activation function. See Section 6.5 for details about the best performing network.

6.3.3 Dataset

All the instances related to the fall events of the R0 room (Section 3.2) has been used in this work. In order to achieve a data augmentation, the instances recorded with all the microphones used during the creation of the dataset has been used:

- one microphone array composed of 3 microphones (here taken as single microphones);
- 2 prototype of Floor Acoustic Sensor: the first, from now on indicated with FAS, was widely described in Section 3.1. For the second the only difference is the microphone used inside[1] that is characterized by an omni-

[1]RODE Lavalier: http://en.rode.com/microphones/lavalier.

directional directivity pattern instead of hyper-cardioid pattern.

Table 6.8 shows the number of instances for each class used in this work considering only audio recorded with FAS.

Table 6.8: Composition of the dataset.

Class	**Nr. of occurrences**	**Total length (s)**
Basket	64	86
Fork	64	82
Ball	64	129
Book	64	63
Bag	64	57
Chair	96	157
Human Falls	44	76

6.4 Experiments

Two methods previously described have been taken as a baseline for the comparison: in Section 4.2 a supervised approach based on bi-class SVM able to discriminate fall event from non-fall event was employed. In this works the training set was composed of labeled data representing fall and non-fall event. In Section 5.1 instead, an unsupervised method based on OCSVM was presented. In this case the training set was composed only from background that comprises human daily activities sounds, classical and rock music played from a loudspeaker. . For a correct comparison baseline, experiments have been repeated using the same data employed here for the SNN as described below Section 6.4.

Data Selection

All the experiments have been conducted with three-way data split and a 10 folds cross-validation strategy. In particular, for each fold, the data has been divided in 90% for indicated, 5% for validation and 5% for the test, so in each set, all the classes indicated in the Table 6.8, are present in a balanced way. Since here we approach the problem as a binary classification, this has led, of course, to an imbalance between the human fall and the rest. Because each evaluated approach differs from the other, it was necessary to select the data for each of them. To keep the various experiments comparable to each other, starting from the set above, we have constructed the various data sets for each approach as follows:

- SNN: first all the signals corresponding to the human fall class have been removed from the training set. Inasmuch as the Siamese network works with paired inputs we generated all the combinations, without repetitions, between all the training examples and then randomly 80000 pairs were selected. In prediction phase, the Siamese Network needs a template to classify the events, so the development set has been subdivided in sub-test: in each of these a human fall present in the validation set has been paired with all the other events present in the same set. In the test, instead, a single human fall was selected randomly to be used as a template for each fold. This was done to keep the test set comparable between the different methods;
- SVM: no changes have been made to the lists;
- SVM-unbalanced: in the training set only a human fall has been left;
- OCSVM: as this is an unsupervised method all the signals corresponding to the human fall class have been removed from the training set.

For each method, the development and test set have been used only the FAS signals while the training set has been augmented with the instances of the events recorded with all others the microphones.

Validation and Evaluation

The performances of the compared algorithm have been evaluated in term of $F_1 - Measure$ referred to the human fall class. Due to the unbalanced nature of development and test set, the metric has been computed starting from a normalized confusion matrix. In particular, for the final evaluation, all the absolute confusion matrices coming from each fold have been summed. The cumulative confusion matrix has been then normalized and the final $F_1 - Measure$ was computed from it.

To optimize the hyper-parameters of the methods the following strategies have been adopted:

- for the SNN a small random search of 30 network configurations has been performed. The hyper-parameters varied during the search are reported in Table 6.9. The search of the threshold value of the output layer of the network which yielded the best $F_1 - Measure$ has been performed on the validation set. The threshold found in this way was then used for the test of the same fold;
- for the SVM methods a grid search strategy has been adopted to optimize the parameters. In particular the parameters have assumed values in the

ranges $\{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ for C (SVM) and ν (OCSVM), $\{2^{-15}$, $2^{-13}, \ldots, 2^{3}\}$ for γ (both SVM and OCSVM) and $\{1, 2$, $\ldots, 64\}$ for the number of mixture of the GMM-UBM. The parameter's values that led to the highest result were then used during the test of the same fold.

For the Siamese network, a Glorot uniform weight initializer has been used for all layers. Adadelta has been used as optimizer algorithm with default initial parameters [86]. Different values of learning rate decay have been tried in the random-search. We trained the network for a maximum of 300 epochs, but an early stopping on the $F_1 - Measure$ has been used to interrupt training if there were no improvements for 25 consecutive epochs. At the ends, the model corresponding to the epoch that gave the best result was selected for the evaluation on the test set.

Table 6.9: Hyper-parameters optimized in the random-search phase, and their range.

Parameter	**Range**
Cnn layer Nr.	[2-5]
Kernel shape	[3x3-9x9] [3]
Kernel Nr.	[8-256]
MLP layers Nr.	[1-5]
MLP layers dim.	[30-8000]
Stride	[1x1-2x2]
Dilation	[1x1-20x20][3]
Batch size	[100-2000]
Max pool shape	[1x1-5x5][3]
Dropout	[Yes-No] [2]
Drop rate	[0.3-0.8]
Learning rate decay	[0-0.2] % [3]
Batch normalization	[Yes-No]

6.5 Results

The results obtained for each method are reported in Figure 6.8. 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.

[2] For all layers.
[3] After each epoch.
[5] Also not squared shape has been used.
[6] Value in the square brackets represents the value adopted for each layer.

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 others 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%.

As the fall detection is a task that needs to give a higher weight to the miss with respect to the false alarm, Figure 6.9 reports those metrics. Here can be seen that both SVMs methods outperform the others, obtaining an optimum false alarm rate of 0%. For the Miss rate, instead, while SVM reach a good result of 11%, SVM-unbalanced give an unacceptable result of about 50%. The Siamese network behaves in an opposite manner with respect to the SVM-unbalanced. Although it has an high false alarm rate, it manages to reduce to zero the miss rate. For a complete overview of the scores, in tables 6.10 to 6.12 are reported the normalized confusion matrix.

Another consideration is about the generalization capacity of these 4 methods. Figure 6.10 reports the score achieved by the algorithms for both validation and test set in the 10 folds. The trends show that while for the SVM there is always a set of hyper-parameter able to achieve a 100 $F_1 - Measure$ in validation phase, this does not happen for the SVM-unbalanced due to the lack of human falls during the training phase. Moreover, the results of validation are not always similar to the test value. The OCSVM instead, have a stable performance in validation, but they drop during the test, showing the poor generalization capacity of the algorithm. For the proposed method, instead, the trends in test follow closely the validation ones. This is highlighted in Figure 6.11 where the trends of the differences in the results between the tests and validations score are shown.

Table 6.10: Normalized confusion matrix of the SVM approach.

%	**Human Falls**	**Objects**
Human Falls	89	11
Objects	0	100

The Siamese Network seems to be promising. By using jus few human fall samples for the training and validation phase, it can reduce to zero the miss rate showing also a good generalization performance with respect to the others methods. All these elements suggest that it can be used in a real scenario, but it has to be assessed also in others conditions. In Section 6.6, a method based on this work will be evaluated inserting in the test even more types of

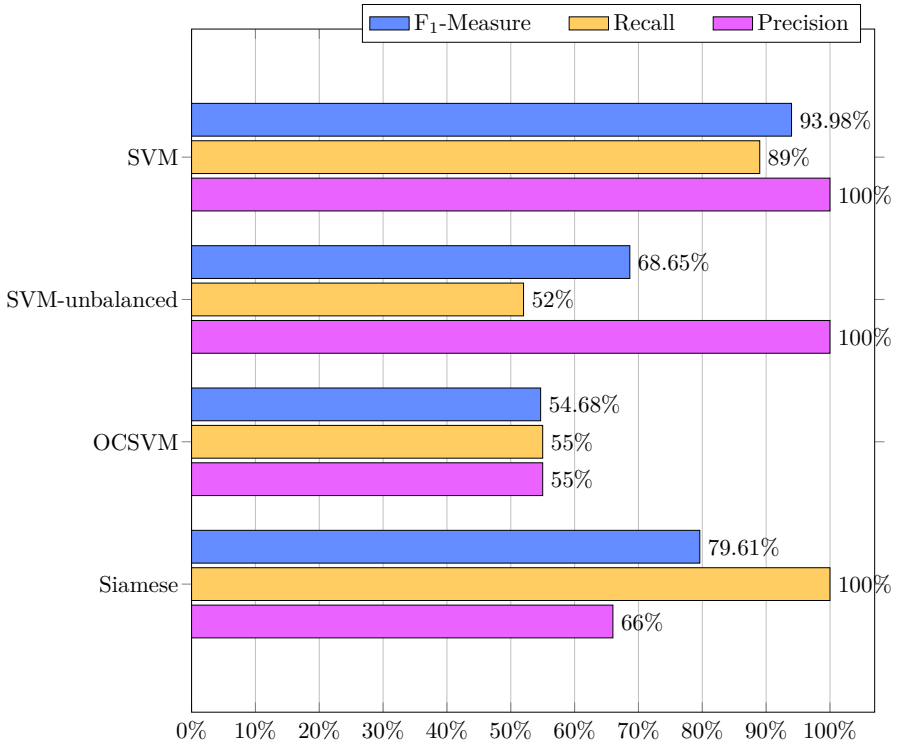


Figure 6.8: $F_1 - Measure$, precision and recall: the metrics are referred to the human fall class

audio events and noises. Another interesting evaluation, as future work, could be to evaluate how the system behaves if trained on different data sets with respect to the one used here, also using transfer learning techniques. This is important because it could relax the necessity of carrying out a data collection campaign in every environment in which the system is going to be installed. In addition, other types of networks such as Recurrent Neural Networks or Long Short-Term Memory that have not been explored in this work could be tested.

Table 6.11: Normalized confusion matrix of the SVM-unbalanced approach.

%	**Human Falls**	**Objects**
Human Falls	52	48
Objects	0	100

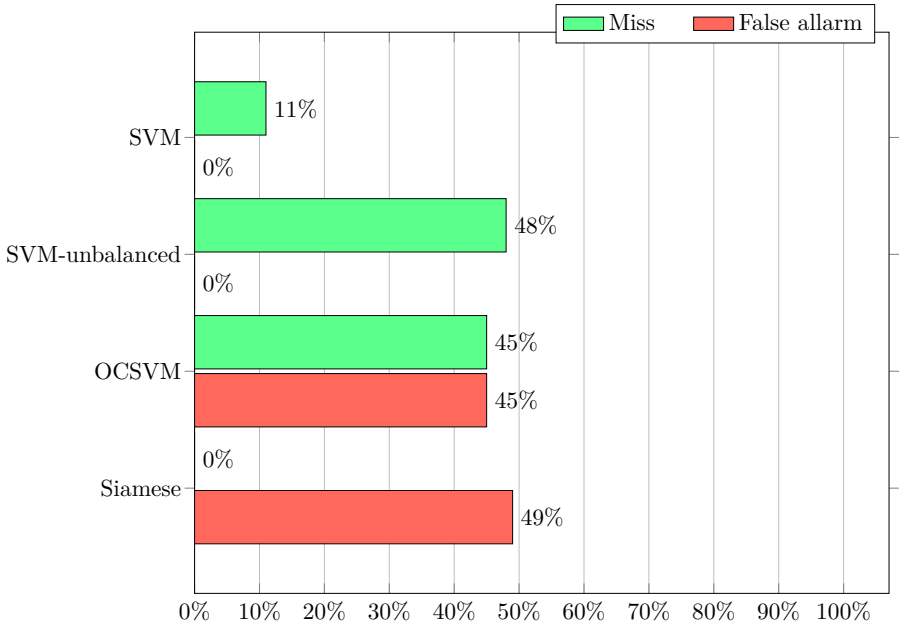


Figure 6.9: Miss and false alarm rate: the metrics are referred to the human fall class

Table 6.12: Normalized confusion matrix of the OCSVM approach.

%	**Human Falls**	**Objects**
Human Falls	55	45
Objects	45	55

6.6 Audio Metric Learning by using Siamese Autoencoders for One-Shot Human Fall Detection

As aforementioned, the fall detection task is very challenging due to the difficulty in retrieving examples for human fall modeling. Falls simulated by using a dummy may not represent properly real human falls (*RHF*), because they can not recreate falls in which arms are used to mitigate the impact. Moreover, the use of protections, such as mattresses, knee pads or foam during the acquisitions of falls performed by volunteers, can significantly modify the samples, especially in the audio field. We assess our proposed method in a complex scenario. The whole dataset described in Section 3.2 has been used. Moreover,

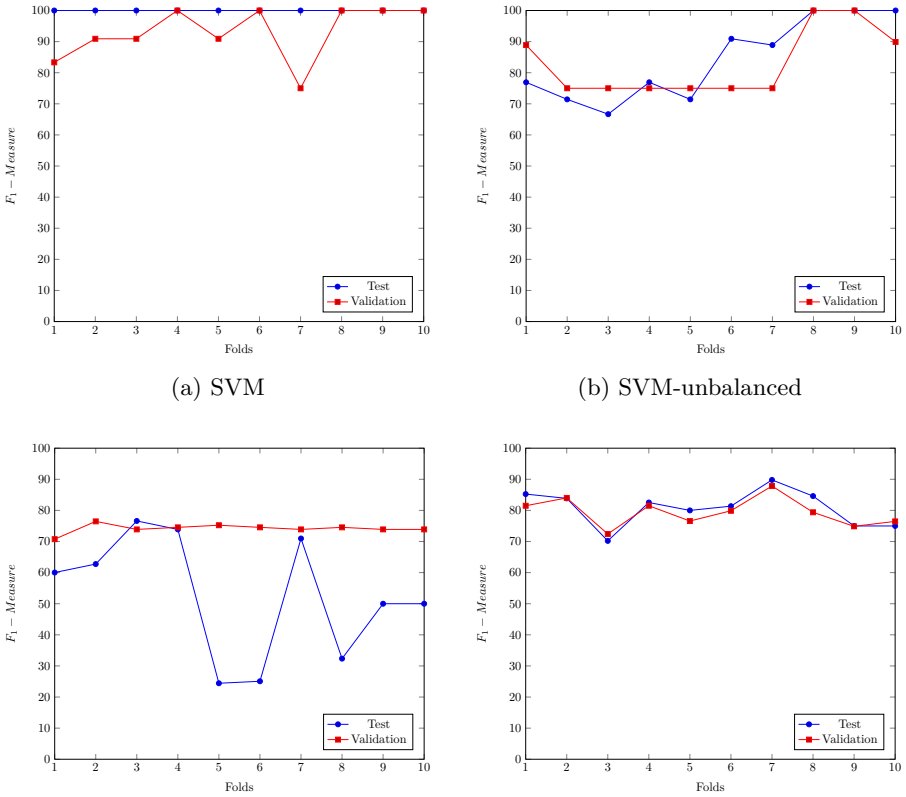
Table 6.13: Normalized confusion matrix of the Siamese Neural Network approach.

%	**Human Falls**	**Objects**
Human Falls	100	0
Objects	51	49

Table 6.14: Best hyper-parameters find in random-search phase for Siamese network[6].

Parameter	**Value**
Cnn layer Nr.	4
Kernel shape	[[3x3],[3x3],[3x3],[3x3]]
Kernel Nr.	[32,32,32,32]
Stride	[[1x1],[1x1],[1x1],[1x1]]
Dilation	[[1x1],[1x1],[1x1],[1x1]]
Max pool shape	[[2x1],[1x3],[1x2],[3x2]]
MLP layers Nr.	3
MLP layers dim.	[3000,300,30]
Batch size	[100]
Dropout	[Yes]
Drop rate	[0.5]
Learning rate decay	[0.1] %
Batch normalization	[yes,no,no,no,yes,yes,yes]

an innovative Siamese approach for fall detection, that exploits the similarities between simulated and real human falls by using only one *RHF* template per each room during training into a one-shot learning perspective is proposed. In order to do this, we used a Siamese Autoencoder (SAE) for metric learning. Selecting the pairs appropriately for training the SAE, the network learns how to map simulated falls to real falls applying a transformation directly into the embedding space. After that, we use the encoder part of the whole network to create new human fall templates by using simulated falls. Concluding, a classifier is trained by using the new synthetic templates in addiction with the real fall templates to discriminate between fall and non-fall event. As shown in Section 1.2, although the literature provides several supervised and unsupervised approaches, no solution has been proposed exploiting one-shot learning for fall detection, nor to fill the gap between simulated falls and scarcely available real human falls.



(a) SVM

(b) SVM-unbalanced

(c) OCSVM

(d) Siamese Network

Figure 6.10: $F_1 - Measure$ in validation and test achieved by the 4 methods in each folds

6.6.1 Dataset

The dataset used here in an extension of the dataset used in all our previous works that was created specially for this work: the new recordings have been performed in 2 different rooms (R1 and R2) with respect to the original one. Each room has different characteristics that make the propagation of waves less favorable: one room is paved with a fitted carpet floor, while in the other, the FAS was placed beyond a soundproof wall and the floor is made of stoneware tile. We also recorded other objects and background types in addition to those present in the previous dataset. Here the human falls were reproduced by volunteers without additional protections. Only the signals recorded with FAS has been used in this work. For a more detailed description, refer to the dedicated Section 3.2 and Table 3.1. This complete dataset allows a more exhaustive experimental evaluation of the Siamese approach highlighting its effectiveness in a one-shot learning framework with respect to other state-of-

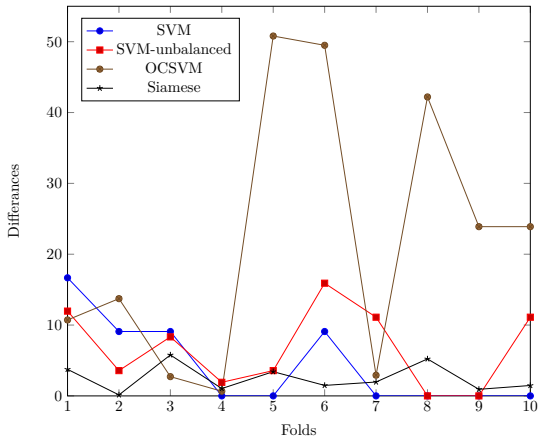


Figure 6.11: Absolute value of differences between the validation and test $F_1 - Measure$ for each fold

the-art methods.

6.6.2 Proposed Method

The proposed fall classification system is composed of 3 main parts described in this section. First, the features are extracted from row audio file and later used to train a Siamese neural network for metric learning. At the end a metric-based classifier is used to discriminate human falls from non-human falls.

Feature Extraction Stage

In the feature extraction stage, the raw audio signals have been preprocessed to compute the log mel-energies, thus obtaining a 2D matrix representing the samples. Such features have been chosen for their popularity in computational audio analysis [83, 85, 84].

Metric Learning Stage

The second block is a nonlinear metric learning stage based on Siamese Neural Network (SNN). The SNN is directly trained on semantic similarity information, that aims at modeling the relationships between classes in order to extract more robust features. The proficiency of a SNN mostly depends on the objective function used to train the network as well as the training set selection strategy. Our contribution consists in defining these two aspects. The proposed neural network architecture, depicted in Figure 6.12, consists in a Siamese Convolutional Autoencoder (SCAE). This is composed by an encoder that applies a transformation of the input into the latent space and a decoder that performs

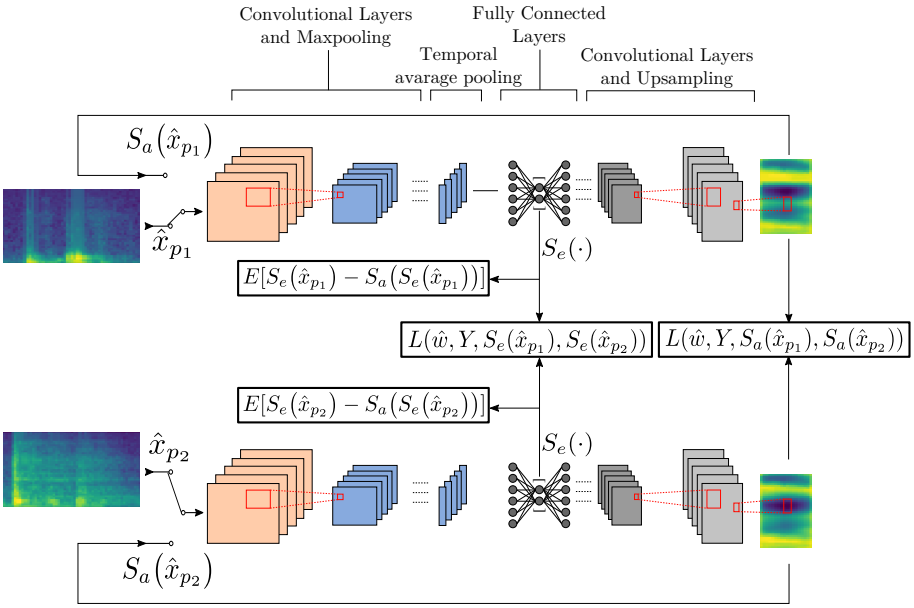


Figure 6.12: The architecture of the SCAE network for metric learning and robust feature extraction. The loss terms used for training the network are shown.

the reverse operation. The encoder includes some convolutional layers alternated - in some configurations - with max-pooling layers, followed by some fully connected layers. The encoder ends with a hidden layer representing the mapping of the inputs. Before the fully connected layers, average pooling is applied along the time dimension of the features map related to the last convolutional layer. This makes the network independent from the temporal length of the input signals. The decoder part is mirrored with respect to the encoder. Our Siamese Neural Network is devoted to learning a projection metric in a weakly-supervised way from input pairs in the form of positive and negative examples:

$$\mathcal{P} = \{(\hat{x}_{p_1}, \hat{x}_{p_2}) : \hat{x}_{p_1} \text{ and } \hat{x}_{p_2} \text{come from same distribution}\}, \tag{6.3}$$

$$\mathcal{N} = \{(\hat{x}_{p_1}, \hat{x}_{p_2}) : \hat{x}_{p_1} \text{ and } \hat{x}_{p_2} \text{come from different distributions}\} \tag{6.4}$$

with $\hat{x}_{p_1}$ and $\hat{x}_{p_2}$ the first and second input of the SCAE respectively. Considering $Y(\hat{x}_{p_1}, \hat{x}_{p_2})$ as the label assigned to the pair, training consists in finding a set of weights $\hat{w}$ that minimizes, for each input pair, the contrastive loss

function defined as

$$L(Y, S_e^w(\hat{x}_{p_1}), S_e^w(\hat{x}_{p_2})) = (1-Y)\frac{1}{2}(E_w)^2 + (Y)\frac{1}{2}\{(max(0, m - E_w)\}^2, \quad (6.5)$$

with

$$E_w = \|S_e^w(\hat{x}_{p_1}) - S_e^w(\hat{x}_{p_1})\|. \quad (6.6)$$

the Euclidean distance between the two mappings performed by the encoder and $S_e^w(x_p)$ the actual encoder function. To strengthen the metric learning, the decoder part of the network has been used to add two more terms to the loss function. The first is the contrastive loss function between the two signals reconstructed at the end of the autoencoder, that is $S_a^w(\hat{x}_{p_1})$ and $S_a^w(\hat{x}_{p_2})$. This is possible thanks to the average pooling layer previously mentioned. In fact, with this layer the autoencoder reconstructs a signal with fixed time length, regardless of the length of the input signals, allowing the Euclidean distance between the two reconstructions to be computed. Without the temporal average pooling layer, the reconstructed zero padded part of the input signals would control the value of the Euclidean distance, invalidating the result and leading to incorrect training. The typical capability of an autoencoder to exactly reconstruct the input signal is now lost. This is a fundamental feature that forces the autoencoder to engage in robust feature learning. Since it is impossible to add this term in a standard way due to the different dimensions between input $\hat{x}_p$ and reconstructed input $S_a^w(\hat{x}_p)$, the network has been forced to rebuild the latent layers of the input $S_e^w(\hat{x}_p)$ and the reconstructed signal $S_e^w(S_a^w(\hat{x}_p))$ by adding the following Mean Squared Error terms to the loss:

$$E[S_e(\hat{x}_{p_2}) - S_a(S_e(\hat{x}_{p_2}))]. \quad (6.7)$$

As mentioned previously, another crucial aspect of SNN is the selection of training pairs. Since we are creating a fall detection system that uses as few examples of *RHF* as possible, it is necessary to take full advantage of the limited information available. Moreover, since examples of simulated falls are available, the similarities between these two types of fall must be exploited. In order to do this, the training set for the SCAE was created by combining the following signals:

- to compose the negative set $\mathcal{N}$, all classes of objects available in the dataset and background noises have been used. No real or simulated human falls have been coupled with other sounds;
- all classes available in the dataset has been used to compose the positive set $\mathcal{P}$.

The idea is to let the network map the real and simulated without particular

constraints, in order to identify the hyper-space region where the *RHF* of the test set will be mapped. The only restriction is to make them similar to each other in the embedding representation of the autoencoder. This allows the network to learn, during the training, to make a transformation of human falls directly into the latent space. For further details, please refer to Section 6.7.1.

Classification Stage

In the end, the resulting function is used to improve the performance of metric-based classifier that discriminates falls from non-falls. All the train set is first transformed using the encoder function $S_e^w(\cdot)$. Moreover, we apply this transformation also to some instances of *SHF* previously left out of the train set of the SCAE, thus obtaining a total number of templates for the human fall equal to

$$\mathcal{T}_{hf} = \mathcal{T}_{shf} + \mathcal{T}_{rhf}, \tag{6.8}$$

with $\mathcal{T}_{shf}$ the number of *SHF* from R0 and $\mathcal{T}_{rhf}$ the total number of *RHF* templates selected from R1 and R2 used in SCAE training, two in our case. To train the *knn* classifier, a set of templates composed of $\mathcal{T}_{hf}$ instances have been selected for each other class in order to obtain a balanced training set. Besides, the parameter K of the classifier has been set to $\mathcal{T}_{hf}$. Finally, a human fall is detected if there is at least one human fall template in the set of $\mathcal{T}_{hf}$ neighbors related to the sample under test at that moment. This classification technique has been used to reduce to a minimum the miss rate which, for this particular application, have a greater weight compared to false alarm rate.

6.7 Comparative methods

In this section, the methods compared with the proposed work are summarized. The first method is based on a bi-class Support Vector Machine. It uses a mixture of gaussians (GMM), trained on a large corpus of audio events with the Expectation Maximization algorithm to model the acoustic space (Universal Background Model, UBM). Then, for each audio segment, the Maximum a Posteriori (MAP) algorithm is used to calculate the Gaussian Mean Supervector (GMS) from the MFCCs. Please refer to Section 4.2 for further details. This method will be shown in 2 variants, the first (SVM from now on) with a balanced train set, while the second (One-shot-SVM from now on) with an unbalanced train set for direct comparison with the proposed approach. The second is the unsupervised variation of the previous one based on OCSVM (Section 5.1). The third is the method that has been extended in this work, presented in Section 6.2 and named *Original Siamese* from now on. It consists of a simple SNN instead of SCAE thus equivalent to the encoded part of the pro-

posed autoencoder architecture, but without the average pooling layer before the fully connected layers. In Section 6.2 the algorithm worked in a more facilitated scenario as several human falls were used during training. Hear instead we perform the method in a one-shot learning framework. Since dummy-like falls were not used in this method, the pairs generation technique consists in the combination of the non-human fall data and the available template of *RHF* in order to compose the positive $\mathcal{P}$ and negative $\mathcal{N}$ as indicated in Eq. 6.3 and Eq. 6.4. Furthermore, a threshold based classifier is used. A human fall is detected if the sample is mapped within a radius from a real human fall template.

6.7.1 Experiments

This section presents, first, how data was selected for each compared methods. Then, an in-depth analysis of the preliminary experiments is discussed to gain insights on the behavior of the proposed approach and some of its variant. In the end, the classification performances of the optimized algorithms are reported. All the experiments have been assessed on the dataset described in Section 3.2, but signals have been downsampled to 8 kHz and the resolution has been reduced to 16 bits. All the following experiments have been conducted with 120000 pairs, on average between folds, for training the SCAE. Results are expressed in terms of F_1-Measure, calculated from the normalized confusion matrix, cumulative of all the folds. The same metric has also been used to optimize the results shown in Section 6.7.2. This choice was made to give more weight to false negatives than false positives, as the test set is highly unbalanced, being composed from 6973 non-human fall events and 390 human fall events in total. In particular, since the background tracks have been divided into segments of 5 seconds each, the non-fall events are composed of 5275 background instances and 1698 object fall events.

Data Splitting

Firstly, dataset has been split into 5 folds for cross-validation: in particular, the data related to R0 room, except for *SHF*, have been used only for training and used in each fold. Differently, the samples related to R1 and R2, except for *RHF*, have been split into 5 folds with 20% for test and 80% for the training set. Both simulated and real human falls have been treated differently, based on the algorithm under examination:

- SCAE: for the proposed approach, one *RHF* per room has been randomly selected for each fold and then added to the related trainset. The *SHF*, instead, have been split in 5 folds with 80% for train the SCAE, while the remaining 20% has been left out from the training set of the Siamese

network but used only to train the classifier as explained in Section 6.6.2. The pairs for training the SNN have been generated keeping balanced all the combinations between the classes.

- OCSVM: since this is a completely unsupervised method, both real an simulated human falls have been removed from trainset.
- SVM: since this is a completely supervised method, the *RHF* have been split in 5 folds with 20% for test and 80% for train and then added to the respective sets.
- One-shot-SVM: in order to keep this experiment comparable with the proposed method, the same selection carried out for the SCAE has been used for train the SVM, i.e. with just one real human fall sample for each environment to monitor.
- *Original Siamese*: the same lists used for SCAE have also been used for this approach. The only difference is that the *SHF* were not used because are not contemplated by this method.

Preliminary Experiments

Before running the optimization phase, the behavior of different pairs generation techniques for SCAE training has been studied. To do this, 4 preliminary experiments have been performed with a fixed autoencoder architecture, that has a hidden layer composed of just 2 neurons in order to visualize how the system maps the input signals when varying the way in which real and simulated falls are paired. Table 6.15 reports the hyper-parameter of that network. In Figure **??**, Figure **??**, Figure **??** and Figure **??** it can be seen how training and testing signals are encoded by the network, after its training, in the 4 different cases named $\mathcal{P}$-$\mathcal{N}$-PAIRS, $\mathcal{N}$-PAIRS, NO-PAIRS and $\mathcal{P}$-PAIRS . The mappings in Figure **??**, Figure **??**, Figure **??** and Figure **??** are then used to train the related *knn* classifier, following what was said in section Section 6.6.2. The final decision boundaries are reported in all figures: the data found in the white area are classified as human fall. Moreover, the *RHF* have been highlighted with orange stars while *SHF* with green X. The turquoise X instead, represent the encoded *SHF* previously left out from the SCAE trainset, but then added to the train set of the *knn* classifier. The 4 strategies for generating the pairs containing *RHF* and/or *SHF* are described below:

- $\mathcal{P}$-$\mathcal{N}$-PAIRS (Figure **??**): the *RHF* and *SHF* were joined together to form positive examples and with other signals to form negative examples. In this case, it can be observed how the network manages to separate the signals of human falls well during training. However, this is not

Table 6.15: Hyper-parameters used in the preliminary experiments, and their value

Parameter	Range	Parameter	Range
CNN layer Nr.	3	Drop rate	0%
Kernel shape	[4x4,4x4,4x4]	CNN Padding	same
Kernel Nr.	[4,4,4]	Batch Size	512
MLP layers Nr.	[3]	MLP Act.	ReLU[4]
MLP layers dim.	[40,512,2]%	Optimizers	Adadelta
Max pool shape	[1x2,2x3,2x3]	Weight Initializers	Glorot Uniform

generalized on the test set where the *RHF* are mapped to a different area. This means that the knowledge of a few human falls is not exploited properly.

- $\mathcal{N}$-PAIRS (Figure **??**): the human fall-related instances have been coupled only with signals of different classes, thus obtaining only a subset to add to the negative examples set. As it can be seen from Figure **??**, in this case, the contrastive loss tries to repulse the human fall instances from everything else, but without grouping them together (no positive examples were generated). This results in poor classification performance as shown in Table 6.16. Moreover, in both the $\mathcal{P}$-$\mathcal{N}$-PAIRS and $\mathcal{N}$-PAIRS methods, the usage of *SHF* seems not useful.

- NO-PAIRS (Figure **??**): neither *RHF* nor *SHF* were used for the training of the SCAE, but only for the training of the classifier. In this situation, the SCAE spreads the simulated human fall signals in the hyperplane (Figure **??**) that, when used to train the classifier, leads to the generation of too many false alarms as can be observed in Figure **??**.

- $\mathcal{P}$-PAIRS (Figure **??**): is best performing strategy. In this case, the human fall-related instances have been coupled only together, thus obtaining only a subset to add to the positive examples set. In this way, the SCAE is free to map human falls (both real and simulated) in the zone of space it wants, but keeping them close to each other. In this way, the network learns to transfer the features of the real falls to the simulated falls directly in the latent space. Now the *SHF* left out of the SCAE training can be used as additional templates for training the *knn* classifier.

The results for these preliminary experiments are reported in Table 6.16, where can be observed the best performance obtained with the fourth pairs generation strategy.

[4]In the decoder, an additional Cnn layer with *tanh* activation function has been used to ensure a good reconstruction.

Table 6.16: Preliminary result for different pairs generation strategies.

Technique	**Result in R1**	**Result in R2**	**Overall**
$\mathcal{P}$-$\mathcal{N}$-PAIRS	55.17%	64.74%	60.13%
$\mathcal{N}$-PAIRS	76.20%	67.53%	72.05%
NO-PAIRS	91.71%	89.88%	90.97%
$\mathcal{P}$-PAIRS	92.54%	92.54%	**92.54%**

Table 6.17: Hyper-parameters optimized in the random-search phase and their range.

Parameter	Range	Distribution
Cnn layer Nr.	[1-3]	Uniform
Kernel shape	[1x1-8x8]	Uniform
Kernel Nr.	[1-32]	Uniform
MLP layers Nr.	[1-2]	Uniform
MLP layers dim.	[1-4096]%	Log-uniform
Max pool shape	[0x0-3x3]	Uniform
Drop rate	[0-0.2]%	Uniform

6.7.2 Optimized results

After observing the results of the preliminary experiments, a random-search of 50 different configurations was performed to optimize the hyper-parameters of the SCAE approach with the pairs generation strategy best performing in the preliminary experiments. At the end, we selected the configuration that reached the highest F_1-Measure in the test set. Table 6.17 shows the value of the hyper-parameters used for network training and the relative search range. The unspecified parameters were left the same as those used in the preliminary experiments. The same random-search was also performed for the *Original Siamese* method, also optimizing the radius of the classifier used with this approach. For the SVM based methods a grid-search strategy has been adopted to optimize the parameters. In particular the parameters have assumed values in the ranges $\{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ for C (SVM) and ν (OCSVM), $\{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$ for γ (both SVM and OCSVM) and $\{1, 2, \ldots, 64\}$ for the number of mixtures of the GMM-UBM.

Figure 6.17 shows the results obtained for each approach. Regarding the completely supervised SVM method, although it is not directly comparable with the others due to different training and test set, it gives an idea of how much the task is hard also for such a system. Moreover, using an extremely unbalanced dataset for a supervised approach, as the same used for the Siamese network, leads to a very marked decrease in performance. In fact, the One-Shot SVM reaches an overall F_1-Measure of only 14.72%. When having an

extremely unbalanced dataset available is better to use a completely unsupervised method such as OCSVM, that achieves a score of about 72%. The best performing method is the SCAE that reaches a 93.58% of F_1-Measure, outperforming the *Original Siamese* of 3.25%. The improvement was significant for p < 0.002 according to one-tailed z-test [87]. The remarkable results obtained by both the *Original Siamese* and SCAE methods show that the use of Siamese framework is very powerful in this type of scenario, where only a few data for the class of interest are available and possibly some additional simulated data. In Table 6.18 and Table 6.19 the normalized confusion matrix for the Siamese

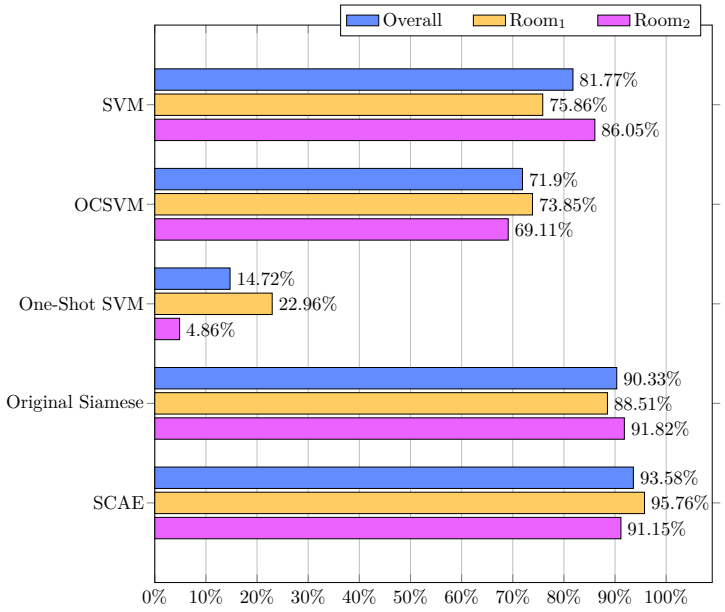


Figure 6.17: Results.

based approach are reported. As can be seen, the miss rate of the proposed method is less than 4% compared to the *Original Siamese* method, without losing in term of false alarm rate that, instead, has increased by 1%, resulting in good improvements regarding the reliability of the system. Since there are many instances of background noise in the dataset, the low number of false alarms indicates that this approach could also be used as a detection system.

6.8 Remarks

In this more realistic scenario, composed of sound events recorded in different environments and with different flooring and augmented with 80 human falls performed by four actors, the preeminence on the Siamese framework for one-

Table 6.18: Normalized confusion matrix of the *Original Siamese* approach. Absolute values are shown in brackets.

%	**Human Falls**	**Objects**
Human Falls	90 (6283)	10 (690)
Objects	9 (37)	91 (353)

Table 6.19: Normalized confusion matrix of the SCAE approach. Absolute values are shown in brackets.

%	**Human Falls**	**Objects**
Human Falls	91 (6362)	9 (611)
Objects	4 (17)	96 (373)

shot learning with respect to conventional methods has been shown. A further improvement in performance has been achieved with an extension of the method previously proposed in Section 6.2. It is composed of 3 stages: log-mel feature extraction, metric learning employing a Siamese autoencoder neural network named SCAE and, in the end, a final decision stage base on *knn* classifier. The network exploits the few information on the real fall by using a particular strategy of pairs generation for the SCAE training. In doing so, the system learns how to transform the available simulated human fall instances to create a more suitable set of templates that can be used to train the final classifier. Although the system seems to be reliable because of the low miss rate, the false alarm rate, of just about 3 false alarms raised every 2 real human falls, may even so be annoying for some users. To reduce this problem, several techniques could be employed. For instance, the system could be extended to include algorithms for fall recovery recognition able to detect whether a person is continuing his normal activity or if he/she is still lying in the ground.

Chapter 7

Other contributions

7.1 An End-to-End Unsupervised Network for Timbre Transfer in Musical Applications

A successful application of end-to-end computational intelligence in the field of image processing is the so-called "style transfer", i.e., the creative modification of a "content" image applying textures, strokes and colours from a reference image providing stylistic information. In the audio field, however, and specifically with music signals, the task is yet to be properly defined and addressed. More recently, researchers and machine learning developers asked themselves whether the algorithms that works for images were able to produce similar results with audio signals, and specifically, musical signals. In the past years, several hybridization techniques have been proposed to synthesize novel audio content owing its properties from two audio sources. These algorithms, however, usually provide no feature learning, leaving the user, often intentionally, exploring parameters by trial-and-error. The introduction of machine learning algorithms in the music processing field calls for an investigation to seek for possible exploitation of their properties such as the ability to learn semantically meaningful features. The main aim of this work is to propose a method to tackle the so-called *timbre transfer*, a topic that we consider being a subproblem of the more general *musical style transfer*. We adopt a Neural Network Autoencoder architecture and we enhance it to exploit temporal dependencies. In our experiments the architecture was able to modify the original timbre, resembling what it learned during the training phase, while preserving the pitch envelope from the input.

7.1.1 Proposed approach

Neural Network Architecture

The proposed neural network architecture is designed to capture spectral characteristics of audio signals in the time and frequency domain. The architecture

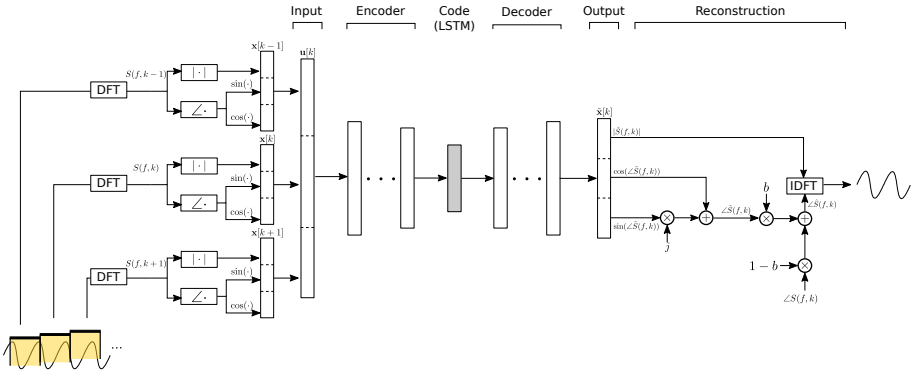


Figure 7.1: Overview of the proposed architecture.

adopted to perform this task is an autoencoder, i.e., a neural network trained to copy its input to its output [88]. A properly trained autoencoder, however, does not perform a simple copy operation, but learns significant characteristics of the training data. This property has been exploited in several different task in the literature, such as novelty detection [89], dimensionality reduction [90], and speech enhancement [91]. Here, the capabilities of the autoencoder are used to transform the input data based on the characteristics of the training data. The rationale here is that the autoencoder will try to reconstruct the input based on the knowledge acquired during the training phase, thus transforming the input signal based on the characteristics of the training data. An overview of the proposed architecture is shown in Figure 7.1 for the sake of clarity, and its details will now be presented.

Denoting with $S(f,k)$ the Short-Time Fourier Transform (STFT) at frame k of an audio signal $s[n]$, the network is designed to accept an input vector $\mathbf{u}[k]$ composed of the STFT magnitude and phase of frames adjacent to the one being processed. The phase is expressed as $\sin(\underline{\angle S(f,k)})$ and $\cos(\underline{\angle S(f,k)})$ terms. Denoting with $\mathbf{x}[k]$ the vector

$$\mathbf{x}[k] = \begin{bmatrix} |S(f,k)| \\ \cos(\underline{\angle S(f,k)}) \\ \sin(\underline{\angle S(f,k)}) \end{bmatrix}, \tag{7.1}$$

the network input vector $\mathbf{u}[k]$ is given by:

$$\mathbf{u}[k] = \begin{bmatrix} \mathbf{x}[k-1] \\ \mathbf{x}[k] \\ \mathbf{x}[k+1] \end{bmatrix}. \tag{7.2}$$

Indicating with F the size of the FFT, the vectors $\mathbf{x}[k]$ and $\mathbf{u}[k]$ are respectively

composed of $3F$ and $9F$ elements.

The encoding section of the network is composed of a stack of fully connected layers of gradually narrower size, reaching the inner hidden layer that yields the latent code of representation. Similarly, the decoding section is composed of a stack of fully connected layers, excepts for the fact that the layer size gradually grows from the inner hidden layer to the output layer.

The output layer of the network is composed of 3 groups of F neurons, so that the output vector is arranged as $\mathbf{x}[k]$. Each group is specialized to reconstruct a component of the complex $S(f,k)$ as:

$$\tilde{\mathbf{x}}[k] = \begin{bmatrix} \left|\tilde{S}(f,n)\right| \\ \cos(\angle\tilde{S}(f,k)) \\ \sin(\angle\tilde{S}(f,k)) \end{bmatrix}. \tag{7.3}$$

The ReLU activation function is only applied to the group related to the magnitude reconstruction, in order to constrain the output values to be positive. The tanh activation function is applied to all other neurons in the architecture, in order to allow the signals to assume values in the range $[-1,1]$.

Such a simple configuration is able to learn and reproduce an averaged spectrum, so that a dataset containing chromatic scales reproduces a signal that contains all musical notes at the same time. This is not sufficient in the current context, thus, temporal dependencies must be learned by the network. This can be obtained by using one or more recurrent layers as inner layer of the network. In particular, we used one hidden layer composed of Long-Short Term Memory block (LSTM). These blocks can efficiently exploit a long-range temporal context by means of connections between units which form directed cycles, and store state information in the cell.

A feature-wise batch normalization is applied to the output of each layer of the network in order to reduce the internal covariance shift and to better distribute the latent representations obtained during the network training procedure. In addition, the dropout technique was employed during the neural network training to prevent overfitting and increase the generalization performance of the neural network in reconstructing the input signal.

Training is performed by using a dataset composed of audio signals characterized by the desired timbre. The network is trained to minimise the mean squared error between the estimated signal $\tilde{\mathbf{x}}[k]$ and the input signal $\mathbf{x}[k]$ by using the AdaDelta stochastic gradient-based optimization algorithm. It was chosen because it is well-suited for dealing with sparse data and is robust to different choices of model hyperparameters. Furthermore no manual tuning of learning rate is required.

Resynthesis

During the generation phase, a novel input is employed featuring a different instrument/timbre from the ones used during training. Special care must be taken in the inverse STFT processing in order to provide time-domain reconstruction without phase artefacts. Owing from works in cross-synthesis and spectral morphing [92], the predicted spectral magnitude and phase can be mixed with the magnitude and phase of the input signal. Denoting with $\hat{S}$ the DFT of the final reconstructed signal, the output magnitude and phase are obtained as:

$$|\hat{S}| = a|S| + (1-a)|\tilde{S}| + a_M\sqrt{|S| \cdot |\tilde{S}|}, \tag{7.4}$$

$$\angle\hat{S} = b\angle\tilde{S} + (1-b)\angle S, \tag{7.5}$$

where $0 < a, b, a_M < 1$ determine the proportion between the estimated and original signal components. In practice, the original magnitude information is not used, i.e., $|\hat{S}| = |\tilde{S}|$, while choosing b close to 0.5 allows to obtain a time domain signal with reduced artefacts. This is the choice followed for all reported experiments.

7.1.2 Comparative Methods

To provide a comparison to the proposed approach, the following methods for timbre transfer have been implemented to be tested with the same audio material. The implemented algorithms are:

- **spectral envelope hybridization**: in this case, the timbre transfer is performed by computing the spectral envelope of both signals, flattening the envelope of the target signal by deconvolution and then multiplying it by the source signal; in this context, the spectral envelope is based on the cepstrum as follows:

$$E = DFT(W_{LP}(DFT^{-1}(log(|DFT(s)|)))) \tag{7.6}$$

  where W_{LP} is a low-pass filter in the cepstral domain called *liftering* filter and S is a signal in time domain;

- **MFCC-based mosaicing**: this method performs timbre transferring by replacing the frames of the source signal by specific frames of a target dataset of sounds (that we call dictionary), where the match is done by some distance measure in a specific domain; in this context, we used a k-nn algorithm applied on the first 14 Mel-frequency cepstral coefficients (MFCC) of each frame [93].

For the MFCC-based approach, the length of the generated output sound is equal to the length of the input sound. For the other methods, instead, the generated output is as long as the longest between input and target, where the shorter one is repeated as necessary during the process.

The spectral envelope hybridization and the MFCC-based mosaicing methods do not require parameters and perform a full timbre transfer between the processed signals; on the other hand, the DFT morphing requires the setting of the interpolation parameters. In this context we decided to create the output sound by using the phases of the source signal and the amplitudes of both signals while keeping some bias on the target signal. To achieve this, we set the interpolation parameters as follows: $a = 0, b = 1, a_M = 1$.

7.1.3 Experiments

Experimental Setup

The algorithm has been implemented in the Python programming language using the Keras deep learning libraries. All the experiments were performed on a CINECA Galileo computational node with Nvidia K80 accelerators.

The neural networks were trained with the Adadelta gradient descent algorithm and a learning rate equal to 1.0, $\rho = 0.95$, $\epsilon = 10^{-6}$. The maximum number of epochs was set to 300 and an early stopping strategy on the validation set loss with 20 epochs of patience was employed for regularization. Each training iteration involved a number of samples (i.e., batch size) between the 5% and the 10% of total samples present in the training set, depending on the amount of samples present on the latter.

The network weights were initialized with a random Gaussian distribution with mean $\mu = 0$ and standard deviation $\sigma = 0.1$, as it usually provides an acceptable initialization in our experience. Several network topologies were tested, varying the number of layers and units per layer. Indeed the number of layers of the encoder has been varied from 1 to 4 while the number of unit for each layer from 80 to 4096. The decoder part was mirrored with respect to the encoder. The number of LSTM layers has been varied from 0 (no LSTM layer) to 1 with a number of units from 80 to 1024.

Two datasets have been employed for the evaluation of the proposed approach. Dataset 1 is an internal dataset of recorded solo instrumental or vocal tracks. Each audio file consisted of 30-120 seconds of audio. The following instruments were considered: clean electric guitar (2 files), distorted electric guitar (3 files), synth pad (2 file), trumpet (3 files), electric piano (3 files), male voice (1 file), female voice (2 files). All the files are real recordings of solo performance and thus contains a rich content of notes, chords (for polyphonic instruments), legatos, glissandi, and other expressive effects. Each file is played

on a single tonality.

Dataset 2 is built from a very large musical instrument dataset shared by the Magenta project team[1], containing single notes for eleven classes of musical instruments. In creating Dataset 2 we picked ten from the eleven classes, discarding the bass class because of its narrow coverage of the frequency spectrum, and randomly selected 3500 audio files for each class, to give each class the same dimensionality. This dataset is composed of short files containing a single note each, at different dynamic levels and pitches. The differences between the two datasets have been exploited to observe the role of the training data on the resulting output.

All files from Dataset 1 have been sub-sampled to 22050 Hz to reduce the computational cost, while the files from Dataset 2 are sampled at 16 kHz and were left as such. The STFT of the audio signals were computed with a 2048-points DFT, 50% overlap and window size calculated in order to reach 43 frames per second.

Audio samples are available at `https://gitlab.com/a3labPapers/CompanionFiles/tree/master/AES-XSynth`.

7.1.4 Results

This section reports experiments with different audio sources as target and input conducted on Dataset 1 and 2 together with a qualitative analysis of the synthesized outputs. A first batch of experiments was conducted with Dataset 1 to tune the network hyperparameters. The outputs produced by autoencoders trained on each file in the dataset have been evaluated by analysis of their spectrograms and informal listening tests. The hyperparameters reported in Table 7.1 have been found to obtain acceptable results, which will be discussed shortly. Without considering the output layer and input layers, the network is composed of five layers: 2 layers respectively of 1024 and 808 units, one LSTM layer composed of 808 units, and 2 layers respectively of 808 and 1024 units.

Reconstruction Tests

Preliminary tests were conducted to ensure that the autoencoder is able to reconstruct an input signal if it is also used for training. The network is able to reconstruct sufficiently well the original file, although some compression is inherent to the compressive autoencoding process. Figure 7.2 shows the spectrogram of an electric guitar from Dataset 1 playing arpeggios and chords and its reconstruction. The compression is visible from the spectrograms especially at high frequency.

[1] https://magenta.tensorflow.org/nsynth

Table 7.1: Hyperparameters used for the experiment with training on the distorted guitar track of the song *Sweet Child O' Mine*.

Network layout	**Dropout**
Encoder: 1024, 808 LSTM: 808 Decoder: 808, 1024	input units to drop $p = 0.1$
Training epochs	**Optimiser parameters**
300, 20 patience Validation split = 10%	learning rate = 1 $\rho = 0.95$, $\epsilon = 10^{-6}$
Batch Normalization: $\epsilon = 10^{-6}$, $\mu = 0.9$	

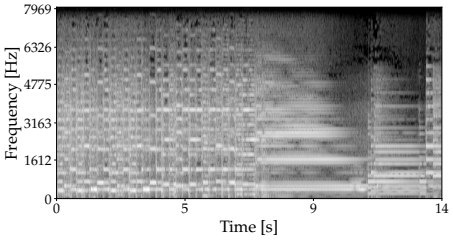
Dataset 1

These experiments were done with a female singing voice input file (from now on, in short FV1). This choice is motivated by the fact that the voice has subtle pitch variations (vibratos, glissandi, etc.) and the voice presents pitched, unpitched and unvoiced audio.

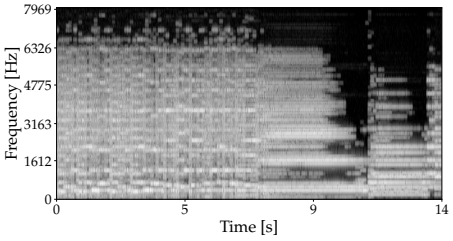
As an example of the results that can be obtained by the proposed architecture, some selected outputs are analysed in detail. A distorted guitar track from Dataset 1 (playing the tune in *Sweet Child O' Mine* by Guns N'Roses) is taken as the target for the proposed architecture with parameters shown in Table 7.1. The resulting output file is named R1 for short, and its spectrogram is shown in Figure 7.3(b). For comparison, the distorted guitar track has also been used for the MFCC-based mosaicing algorithm and for the spectral flattening algorithm. The input file used for all techniques is FV1. Its spectrogram is shown in Figure 7.3(a), followed by the spectrograms obtained by the other methods.

With the proposed approach, timbre is quite coherent from frame to frame if compared, for example, to the MFCC-based method, thus resulting in a more convincing output. In the MFCC hybridization, furthermore, it is also quite apparent that frames from the target signal appear from time to time in an inconsistent way exposing explicit features of the target (for example, a recognizable note in a riff). Finally, the spectral flattening method has a recognizable vocoder-like timbre, with the musical structure of the target file appearing together with its spectral content, which is an undesired feature in this context (see, for example, the arpeggios of the target song appearing in the spectrogram and chromagram approximately at 5 s on to the end).

Chromagrams from audio files shown in Figure 7.3 are shown in Figure 7.4, to compare the pitch trajectories and the presence of spurious chromatic components. The chromagrams obtained from the proposed architecture (Figure



(a) Original electric guitar track (input and target).



(b) Reconstruction.

Figure 7.2: Spectrograms from (a) an electric guitar track, (b) its reconstruction when using (a) as both target and input. The hyperparameters for (b) are reported in Table 7.1.

7.4(b),(e)) show similar pitch trajectories to the input signal (Figure 7.4(a)). We observe that the network fails to follow the pitch of the input when it is outside the range learned during the training phase. This can be seen around second 2 in Figure 7.4(e), where the high pitch of the input file reaches a B4 which the network cannot match, due to the lack of notes above G#4 in the training set.

Dataset 2

The experiments with Dataset 2 were conducted by training an autoencoder for each of the ten instrument classes, thus greatly increasing the complexity of the problem. Each trained autoencoder was subsequently used to synthesize audio with FV1 as input, resulting in ten audio files of different timbre and similar pitch contour. As an example of the good pitch tracking capabilities of the proposed architecture, we report a DFT (4096 points) calculated for each file at a specific time interval, where the input file shows a pitched */i:/* phoneme at frequency 497.6 Hz (B4 + 13 cents), see Figure 7.5. Each DFT has different features (e.g., spectral slope, spectral centroid, presence of noise, etc.) but all have same pitch.

From the experiments with Dataset 2 we observed that the network cannot

follow glissando, pitch bending and vibrato from the input because the dataset has no time-varying pitch to be learned, being composed of static single notes only. This is apparent by applying FV1 as input and results in lower-quality note transitions compared to the autoencoder trained on Dataset 1.

Judging timbre learning and transferring with this dataset is difficult because of the extremely large variety of tones in an instrument class and the large number of files to evaluate. The network cannot learn all the timbres of the different instruments in a class, but it learns instead an averaged spectrum of the whole class. It must be noted that we are not conducting supervised training in this work, thus, the network was not instructed with labels regarding the instrument type or any other property of the target sound that could help in clustering the timbre families inside a class. The bandwidth of the output is related to that of the class used for training with, e.g.,brass instruments having a wide spectrum and flutes having a reduced bandwidth when producing an output with the same input. We also noted that the autoencoder trained on vocal samples produces tones with a voice-like texture.

Pitch Tracking Accuracy

We conducted more systematic tests to quantify the pitch tracking accuracy of the architecture. These tests were conducted with Dataset 2 because of the precise pitch of its content and its stability over time, allowing for an easier pitch accuracy evaluation. Five audio files were generated systematically from a MIDI file containing 20 random notes. Each audio file was generated from a different digital instruments using sampling synthesis and employing equal temperament and 440.0 Hz tuning. Using the 5 files as input to the 10 networks, resulted in 50 outputs for a total of 1000 notes.

Pitch accuracy was evaluated by employing a peak picking algorithm in the frequency domain [94], using a thresholded parabolically-interpolated STFT[2]. For each note, the accuracy was evaluated considering 128 pitch classes (those defined by the MIDI protocol), plus one unpitched class (i.e.,the output shows no clear pitch information, despite the input content which was always pitched). A pitch tolerance of ± 50 cents and an octave tolerance of ± 1 were allowed.

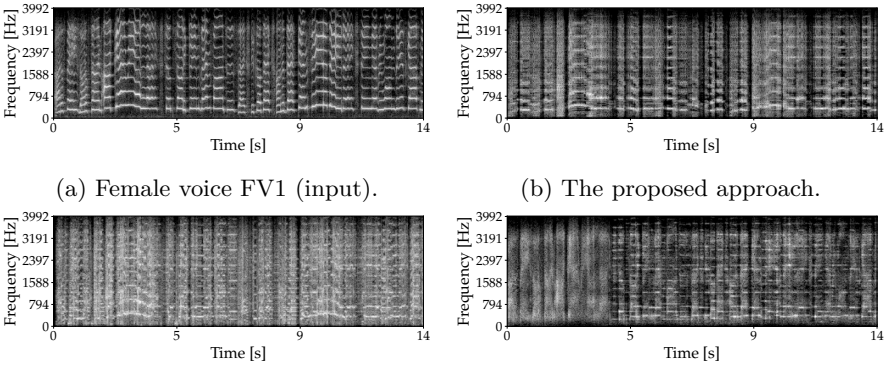
Only 12% of the 1000 notes lost the original pitch, showing a good reliability of the network in retaining the original pitch of the input.

Discussion

Overall, some properties of the proposed method are summarized.

- The output pitch follows quite closely the pitch of the input signal if the training set contains pitched audio in the same range as the input,

[2]https://librosa.github.io/librosa/generated/librosa.core.piptrack.html



(a) Female voice FV1 (input).

(b) The proposed approach.

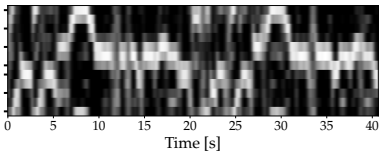
(c) k-nn matching techinque.

(d) Spectral flattening technique.

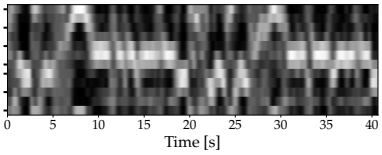
Figure 7.3: Spectrograms from the input female voice FV1 (a), the proposed approach and the comparative methods(c-d). The hyperparameters for (b) are reported in Table 7.1.

however, the network is not able to generalize above or below the pitch range learned from the target file.

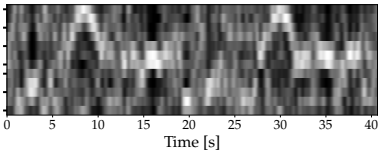
- For multi-pitched inputs (e.g., containing chords) the network is usually able to generalize providing a harmonization similar to the input.
- The timbre of the output signal resembles that of the training set, if the latter is sufficiently homogeneous (e.g., solo instrument from a track or separate notes from a specific instrument).
- Unpitched frames in the input audio are mapped to unpitched frames in the output if the training set contains unpitched material.
- Spectral continuity has intermediate quality between k-nn approaches and whitening approaches, i.e., frames do not change abruptly, thanks to the input context and the LSTM layer, but may still have deviations on a larger time basis. Previous experiments without context and the LSTM layer resulted in an output subject to abrupt variations from frame to frame similar to mosaicing.
- The algorithm does not guarantee that the frame energy of the input is transferred to the output. If the training set does contain sufficient levels of dynamic to learn there is a higher chance that energy is preserved, but it is not always guaranteed. The shortcoming of this is the generation of outlier frames when the input is silent or of low energy; this can be addressed by applying the input frame energy to the output by conventional signal processing techniques, but some constraints may be applied to allow the network learning energy preservation.



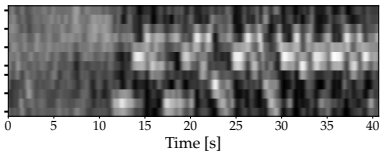
(a) Female voice FV1 (input).



(b) The proposed approach.



(c) k-nn matching technique.



(d) Spectral flattening technique.

Figure 7.4: Chromagrams from (a) FV1, (b) to (d) different timbre transfer approaches using a distorted guitar track as a target and FV1 as input. Vertical axis shows the 12 notes (ticks correspond to natural notes).

The proposed architecture is also able to perform morphing in the frequency domain, according to equations 7.4-7.5.

7.2 Conclusion

This work discussed how machine learning can be applied to the synthesis of novel sounds from existing sources, i.e. how it can provide a new class of *transfer* algorithms that has different properties with respect to conventional algorithms for morphing, hybridization, etc. A neural autoencoder has been proposed to the task and tested with different settings. Salient properties are the ability to transfer the timbre of a training set to an input file, while preserving its pitch. Pitch preservation has been proved to be reasonably good with systematic tests, while timbre transfer still needs improvement, especially with a large corpus of input signals.

Compared to dictionary-based algorithms, such as k-nn matching on MFCC, it proves to be more robust in term of frame-to-frame coherence, because of its improved generalization properties and possibly more relevant frame mapping.

Nonetheless, the timbre transferring capabilities of the algorithm are not

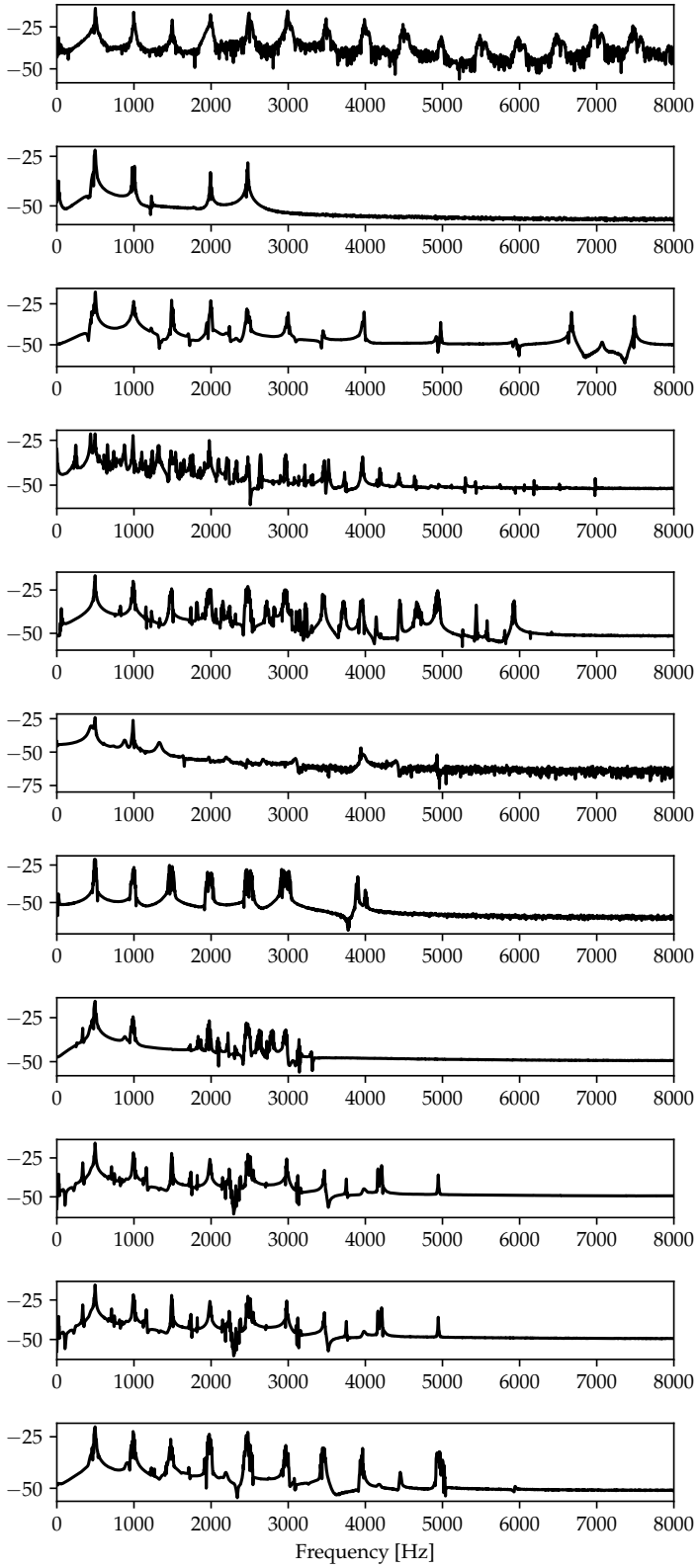


Figure 7.5: Windowed DFTs taken from each instrument class output with FV1 as input, at the position where a pitched /i:/ phoneme takes places in FV1. Each DFT shows similar pitch, although timbres differ. Instrument classes, top to bottom: brass, flute, guitar, keyboard, mallet, organ, reed, string, synth-lead, vocal.

systematically evaluated because it is very sensitive to the features of its input and target signals and an evaluation framework is missing. More research work is required to make the network robust to changes in frame energy, spectral bandwidth, time decay and pitch range of the target and input audio, in order to increase the intelligibility and the quality of the output signal. Furthermore, the network must be able to generalise for pitch ranges not learned from the target signal, and must be more expressive in order to learn different timbres in a large dataset.

Chapter 8

Conclusions

In this dissertation, the problem of human fall detection has been widely addressed. An innovative sensor named FAS was initially proposed Section 3.1. Its operating principle is similar to the one of stethoscopes: a membrane is in contact with the floor, i.e., the transmitting medium of the fall waves and a resonance enclosure accommodates a microphone that captures the waves and converts them in electrical signals. The floor sensor minimizes the impact of aerial sounds into the audio recordings, making it suited to deal with sounds induced by falls. In order to evaluate the effectiveness of the floor acoustic sensor and due to the lack of publicly available audio dataset for fall detection, we have created a dedicated corpus comprises recordings of fall events related to everyday objects and background noises. The sounds have been collected in 3 different rooms with different acoustic characteristics. The human fall has been simulated by means of a human mimicking doll. The dataset has been made publicly available to the research communities. By analyzing the signals collected in the database Section 3.2.3, we have highlighted the behaviour of the FAS that show a good SNR at low frequencies with respect to the other standard microphones. That allowed to work at low sampling frequencies, considerably reducing the computational cost of the algorithms. In particular, in Chapter 4 the two types of microphones were compared using SVM-based supervised approach. It has been shown that by working at lower sampling rates the FAS can achieve better results than the other microphones due to its acoustic properties. However, human falls are tough to find in a real situation. The supervised approaches are therefore not feasible. In chapter x, therefore, methods have been proposed that work in the opposite condition: the non-supervised approaches indeed can work without examples of the target class for their training. In Section 5.1 an OCSVM method has been proposed and trained with background noises only. Although the proposed system can work well when the test set is composed by only the same categories used for the training plus the human fall class, performance decays when the test set is composed of signals not included in the normality model, such as events of falling objects. Indeed, the challenge in such approaches is to have enough

data to shape normality. In Section 5.2 has been proposed a different novelty detection approach that works in an end-to-end learning process. A different novelty detection approach that works with an end-to-end learning process. This system employs a neural network autoencoder and to remove the need for handcrafted features. It has been shown that this system outperforms the OCSVM based method in the less difficult scenario where the teste is composed only of the background noises and the human fall signals.

List of Publications

[1] 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. 51–60. vol. 60. Elsevier, 2016.

[2] D. Droghini, E. Principi, S. Squartini, P. Olivetti, and F. Piazza, "Human Fall Detection by Using an Innovative Floor Acoustic Sensor," *Proc. of WIRN*, Vietri sul Mare, Italy. May, 18-20, 2016.

[3] E. Principi, D. Droghini, S. Squartini, P. Olivetti, and F. Piazza, "A Combined One-Class SVM and Template Matching Approach for User-Aided Human Fall Detection by Means of Floor Acoustic Features," *Computational Intelligence and Neuroscience*, Hindawi, 2017.

[4] D. Droghini, D. Ferretti, E. Principi, S. Squartini, and F. Piazza, "An End-To-End Unsupervised Approach employing Convolutional Neural Network Autoencoders for Human Fall Detection," *Proc. of WIRN*, Vietri sul Mare, Italy. June, 14-16, 2017.

[5] 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.

[6] 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.

[7] 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.

[8] E. Principi, D. Droghini, S. Squartini, P. Olivetti, and F. Piazza, "A Combined One-Class SVM and Template Matching Approach for User-Aided Human Fall Detection by Means of Floor Acoustic Features," *Engineering Applications of Artificial Intelligence*, Elsevier, 2018. Submitted.

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] 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] K. Ducatel, U. européenne. Technologies de la société de l'information, U. européenne. Institut d'études de prospectives technologiques, and U. européenne. Société de l'information conviviale, *Scenarios for ambient intelligence in 2010*, Office for official publications of the European Communities Luxembourg, 2001.

[2] M. Weiser, "The computer for the 21 st century," *Scientific american*, vol. 265.3, pp. 94–105, 1991.

[3] "Eurostat statistic explained: Fertility statistics," March 2016.

[4] G. Carone and D. Costello, "Can europe afford to grow old?," *Finance and Development*, vol. 43, no. 3, pp. 28–31, 2006.

[5] P. Dawadi, D. Cook, and M. Schmitter-Edgecombe, "Automated cognitive health assessment from smart home-based behavior data," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 4, pp. 1188–1194, 2016.

[6] 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.

[7] M. Mubashir, L. Shao, and L. Seed, "A survey on fall detection: Principles and approaches," *Neurocomputing*, vol. 100, pp. 144–152, 2013.

[8] "Department of Health and Human Services: World's older population grows dramatically," `http://www.who.int/en/news-room/fact-sheets/detail/falls`, [Online; accessed 30-Oct-2018].

[9] "World Healt Organization: Falls," `http://www.who.int/en/news-room/fact-sheets/detail/falls`, [Online; accessed 30-Oct-2018].

[10] S. S. Khan and J. Hoey, "Review of fall detection techniques: A data availability perspective," *Medical engineering and physics*, vol. 39, pp. 12–22, 2017.

[11] N. Lapierre, N. Neubauer, A. Miguel-Cruz, A. R. Rincon, L. Liu, and J. Rousseau, "The state of knowledge on technologies and their use for fall detection: A scoping review," *International journal of medical informatics*, 2017.

[12] N. Pannurat, S. Thiemjarus, and E. Nantajeewarawat, "Automatic fall monitoring: a review," *Sensors*, vol. 14, no. 7, pp. 12900–12936, 2014.

[13] T. Xu, Y. Zhou, and J. Zhu, "New advances and challenges of fall detection systems: A survey," *Applied Sciences*, vol. 8, no. 3, pp. 418, 2018.

[14] N. El-Bendary, Q. Tan, F. C. Pivot, and A. Lam, "Fall detection and prevention for the elderly: A review of trends and challenges.," *International Journal on Smart Sensing & Intelligent Systems*, vol. 6, no. 3, 2013.

[15] Q. Wu, Y. D. Zhang, W. Tao, and M. G. Amin, "Radar-based fall detection based on doppler time–frequency signatures for assisted living," *IET Radar, Sonar & Navigation*, vol. 9, no. 2, pp. 164–172, 2015.

[16] T. Liu, H. Yao, R. Ji, Y. Liu, X. Liu, X. Sun, P. Xu, and Z. Zhang, "Vision-based semi-supervised homecare with spatial constraint," in *Pacific-Rim Conference on Multimedia*. Springer, 2008, pp. 416–425.

[17] F. Werner, J. Diermaier, S. Schmid, and P. Panek, "Fall detection with distributed floor-mounted accelerometers: An overview of the development and evaluation of a fall detection system within the project ehome," in *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2011 5th International Conference on*. IEEE, 2011, pp. 354–361.

[18] Y. Zigel, D. Litvak, and I. Gannot, "A method for automatic fall detection of elderly people using floor vibrations and sound—proof of concept on human mimicking doll falls," *IEEE Trans. Biomed. Eng.*, vol. 56, no. 12, pp. 2858–2867, 2009.

[19] Y. Li, K. Ho, and M. Popescu, "A microphone array system for automatic fall detection," *IEEE Trans. Biomed. Eng.*, vol. 59, no. 5, pp. 1291–1301, 2012.

[20] M. Popescu, Y. Li, M. Skubic, and M. Rantz, "An acoustic fall detector system that uses sound height information to reduce the false alarm rate," in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*. IEEE, 2008, pp. 4628–4631.

[21] M. S. Khan, M. Yu, P. Feng, L. Wang, and J. Chambers, "An unsupervised acoustic fall detection system using source separation for sound interference suppression," *Signal processing*, vol. 110, pp. 199–210, 2015.

[22] X.-X. Zhang, H. Liu, Y. Gao, and D. H. Hu, "Detecting abnormal events via hierarchical dirichlet processes," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2009, pp. 278–289.

[23] M. Popescu and A. Mahnot, "Acoustic fall detection using one-class classifiers," in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*. IEEE, 2009, pp. 3505–3508.

[24] E. E. Stone and M. Skubic, "Fall detection in homes of older adults using the microsoft kinect," *IEEE J. Biomed. Health Inform.*, vol. 19, no. 1, pp. 290–301, 2015.

[25] A. Bourke, J. O'Brien, and G. Lyons, "Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm," *Gait and Posture*, vol. 26, no. 2, pp. 194–199, 2007.

[26] Y. Charlon, N. Fourty, W. Bourennane, and E. Campo, "Design and evaluation of a device worn for fall detection and localization: Application for the continuous monitoring of risks incurred by dependents in an Alzheimer's care unit," *Expert Systems with Applications*, vol. 40, no. 18, pp. 7316–7330, 2013.

[27] A. Özdemir and B. Barshan, "Detecting Falls with Wearable Sensors Using Machine Learning Techniques," *Sensors*, vol. 14, no. 6, pp. 10691–10708, 2014.

[28] M. Lustrek, H. Gjoreski, N. Gonzalez Vega, S. Kozina, B. Cvetkovic, V. Mirchevska, and M. Gams, "Fall detection using location sensors and accelerometers," *Pervasive Computing*, vol. 14, no. 4, pp. 72–79, Oct 2015.

[29] M. Alwan, P. J. Rajendran, S. Kell, D. Mack, S. Dalal, M. Wolfe, and R. Felder, "A smart and passive floor-vibration based fall detector for elderly," in *Proc. of Inf. Commun. Technol.*, 2006, vol. 1, pp. 1003–1007.

[30] A. Yazar, F. Keskin, B. U. Töreyin, and A. E. Çetin, "Fall detection using single-tree complex wavelet transform," *Pattern Recognition Letters*, vol. 34, pp. 1945–1952, 2013.

[31] X. Zhuang, J. Huang, G. Potamianos, and M. Hasegawa-Johnson, "Acoustic fall detection using Gaussian mixture models and GMM supervectors," in *Proc. of ICASSP*, Taipei, Taiwan, Apr. 19-24 2009, pp. 69–72.

[32] L. Liu, M. Popescu, M. Skubic, and M. Rantz, "An automatic fall detection framework using data fusion of Doppler radar and motion sensor network," in *Proc. of the 36th International Conference of the Engineering in Medicine and Biology Society (EMBC)*, 2014, pp. 5940–5943.

[33] C. N. Doukas and I. Maglogiannis, "Emergency fall incidents detection in assisted living environments utilizing motion, sound, and visual perceptual components," *IEEE Trans. Inf. Technol. Biomed.*, vol. 15, no. 2, pp. 277–89, Mar. 2011.

[34] B. Toreyin, A. Soyer, I. Onaran, and E. Cetin, "Falling person detection using multi-sensor signal processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, no. 1, pp. 7, 2008.

[35] N. Noury, A. Fleury, P. Rumeau, A. Bourke, G. Laighin, V. Rialle, and J. Lundy, "Fall detection-principles and methods," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*. IEEE, 2007, pp. 1663–1666.

[36] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[37] C. Bishop, *Pattern Recognition and Machine Learning*, Springer Science+Business Media, LLC, New York, 2006.

[38] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in neural information processing systems*, 2000, pp. 582–588.

[39] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems*. 2000, vol. 12, pp. 582–588, MIT Press.

[40] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge University Press, 2004.

[41] 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.

[42] W. Campbell, D. Sturim, and D. Reynolds, "Support vector machines using GMM supervectors for speaker verification," *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 308–311, 2006.

[43] J. A. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," Tech. Rep. ICSI-TR-97-021, University of Berkeley, 1997.

[44] D. Reynolds and T. Quatieri, "Speaker verification using adapted gaussian mixture models," *Digital Signal Process.*, vol. 10, no. 1, pp. 19–40, 2000.

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

[46] 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.

[47] I. Goodfellow, H. Lee, Q. V. Le, A. Saxe, and A. Y. Ng, "Measuring invariances in deep networks," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds., pp. 646–654. Curran Associates, Inc., 2009.

[48] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. Platt, and T. Hoffman, Eds., pp. 153–160. MIT Press, 2007.

[49] E. C. E. G. S. G. S. Spinsante, "Tst fall detection dataset v1," 2016.

[50] B. Kwolek and M. Kepski, "Human fall detection on embedded platform using depth maps and wireless accelerometer," *Computer methods and programs in biomedicine*, vol. 117, no. 3, pp. 489–501, 2014.

[51] I. Charfi, J. Miteran, J. Dubois, M. Atri, and R. Tourki, "Optimized spatio-temporal descriptors for real-time fall detection: comparison of support vector machine and adaboost-based classification," *Journal of Electronic Imaging*, vol. 22, no. 4, pp. 041106, 2013.

[52] A. T. Özdemir and B. Barshan, "Detecting falls with wearable sensors using machine learning techniques," *Sensors*, vol. 14, no. 6, pp. 10691–10708, 2014.

[53] E. Casilari, J.-A. Santoyo-Ramón, and J.-M. Cano-García, "Analysis of public datasets for wearable fall detection systems," *Sensors*, vol. 17, no. 7, 2017.

[54] M. Vacher, S. Bouakaz, M.-E. B. Chaumon, F. Aman, R. A. Khan, S. Bekkadja, F. Portet, E. Guillou, S. Rossato, and B. Lecouteux, "The cirdo corpus: Comprehensive audio/video database of domestic falls of elderly people," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, N. C. C. Chair), K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis, Eds., Paris, France, may 2016, European Language Resources Association (ELRA).

[55] P. Olivetti, "Sistema per la rilevazione e prevenzione di caduta anziani, mediante cassa di risonanza a pavimento," Italian Patent 0001416548, July 1, 2015.

[56] A. Temko, C. Nadeu, D. Macho, R. Malkin, and M. Omologo, "Acoustic Event Detection and Classification," in *Hum. Comput. Interact. XXII, Comput. Hum. Interact. Loop*, W. Waibel and R. Stieflhagen, Eds., chapter 7, pp. 61–73. Springer-Verlag, Berlin, 2009.

[57] 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.

[58] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 28, no. 4, pp. 357–366, 1980.

[59] A. Temko and C. Nadeu, "Classification of acoustic events using SVM-based clustering schemes," *Pattern Recognition*, vol. 39, no. 4, pp. 682–694, Apr. 2006.

[60] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, "The DET curve in assessment of detection task performance," in *Proc. of the European Conference on Speech Communication and Technology*, 1997, pp. 1895–1898.

[61] N. Noury, A. Fleury, P. Rumeau, A. Bourke, G. Laighin, V. Rialle, and J. Lundy, "Fall detection - principles and methods," in *Proc. of the Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Lyon, France, 2007, pp. 1663–1666.

[62] M. Markou and S. Singh, "Novelty detection: a review – part 1: statistical approaches," *Signal processing*, vol. 83, no. 12, pp. 2481–2497, 2003.

[63] M. Markou and S. Singh, "Novelty detection: a review – part 2: neural network based approaches," *Signal processing*, vol. 83, no. 12, pp. 2499–2521, 2003.

[64] M. Popescu and A. Mahnot, "Acoustic fall detection using one-class classifiers," in *Proc. of the Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Minneapolis, MN, USA, 2009, pp. 3505–3508.

[65] E. Principi, S. Squartini, and F. Piazza, "Power Normalized Cepstral Coefficients based supervectors and i-vectors for small vocabulary speech

recognition," in *Proc. of the International Joint Conference on Neural Networks (IJCNN)*, Beijing, China, Jul. 6-11 2014, pp. 3562–3568.

[66] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in neural information processing systems*, 2006, pp. 739–746.

[67] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *International Conference on Machine Learning*, 2013, pp. 115–123.

[68] F. Werner, J. Diermaier, S. Schmid, and P. Panek, "Fall detection with distributed floor-mounted accelerometers: An overview of the development and evaluation of a fall detection system within the project eHome," in *Proc. of the 5th Int. Conf. on Pervasive Computing Technologies for Healthcare and Workshops*, Dublin, Ireland, May 23-26 2011, pp. 354–361.

[69] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6964–6968.

[70] A. Ng, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[71] 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.

[72] 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.

[73] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[74] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks.," in *Aistats*, 2010, vol. 9, pp. 249–256.

[75] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, no. 1, pp. 44–53, 2017.

[76] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[77] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in Neural Information Processing Systems*, 1994, pp. 737–744.

[78] 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.

[79] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, 2015, vol. 2.

[80] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al., "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 3630–3638.

[81] B. Lake, C.-y. Lee, J. Glass, and J. Tenenbaum, "One-shot learning of generative speech concepts," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, 2014, vol. 36.

[82] P. Manocha, R. Badlani, A. Kumar, A. Shah, B. Elizalde, and B. Raj, "Content-based representations of audio using siamese neural networks," *arXiv preprint arXiv:1710.10974*, 2017.

[83] J. F. Gemmeke, L. Vuegen, P. Karsmakers, B. Vanrumste, et al., "An exemplar-based nmf approach to audio event detection," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2013 IEEE Workshop on*. IEEE, 2013, pp. 1–4.

[84] G. Parascandolo, T. Heittola, H. Huttunen, T. Virtanen, et al., "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.

[85] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *Signal Processing Conference, 2010 18th European*. IEEE, 2010, pp. 1267–1271.

[86] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[87] M. D. Smucker, J. Allan, and B. Carterette, "A comparison of statistical significance tests for information retrieval evaluation," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. ACM, 2007, pp. 623–632.

[88] I. Goodfellow, Y. Bengio, and A. Courville, "Autoencoders," in *Deep Learning*, chapter 14, pp. 502–525. MIT Press, 2016, `http://www.deeplearningbook.org`.

[89] E. Principi, F. Vesperini, S. Squartini, and F. Piazza, "Acoustic Novelty Detection with Adversarial Autoencoders," in *Proc. of the International Joint Conference on Neural Networks (IJCNN)*, Anchorage, AK, USA, May 14-19 2017, pp. 3324–3330.

[90] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[91] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder.," in *Proc. of Interspeech*, Lyon, France, Aug. 25-29 2013, pp. 436–440.

[92] C. E. Cella and J. J. Burred, "Advanced sound hybridizations by means of the theory of sound-types," in *International Computer Music Conference*, 2013.

[93] J. J. Burred, "A framework for music analysis/resynthesis based on matrix factorization," in *International Computer Music Conference*, 2014.

[94] J. O. Smith, *Spectral Audio Signal Processing*, Stanford University, CCRMA, 2010, (online book, last viewed 9/3/2011).