# Mixing user-centered and generalized models for Fall Detection

Mirko Fáñez[a], José R. Villar[b,*], Enrique de la Cal[b], Víctor M. González[c], Javier Sedano[a], Samad B. Khojasteh[d]

[a]*Instituto Tecnológico de Castilla y León, Burgos, Spain*
[b]*Computer Science Department, University of Oviedo, Oviedo, Spain*
[c]*Electrical Engineering Department, University of Oviedo, Spain*
[d] *Department of Computer Engineering, Faculty of Technology, Selcuk University, Konya, Turkey*

**Abstract**

Fall detection (FD) using wearable devices has been the focus of many research studies during the last years. Solutions are expected to work autonomously and securely; however, even the commercial products are still far from being reliable in elderly people autonomously everyday life. This research focuses on developing a FD method valid to be deployed on a wearable device with tri-axial accelerometer as sensing unit, and working autonomously without external services. Moreover, each fall type is considered independently and the solution adapts to its current user. The proposal includes three stages: a novel event detection stage followed by a one-class problem classification and a final classifier that labels the anomaly events as Fall or Normal. The process starts gathering data from the current user's wearable provided the data does not include any fall; this data is used to tune the event detection and to train a one-class classifier. Two different methods are tested: a One-Class Support Vector Machine and a classifier based on the centroid of the normal activities' instances. The detected anomaly events are then classified as Fall or Normal; this second classifier is previously obtained from data from other users' past experiences. For this two-class model, two

*Corresponding author

*Email addresses:* mirko@mirkoo.es (Mirko Fáñez), villarjose@uniovi.es (José R. Villar), delacal@uniovi.es (Enrique de la Cal), vmsuarez@uniovi.es (Víctor M. González), javier.sedano@uniovi.es (Javier Sedano), samad.khojasteh@lisansustu.selcuk.edu.tr (Samad B. Khojasteh)

different options are tested as well: a Support Vector Machine and a feed-forward Neural Network. The experimentation stage includes two different published simulated falls data sets. The obtained results show that filtering the detected peaks corresponding to normal activities of the current user helps the two-class classifier for some types of fall events, suggesting to introduce specific one-class filtering per fall type. However, the number of false alarms is still high, and the two-class classifiers have a high variability in their performances according to the user, which still needs further research. The results suggest that it might be interesting to obtain a more accurate two-class classifier using data only from subjects with similar activity levels; the use of online learning might also improve the general performance in the classification stages.

*Keywords:* Fall Detection, Machine Learning, One-class Classifier, Event Detection, Elderly Population

---

## 1. Introduction

Fall Detection (FD) refers to the detection of fall events of human beings while performing their usual Activities of Daily Living (ADL); it might be considered as part of the Human Activity Recognition. FD can be applied in several different fields, being the support for the elderly population among them [1]. It has been reported that one out of three elderly people suffer a fall [2]. Whenever a senior suffers a fall, the faster the assistance the better [3]. Moreover, falls have a direct consequence in the confidence and the ability of living autonomously of the subject.

Consequently, FD is a topic that has been researched for more than a decade. There are commercial devices using different technology, such as yell detection [4], video and radar [4, 5] or smart tiles [6]. However, either the false alarm rate is high or they are severely constrained (restricted to specific parts of the home or not allowing to have pets). There are also wearable devices, either personal emergency response systems (PERS) [7] or smart watches [8]. Nevertheless, PERS are found useless (high false alarm rate, the need of the person to be conscious), or their use undesirable. In the case of smart watches, they are only designed to detect "high impact falls", but the performance is still poor [9, 10].

Concerning the scientific literature, the different studies also focus on different technologies: surveillance or 3D range cameras [11, 12, 13, 14],

motion sensors [15, 16] or floor and sound sensory systems [17, 18, 19]. As before, the main problem with these solutions are that they are constrained to specific rooms in a home; therefore, there is no FD whenever the user is out of coverage. Also, some solutions introduce more constrains in the number of people in a room, or pets, etc.

This study is focused on autonomous FD using wearable devices: this type of devices allows to continuously monitor the activity of the user in their everyday life, they are both ergonomic and appealing while at the same time the privacy is not in compromise. There is a wide literature on the topic of FD (interested readers might find worthy reviews such as [2, 20, 21, 22, 23]); the majority of the studies make use of tri-axial accelerometers (3DACC) [24, 25, 26, 27, 28], although different types of sensors have been also used: barometer [29, 30], gyroscope [31, 32, 29, 30, 33], electromyography [34] or angle [35]. Some of the studies also perform a fusion of several sensors [36, 31, 37, 29, 30].

In this research, we propose a FD approach using a single 3DACC placed on a wrist, which is easier to conceal as a smart watch or an intelligent bracelet. The main contributions of this study include an enhancement in the event detection stage and the collaboration between user-centred and generalized models in order to correctly classify a peak candidate. The aim is to produce an autonomous solution that does not rely on external services. In this study, we propose an approach that makes use of both user centered and generalized models: the former is devoted to detect anomalies in the user behaviour, while the latter considers the previous a priory knowledge extracted from the available fall data sets in order to classify these anomalies. To our knowledge, this is the first solution that mixes both types of modelling, which takes the best of both. The solution includes three stages: a novel event detection, an anomaly user centered classification and a final generalized classifier to discriminate the anomalies. The solution is evaluated using two different publicly available simulated falls data sets, the experimentation is exhaustively detailed. The results show that filtering those peaks generated from the peak detection helps the two-class classifier for the forward fall type. Furthermore, the event detection designed in this research shows a robust performance for different types of falls.

The structure of this study is as follows. The next section details the related work. Sect. 3 explains all the details of this study: the data sets, the modelling techniques and the experimentation stages. The obtained results from the experimentation are clearly shown and discussed in Sect. 4, extract-

ing the main ideas and consequences from the performance measurements. Finally, the study draws the conclusions.

## 2. Related work

One of the first 3DACC researches concerning FD was proposed in [38]. The authors proposed a 3DACC sensor placed on a belt of participants (some of them were elderly people) while performing several ADLs; they also used a dummy to simulate the falls. A One Class Support Vector Machine (OSVM) model classifies the time window as normal, signaling the remaining cases as fall alarms. The series of studies of Bourke et al started with two 3DACC (one placed on the chest and one on the thigh) and simple detection methods based on thresholds [39, 40]. Recently, the authors [31] analyzed the real fall data set from patients of Parkinson [41], where the patients wore a 3DACC plus a gyroscope on either the waist or the thigh. The event detection was performed through a threshold and several features were calculated for each detected event. The generated data set was balanced using SMOTE and a C4.5 decision tree classifier was proposed.

Several different measurements of the fall dynamics were used in [42] with data gathered from a 3DACC on the waist (or head). Thresholding was the decision making algorithm to decide whether the current instance corresponds or not with a fall event. Using this solution, the authors compared the dynamic of real and simulated fall events [43].

Hidden Markov Models have been also proposed to continuously analyze the acceleration values [27, 28] labelling the samples correspondingly as fall or normal. Besides, [24] proposed a 3DACC placed on a wrist while continuously analyzing the acceleration values using either a Support Vector Machine (SVM), a Neural Network (NN) or K-Nearest Neighbour (KNN).

Moreover, 3DACC has also been combined with barometric sensors to detect fall events in [36]; in this case, the sensor was placed on the waist. An heuristic set of rules and thresholds were proposed to determine whether there is a fall or not. Besides, Sorvala et al ([33]) combined 3DACC on the waist and a gyroscope on the ankle, using the magnitude of the acceleration and the angular velocity together with an heuristic algorithm based on thresholds to classify the signals. In addition, 3DACC was also combined with gyroscope in [44], where a study of the performance of several threshold based FD methods were analyzed when running on a Smartwatch and on a Smartphone. The threshold based methods were also used to change the current state, similar

to a Finine State Machine (FSM). The same combination of sensors but placed on the chest are used in [32] for FD. The decision was based on three thresholds: if a small acceleration magnitude is followed by a high acceleration magnitude and a high angular velocity, then a fall alarm is fired.

Furthermore, [30] combined 3DACC together with gyroscope and barometer, placing the sensory system on the waist. Thresholds on the vertical speed were used to detect the events: whenever an increase in this signal is observed, up to 7 different combinations of the acceleration, posture and height are surveyed; if any of these combinations is higher than the corresponding threshold, an alarm is signaled. [29] combined 3DACC and barometer sensors on a device; this device was placed on a wrist. In this study, the acceleration magnitude drives the peak event detection. Three features were computed for each detected peak using a 6 second length pressure window centered on the peak. Then, the features are classified using a SVM.

Using a single 3DACC on the waist, the study in [45] proposed a FD system based on an event detection plus a classifier. The event detection stage was a peak detection based on a FSM and predefined thresholds; then, a feature extraction is performed on the time slice surrounding the detected peaks. Finally, a NN is used to classify each instance of 8 transformations. Previously, the authors performed an in-depth analysis of the falls and their dynamics, taxonomy and causes [46].

The solution of Abbate et al in [45] was extended in [47] and [48]. In the former, kNN was used instead of NN. In the study in [48], the approach was adapted to be used with the sensor on a wrist, several features were revised and, finally, different models were evaluated (NN, SVN, kNN, Decision Trees -DT- and Rule Base Systems -RBS-). The same research team analyzed the use of kNN with a reduced data set including selected instances from clustering [49]. In [50], the authors proposed a one-class SAX-based dictionary to learn the user behavior; this dictionary was developed for each specific user considering only the ADLs.

FSM were also used as event detection in [51]; whenever a peak was detected, the surrounding window was analyzed and several transformations were computed. The classification of these features was performed using a classification and regression tree, a kNN, LR and a SVM. Instead of a FSM, [52] proposed to detect high peaks, low peaks and the time between a sequence of a high and low consecutive peaks. They developed an Android Wear app to use the 3DACC measurements from a Smartwatch.

An FSM is also proposed to detect the fall events if the subject does not

move after the fall [52]. Thresholds of the acceleration and the angle of the gravity are used together with the time in each state to drive the FSM. The 3DACC sensor is located on the waist in this study. Similarly, thresholds are used to detect fall events in [53]; if a fall event is followed by a 20 seconds calm period (that is, with a reduced amount of movement), then the fall alarm is signaled. Thresholds were also used for FD in [54], the sensor was the 3DACC signal from a Smartphone.

A comparison of several published simulated falls data sets presented in [55] used a threshold on the acceleration magnitude to detect fall candidates; afterwards, 6 seconds windows are classified using either SVM or NN. NN have been used in [56] too. In this study, a 3DACC was located on a writs and three different NN models were obtained: i) using 3 seconds of acceleration magnitude windows as inputs of the NN, ii) the acceleration magnitude peak and the times of the fall event as the input features of the NN and, iii) these three features plus the mean and deviation were the inputs of the NN. In all the cases, Multi-layer perceptrons were proposed.

Furthermore, Medrano et al [57] analyzed the 3DACC signals from Smatphones placed inside the frontal pockets reporting the use of three different models (kNN, SVM and OSVM). Thresholds were used to detect peaks; the three acceleration components' values within a 6 seconds length sliding window centered on the peak were used as the inputs of the classification models. Similarly, Ngu et al proposed to classify 250 milliseconds windows of the acceleration magnitude; these windows were transformed into a 4 dimensional vector and considered the inputs of the two modeling techniques (SVM and kNN) [24].

Finally, Deep Learning is currently being employed in FD, although developing such models on wearable devices will need more powerful Smartwatches than the ones in the market nowadays. Nevertheless, the study in [58] proposed to pair the Smartwatch to a Smartphone, which is the responsible of running the Recurrent Neural Network. Autoencoders have been proposed as a feature extraction technique before a classification stage based on SVM [59]. For a more in-depth review on this topic, please refer to [23]. However, because nowadays Deep Learning is not feasible to be deployed on wearable devices such as Smartwatches (as stated in [58]), we do not develop on this type of solutions further.

Concerning the experimentation, the studies make use of their own data set or evaluate their solutions using several published data sets of simulated falls. In some cases, extra data sets are used in order to evaluate the approach

with different ADL coming from different sources and participants. Almost all the solutions propose generalized models (that is, learning a model that cope with the complete population), either for all type of falls or for specific type of falls. Only the studies in [44, 57, 50] propose user-centred solutions. In this case, a model is learned for each specific user, allowing the model to adapt to the variability of behaviours. Nevertheless, the problem is still a challenge due to the lack of robustness of the outcome of the solutions when evaluated with different data sets [50].

## 3. Materials and methods

This research is focused on training a set of fall-detection classifiers using data gathered from participants' ADLs. This is a three-stage process: a fall-event detection, a one-class classifier to mark those events that differs from the normal behaviour of the user, and a final classifier that labels each anomaly event as FALL (F) or NOT FALL (NF). An overview of the solution is depicted in Fig. 1. The proposal is instantiated for each type of fall to be identified; therefore, if we consider two types of falls then the structure in the figure is duplicated, one for each type of fall.

Interestingly, this approach includes a mixture of user centered and generalized solutions. We refer to user centered solutions those that develop specific models for the current user, aiming to adapt to the particularities of his/her behaviour. Conversely, generalized solutions develop a general model for the whole population, assuming the problem presents similar implications for all the subjects.

It is worth mentioning that because we are designing a solution that can be deployed on wearable devices, the modelling techniques should keep a low computing demanding profile. In other words, current wearable devices cannot deploy Deep Learning models because this type of models would eventually drain the battery in a short period of time.

Thus, the one-class classifier models the behavior of the current user, gathering data from an unsupervised training stage that the user must perform previous to the exploitation of the solution. On the other hand, the multi-class classifier is obtained from historical data of ADLs and fall events that would have been gathered from a sampling population.

Therefore, the solution includes gathering data from the current user and exploiting data from a sampling population. To evaluate the solution, we have included several publicly available simulated fall data sets. The

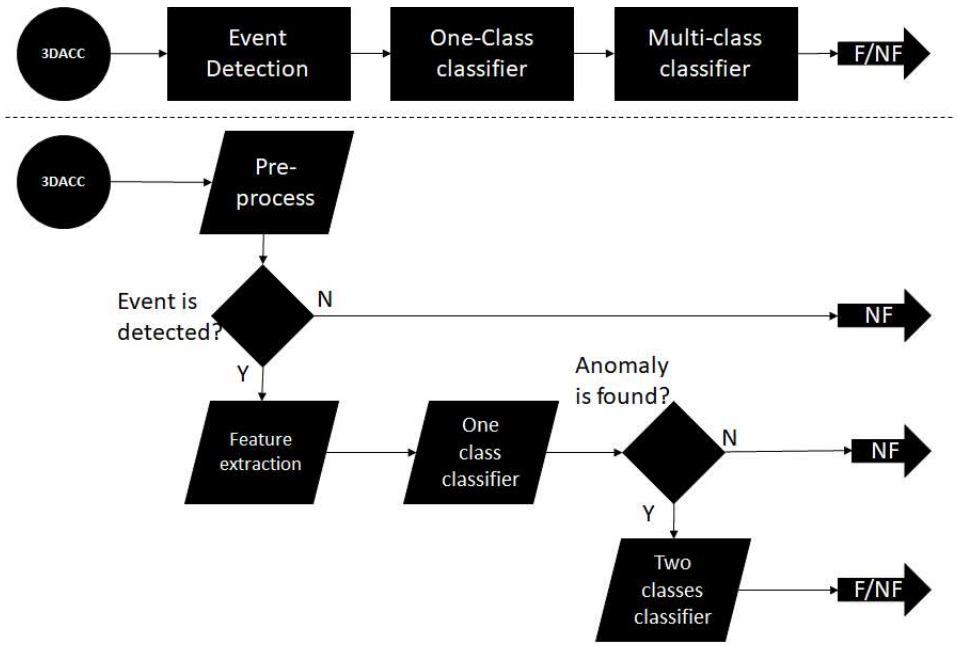


Figure 1: Simplified block diagram of the solution. The samples are analyzed to detect the events from a suspicious behaviour. Several features are extracted from the window of data and a one-class SVM determines whether it is normal or not. In case of abnormal scenario, a second classifier labels the event as F or NF.

solution is evaluated independently for each of them, introducing a complex experiment design to allow developing both the user centered and generalized parts.

In the next subsections we will be covering all of these issues. Firstly we will start with the description of the data sets used for in this research, then each of the main stages of the solution will be detailed (in this order, the event detection, and pre-processing, the features extraction, the one-class classifier and the multi-class classifier). Finally, the experimental design will be completely explained.

### *3.1. Publicly available data sets*

Two publicly available simulated falls data sets are used in this experimentation. Each of these data sets includes data from several participants wearing, at least, a 3DACC sensor placed on a wrist. However, all of them include data from more than one sensor. Each participant performed several repetitions from a certain catalog of ADLs and fall types. Each of the repetitions is stored as a multivariate TS with the three acceleration axis, along with the corresponding label of the ADL or fall type.

Unfortunately, in the publicly available data sets there is a lack "of a common experimental bench-marking procedure and, consequently, the large heterogeneity of the data sets from a number of perspectives (length and number of samples, typology of the emulated falls and ADLs, characteristics of the test subjects, features and positions of the sensors, etc.)" [60]. Therefore, in this study we will manage each data set independently.

The data sets that are going to be considered are:

**UMAFall data set [61]** : including TS from up to 17 participants with 3DACC placed on the wrist, on the waist and on the ankle; the sampling frequency is 20 Hz. Each participant performed a set of ADL (such as running, walking, siting, hopping, etc.) and staged three types of falls (lateral, forward and backwards falls); however, not all the participants did the same number of repetitions (even several participants did not stage falls). There is a total of 531 TS (208 of them are labelled with one of the possible fall types). We have only considered those participants with more than 20 TS, provided that at least 9 of them are simulated falls. Although participant 17 did not performed any backward fall, we kept it due to the number of falls of other types.

**Özdemir&Barshan [62]** : includes 17 participants with 3DACC placed on the wrist, on the waist and on the ankle; the sampling frequency is 25 Hz. There is a catalog of ADL (running, walking, hopping, etc.) and a catalog of fall types (up to 20 different fall types, forward, backwards and lateral falls among them). There is up to 3060 TS (1700 of them labelled as FALL). Each participant performed 5 runs on average of every ADL and FALL simulation.

There are more simulated falls data sets published in the community [60]; however, many of them do not include sensors on a wrist. Moreover, some of the data sets that include sensors on the wrist introduced protocols for several activities that were different than normal behaviour. For instance, the TST data set [63] includes walking patterns that are peculiar (the patterns are really different from those of walking measured with a 3DACC on a wrist); as it is going to be stated afterwards, the walking activity was chosen as the standard activity for computing thresholds and TS normalization. Therefore, we discarded this data set as part of the test bed.

From now on, we are going to focus on three types of falls: those that are common to all the data sets. Therefore, only forward, backward and lateral falls, when the participant is standing still upright, are considered.

### *3.2. Event detection and pre-processing*

As a consequence of all the previous reasoning, we propose to use an event detection method per fall type. Evidently, the event detection method is highly dependent of the context, so it is not possible to have a general event detection method able to cope with high intensity falls (say, walking and then falling) and at the same time low intensity falls (say, seated and then fainting).

In this study we have decided to use the peak detection method detailed in [45] and slightly modified in [48] to detect the high intensity events for each of the considered type of fall. Nevertheless, some modifications are introduced.

The event detection is a very simple finite state machine that tries to find acceleration peaks to detect the high intensity falls (refer to Fig. 2). Let us assume that gravity be $g = 9.8m/s$. Let also assume that we use the acceleration magnitude $a_t$ (see Eq. 1), where $a_{tx}$, $a_{ty}$ and $a_{tz}$ are the components of the acceleration in each axis at time $t$.

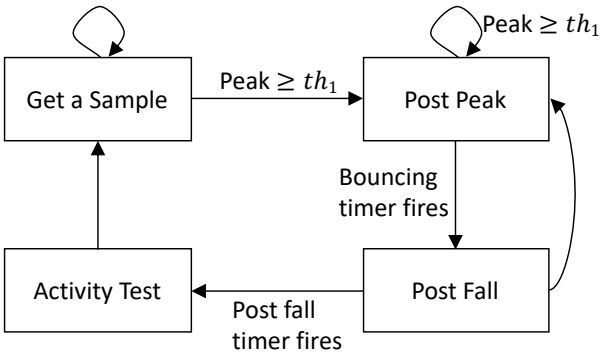


Figure 2: State Machine proposed in [45]. At time $t$, a peak is detected in $t_p = t - 2500ms$ if $acc(t_p)$ is the latest value higher than $th_1$ and no acceleration value is higher than $th_1$ in the period $(t_p, t]$.

$$a_t = \sqrt{a_{tx}^2 + a_{ty}^2 + a_{tz}^2} \tag{1}$$

A peak is defined as the time stamp for which the acceleration is higher than $th_1$, but there is no acceleration value higher than $th_1$ during the next following period of 2500 milliseconds (ms), with $th_1$ being a predefined peak threshold (see Abbate et al, [45]). A second threshold $th_2$ was predefined and used to obtain the limits of the *fall window* (the window of acceleration values surrounding the peak time). In their original study, Abbate et al set these threshold values $th_1$ and $th_2$ to $3.0 \times g$ and $1.5 \times g$, respectively.

We propose to smooth the 3DACC TS by computing the mean value of a sliding window of size $\frac{1}{4}$s to filter the signal. Furthermore, we do not use the Abbate et al method for determining the fall window. Instead, we propose to directly use the limits stated in Fig. 3, setting the fall window as the interval $[t_p - 200, t_p + 1000]$ ms. Whenever a peak is detected, the data from the fall window is used to compute several features. Consequently, in this research we do not make use of $th_2$ anymore, simplifying the threshold setting.

Moreover, the thresholds proposed in the original work have been studied so far in previous work [50, 49]. As a conclusion, these values must be specific for each user and fall type. Also, they must be automatically determined according to the intensity of the user's activity. We decided to use the walking activity of a subject as the standard activity to determine these thresholds. To do so, the user must walk during a certain predefined time $TrainTime$, which must be defined according to the specific population that is being

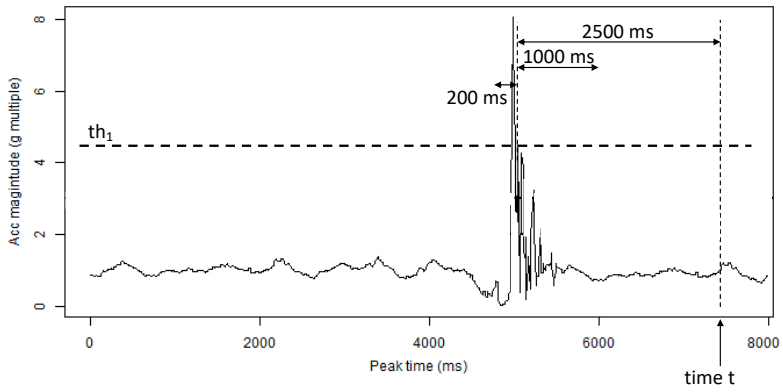


Figure 3: Dynamic of a fall event. A peak due to a fall is determined as an acceleration value higher or equal than $th_1$ followed by a relatively calm period of time. The fall window is the period surrounding the peak time at $t - 2500$ ms.

studied. The $th_1$ threshold is computed as stated in Eq. 2, where $maxWalk$ is the mean value of the set of the 1000 highest 3DACC values while walking.

$$th_{peak} = 0.9 \times maxWalk \tag{2}$$

Finally, all the samples in a TS must be z-scored; again, the walking activity is proposed as the standard. The mean and the deviation ($wkmean$ and $wkstd$) of the acceleration magnitude computed when carrying on with this ADL will be used to perform these pre-processing tasks on the TS for a user, independently of the type of fall. Obviously, the thresholds must be normalized as well.

### *3.3. Feature extraction*

The feature extraction is executed whenever a peak is detected and tries to represent the information related with the dynamics within a fall. Given the current timestamp $t$, we find a peak at **peak time** $pt = t - 2500ms$ if at time $pt$ the magnitude of the acceleration $a_t$ -see Equation 1- is higher than $th_1$ and there is no other peak in the period $(t - 2500ms, t]$ (no other $a$ value higher than $th_1$). If this condition holds, then it is stated that a peak occurred at $pt$. Finding a peak also induces the fall window, as explained before. The **impact end** ($ie$) denotes the end of the fall event; it is fixed to $pt + 1000$ ms. The **impact start** ($is$) denotes the starting time of the fall event, which is equal to $pt - 200$ ms.

With these three times -is, pt, ie- calculated, the following transformations should be computed:

- Average Absolute Acceleration Magnitude Variation (AAMV) calculated as stated in Eq. 3, where $N$ is the number of samples in the interval.

$$AAMV = \sum_{t=is}^{ie} \frac{|a_{t+1} - a_t|}{N} \tag{3}$$

- Maximum Peak Index (MPI), measuring the maximum in the fall window, Eq. 4.

$$MPI = max_{t\in[is,ie]}(a_t) \tag{4}$$

- Minimum Valley Index (MVI), measuring the minimum in the fall window, Eq. 5.

$$MVI = min_{t\in[is-500,ie]}(a_t) \tag{5}$$

- Pre-impact Activity Index (PrAI), measuring the AAMV in the part of the fall window before the peak, Eq. 6.

$$PrAI = \sum_{t=is}^{pt} \frac{|a_{t+1} - a_t|}{N} \tag{6}$$

- Post-impact Activity Index, $PoAI$, measuring the AAMV in the part of the fall window after the peak, Eq. 7.

$$PoAI = \sum_{t=pt}^{pt+500} \frac{|a_{t+1} - a_t|}{N} \tag{7}$$

- Long-term Activity Index, $LAI$, measuring the AAMV in the interval where the user must be more or less still after a fall, Eq. 8.

$$LAI = \sum_{t=pt+500}^{ie} \frac{|a_{t+1} - a_t|}{N} \tag{8}$$

- Free Fall Index (FFI), the average magnitude in the interval $[t_{FFI}, pt]$. The value of $t_{FFI}$ is the time between the first acceleration magnitude below $th_{FFI} = 0.8 \times g$ occurring up to 200 ms before $pt$; if not found, it is set to $pt - 200\ ms$.

- Step Count Index (SCI), that counts how many steps the user hypothetically has done before a fall. To compute SCI we use the number of the peaks that are followed by a valley in the interval $[pt - 2200, pt]$.

This set of features differs from those proposed in [45] and used in our previous research [48, 64, 50, 49]. The change is due to several reasons. On the one hand, fixing the fall window limits instead of dynamically determining them made several of the original features meaningless. On the other hand, the original set of features introduced a big number of thresholds; the new features have considerably reduced this amount.

### *3.4. The one-class classifier*

This model is responsible of identifying normal or common values of the features that have been computed for a fall window. Moreover, this model is specific for the current user as it is expected to get data from normal user's behaviour in order to learn the classifier. Therefore, the user must perform a training stage using the sensor during his/her everyday life; the detected peaks will help in learning the classifier.

For sure, the training of the model is not expected to be done in the wearable device: this task must be performed on the cloud (through web services) or on the edge (specific near the edge computing resources) or even on the fog (if we consider the user's smart phone as a candidate). Nevertheless, the study of where to perform the training of the models is out of the scope of this research.

Two different solutions are tested in this study as the one-class classifier: a One-class Support Vector Machine (OCSVM) and a very simple classifier using the centroid of the instances and the mean distance (here in after, we refer to this classifier as CENTROID).

OCSVM is used to identify the normal behavior; we have selected it for several reasons. Firstly, because it is a very simple model that can be easily deployed in a wearable. Secondly because it has been successfully applied in anomaly detection [38, 65]. The latter of these two studies use OCSVM to detect anomalies in sound recordings, while the former is highly related with this work. The authors proposed gathering data from a 3DACC, low pass filtering the acceleration and then using a SVM to detect anomalies; these anomalies are then classified using a K-Nearest Neighbour (KNN) and a Kernel Fisher Discriminant model. Nevertheless, the main drawback of this approach is that KNN models tend to drain the battery, thus is not

suitable unless the number of samples is kept reduced. Moreover, the use of the acceleration magnitude directly may also introduce a high ratio of errors. Finally, the OCSVM is a generalized approach, which also might induce an increment in the false alarm ratio.

To overpass these drawbacks, the current OCSVM only analyzes the features extracted from a fall window. We consider one OCSVM per user and type of fall; therefore, the variability in the intensity of the movements from one user to another gets reduced. Finally, the design of this solution avoids analyzing every single sample, only those that are identified as a peak are analyzed. It is worth mentioning that all the instances of features computed for each peak should be also z-scored previous to the training of the OCSVM. The mean and standard deviation used in this z-score will be later on used to z-score the new instances to classify as normal or anomalies; we call them *ocmean* and *ocstd*, respectively, and are specific for each user and fall type. In this research, we have chosen the OCSVM implemented in [66, 67] and described in [68].

The CENTROID classifier is a very simple solution. Its gathers the detected peaks from all the ADL's TS for the current user, compute the centroid and the mean distance from the centroid to each instance. An incoming instance is classified as an anomaly if it is out of the hypersphere centered in the centroid and radius the mean distance. All the previous discussion concerning the z-scoring and the calculation of *ocmean* and *ocstd* still holds, so these values are stored for later use.

### *3.5. The multi-class classifier*

The features extracted from a fall window due to the detection of a peak, which have been labelled as anomaliy (using either the OCSVM or the CENTROID) are then analyzed in a second classifier. In this case, we propose to use all the knowledge extracted from past experiences in the form of data from other users, so a two-class model can be used to label new samples. These samples will later be classified as Fall (F) or Not Fall (NF), correspondingly.

Let us assume the existence of a data set with TS recorded from different users performing ADLs but also with real falls. For all these TS we apply the same event detection and features extraction as explained in the previous sub-sections (including the threshold automatic learning and the scaling and standardization of the data); in doing so, we obtain a new data set with the features extracted from the fall window of each of the detected peaks, a label

is also given to the data as we a priory know whether each peak comes from a fall event or not.

With this new data set we learn a two-class classifier. Actually, we test two different solutions: a Two-Class Support Vector Machine (TCSVM) [66, 67] and a feed-forward Neural Network (TCNN) [66, 69]. The use of SVM and NN for this kind of task has been reported profusely in the literature [45, 54, 55, 29, 48, 25, 49]. However, the main difference is that this solution is specific for each type of fall, and that it is intended to classify samples that have been already detected as anomalies. Again, as before, the instances must be z-scored, the values of the mean and standard deviation are stored for their use as *tcmean* and *tcstd*, respectively.

### *3.6. Experimentation setup*

The following rules apply in the experimentation design:

- The experiments are instantiated independently for each data set. Therefore, we do not mix the data from users belonging to different data sets.
- The experiments are instantiated independently for each user and fall type.

The experimentation is split in two main parts: the first one is devoted to learning the generalized TCSVM/TCNN model, while the second aims to evaluate the complete solution. For the first stage, called the *Two-Class experimentation*, we are going to measure the performance of the TCSVM and TCNN using 10-fold cross validation. For the second stage, called the *Complete Solution Experimentation*, we are going to measure the performance of the OCSVM, the CENTROID and also the performance of the complete solution.

The description of the two stages is included next. For the sake of comprehension, these descriptions are presented as algorithms because we do think it is the best way to explain how to evaluate each part of the proposal, and to also contribute to the explanation of the whole solution. In all the cases, the performance of the methods is measured using the Sensitivity and Specificity.

The *Two-class experimentation* is divided in three main parts: i) initialization of data sets, peak thresholds, and walking statistics, ii) evaluation of a peak detection method and iii) two-class classifier learning and evaluation.

**Result:** $TCdataset^{f}_{d,u}$, $th_1^{d,u}$, $walkmean_{d,u}$, $walkstd_{d,u}$ $\forall d, f, u$
**for** *each data set* $d$ **do**
  **for** *each fall type* $f$ **do**
    **for** *each user* $u$ **do**
      $TCdataset^{f}_{d,u} \leftarrow \Phi$;
      temp $\leftarrow$ Retrieve all the walking $TS_u^{walking}$ for user u;
      Compute $walkmean_u$ and $walkstd_u$ for temp;
      Compute $th_1^u$ follownig Eq. 2;
    **end**
  **end**
**end**

**Algorithm 1: Initialization of data sets, peak thresholds, and walking statistics**. $TCdataset^{f}_{d,u}$ is the data to train and test the Two-Class classifiers, $th_1^u$ is the threshold used in peak detection, $walkmean_u$ and $walkstd_u$ are the mean and standard deviation measured on the walking TS.

Each of these parts is described in Algorithms from 1 to 3, correspondingly. Stages ii) and iii) will produce results to be shown in the results section.

The second experimentation part (the *Complete Solution Experimentation*) evaluates how the complete method performs for each of the participants included in the data sets. The proposed solution is compared with a base-line method: the on-wrist Abbate et al method proposed in [45] and adapted to the wrist in [64]. Again, the explanations of this experimentation are presented in three algorithms: Algorithm 4 split the TS from a participant in train and test, Algorithm 5 is devoted to learn the anomaly detector (the OCSVM or the CENTROIDS) and Algorithm 6 evaluates the performance of the anomaly detection together with the best Two-Class model found so far. The two latter algorithms produce numerical outcome that is going to be shown in the results section.

## 4. Results and discussion

The results are grouped according to the defined experimentation; however, and for the sake of clarity, three subsections are included to show these results. The next subsection deals with the two-class model learning, including the performance of the TCSVM and TCNN. The results from the

**Result:** $TCdataset^{f}_{d,u}$, $tcmean_u$, $tcstd_u$, $TP^{f}_{d,u}$, $TN^{f}_{d,u}$, $FP^{f}_{d,u}$ and $FN^{f}_{d,u}$ $\forall d, f, u$

```
for each data set d do
    for each fall type f do
        for each user u do
            tempDTS^f_{d,u} ← Φ;
            for each TS_{d,u} do
                for each detected peak in TS_{d,u} do
                    Determine the fall window;
                    Compute the feature set;
                    Add instance to tempDTS^f_{d,u};
                end
                Update the TP^f_{d,u}, TN^f_{d,u}, FP^f_{d,u} and FN^f_{d,u} counters;
            end
            Compute tcmean_u and tcstd_u from tempDTS^f_{d,u};
            tempDTS^f_{d,u} ←z-score(tempDTS^f_{d,u}, tc − mean_u, tc − std_u
             );
            Add tempDTS^f_{d,u} to TCdataset^f_{d,u};
        end
    end
end
```

**Algorithm 2: Evaluation of the peak detection method**. This algorithm updates $TCdataset^{f}_{d,u}$ data set, computes the tcmean and tcstd for each tuple $< d, f, u >$ and evaluates the peak detection method for each fall type by updating the TP, TN, FP and FN counters.

**Result:** $TCMODEL^f_{d,u}$, $TCPARAMS^f_{d,u}$, $RESULTS^f_{d,u}$ $\forall d, f, u$
**for** *each data set* $d$ **do**
  **for** *each fall type* $f$ **do**
    **for** *each user* $u$ **do**
      Retrieve $TCdataset^f_{d,U}$ $\forall U! = u$;
      Perform 10-fold cv to obtain the best $TCPARAM^f_{d,u}$;
      $RESULTS^f_{d,u} \leftarrow$ results of best model;
      Train $TCMODEL^f_{d,u}$ with $TCPARAM^f_{d,u}$ and the complete data set;
    **end**
  **end**
**end**

**Algorithm 3: Two-class classifier learning and evaluation**. The Two-Class Model ($TCMODEL^f_{d,u}$) will be used later in the second part of the experimentation. $TCPARAM^f_{d,u}$ and $RESULTS^f_{d,u}$ will be shown in the results section.

**Result:** $train^f_{d,u}$, $test^f_{d,u}$ $\forall d, f, u$
**for** *each data set* $d$ **do**
  **for** *each fall type* $f$ **do**
    **for** *each user* $u$ **do**
      **begin** Split all the $TS^f_{d,u}$ in two: train and test
        $test^f_{d,u}$ includes all the falls plus 10% of ADL;
        $train^f_{d,u}$ includes the remaining TS;
      **end**
    **end**
  **end**
**end**

**Algorithm 4: The Complete Solution Experimentation: splitting the TS from a participant in train and test**.

**Result:** $OCMODELd, u^f$, $OCPARAM^f_{d,u}$, $ocmmean^f_{d,u}$, $ocmstd^f_{d,u}$, $RESULTS^f_{d,u}$ $\forall d, f, u$

```
for each data set d do
    for each fall type f do
        for each user u do
            Retrieve th_1, walkmean_{d,u}, walkstd_{d,u};
            Retrieve train^f_{d,u};
            for each TS in train do
                ocDTS ← Φ;
                zscore TS with walkmean_{d,u} and walkstd_{d,u};
                for each detected peak in TS do
                    Determine the fall window;
                    Compute the feature set;
                    Add instance to ocDTS;
                end
            end
            Compute ocmean^f_{d,u} and ocstd^f_{d,u} from ocDTS;
            ocDTS ← zscore(ocDTS, ocmean^f_{d,u}, ocstd^f_{d,u});
            (OCPARAMS^f_{d,u}, RESULTS^f_{d,u}) ← Find the best
             parameter set for the one-class model using 10-fold cv on
             ocDTS;
            Train the OCMODEL^f_{d,u} with the OCPARAMS^f_{d,u} and
             ocDTS;
        end
    end
end
```

**Algorithm 5: The Complete Solution Experimentation: learning the anomaly detection method**. The One-Class model $OCMODEL^f_{d,u}$ will be used in the next algorithm, as well as $ocmean^f_{d,u}$ and $ocstd^f_{d,u}$. $OCPARAMS^f_{d,u}$ and $RESULTS^f_{d,u}$ will be depicted in the results section.

**Result:** $RESULTS^{f}_{d,u}\ \forall d, f, u$
**for** *each data set* $d$ **do**
  **for** *each fall type* $f$ **do**
    **for** *each user* $u$ **do**
      Retrieve $th_1$, $walkmean_{d,u}$, $walkstd_{d,u}$;
      Retrieve $OCMODEL^{f}_{d,u}$, $ocmean^{f}_{d,u}$, $ocstd^{f}_{d,u}$;
      Retrieve $TCMODEL^{f}_{d,u}$, $tcmean^{f}_{d,u}$, $tcstd^{f}_{d,u}$;
      Retrieve $test^{f}_{d,u}$;
      **for** *each TS in test* **do**
        zscore TS with $walkmean_{d,u}$ and $walkstd_{d,u}$;
        **for** *each detected peak in TS* **do**
          Determine the fall window;
          Compute the feature set $\rightarrow features$;
          $oc_feats \leftarrow$zscore $features$ with $ocmean^{f}_{d,u}$ and $ocstd^{f}_{d,u}$ ;
          **if** $OCMODEL^{f}_{d,u}(oc\_feats)$ *is anormaly* **then**
            $tc\_feats \leftarrow$ zscore $features$ with $tcmean^{f}_{d,u}$ and $tcstd^{f}_{d,u}$;
            Update $RESULTS^{f}_{d,u}$ with $TCMODEL^{f}_{d,f}(tc\_feats)$;
          **end**
        **end**
        Update $RESULTS^{f}_{d,u}$ when no alarm is found for TS;
      **end**
    **end**
  **end**
**end**

**Algorithm 6: The Complete Solution Experimentation: evaluation of the complete solution**. This stage includes the event detection, the anomaly detection and the Two-Class classification steps for each user. The $RESULTS^{f}_{d,u}$ will be shown in the results section.

solution experimentation are split in two, the first one showing the results from the learning of the one-class models (Subsect. 4.2), while the second one is devoted to depicting the performance of the complete solution (Subsect. 4.3). Each subsection includes the discussion on the results.

### *4.1. Two-class experimentation's results*

For the sake of readability, only the forward fall results are included in this section (Table 1 for the TCSVM and Table 2 for the TCNN); the results for the other types of fall are available under request for interested readers. These tables show the best parameter subset for each participant, the mean and standard deviation of the statistical measurements ROC, Sensitivity (Sens) and the Specificity (Spec) for each data set and participant. They also include the global mean, median and standard deviation among the participants for each data set.

In general, the TCNN shows better performance than the TCSVM, suggesting that the extracted features are not clearly divisible by boundaries. Moreover, the performance of the models when working with participants from the Özdemir & Barshan data set are better than when using the UMAFall. This comparison is interesting, suggesting that the Özdemir & Barshan's participants ran their ADLs and fall simulations systematically, while the UMAFall's participants were free to perform the ADLs and simulations more freely.

Unfortunately, the experimentation results can not be directly compared to those obtained in other research studies because of their considerably different experimentation design; however, what is remarkable is that the models are very robust independently of the participant within each data set, especially for the TCNN. This means that the two models performed similarly for all the individuals, with a highly small value of the standard deviation.

### *4.2. Performance of the one-class classifiers*

This section shows the results from the learning of the one-class models for each participant and data set. Due to the poor OCSVM performance,

Table 1: Forward Falls results for the TCSVM experimentation. The acronym ID stands for participant ID.

| **UMAFall data set** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **ID** | **sigma** | **C** | **ROC** | | **Sens** | | **Spec** | |
| | | | **mean** | **std** | **mean** | **std** | **mean** | **std** |
| 1 | 0.650 | 3.25 | 0.8190 | 0.1602 | 0.7350 | 0.2212 | 0.7700 | 0.2111 |
| 2 | 0.650 | 3.00 | 0.8718 | 0.0913 | 0.7200 | 0.1932 | 0.8650 | 0.1180 |
| 3 | 0.650 | 3.00 | 0.8460 | 0.1423 | 0.7150 | 0.1857 | 0.8300 | 0.1767 |
| 4 | 0.700 | 4.50 | 0.8417 | 0.1316 | 0.7350 | 0.2310 | 0.8067 | 0.2443 |
| 9 | 0.800 | 3.25 | 0.8700 | 0.1436 | 0.8400 | 0.2066 | 0.7600 | 0.2196 |
| 16 | 0.775 | 3.00 | 0.9222 | 0.1213 | 0.8000 | 0.1721 | 0.8250 | 0.1942 |
| 17 | 0.800 | 3.00 | 0.8692 | 0.1216 | 0.8950 | 0.1739 | 0.7550 | 0.2061 |
| **mean** | | | **0.8628** | | **0.7771** | | **0.8017** | |
| **median** | | | **0.8692** | | **0.7350** | | **0.8067** | |
| **std** | | | **0.0325** | | **0.0696** | | **0.0414** | |

| **Özdemir & Barshan data set** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **ID** | **sigma** | **C** | **ROC** | | **Sens** | | **Spec** | |
| | | | **mean** | **std** | **mean** | **std** | **mean** | **std** |
| 1 | 0.650 | 5.00 | 0.9428 | 0.0140 | 0.9039 | 0.0195 | 0.8791 | 0.0360 |
| 2 | 0.650 | 3.00 | 0.9429 | 0.0152 | 0.8997 | 0.0313 | 0.8842 | 0.0160 |
| 3 | 0.650 | 3.00 | 0.9368 | 0.0158 | 0.8975 | 0.0244 | 0.8696 | 0.0318 |
| 4 | 0.650 | 3.25 | 0.9390 | 0.0155 | 0.9021 | 0.0271 | 0.8652 | 0.0207 |
| 5 | 0.650 | 3.25 | 0.9416 | 0.0098 | 0.9006 | 0.0183 | 0.8695 | 0.0263 |
| 6 | 0.650 | 3.50 | 0.9414 | 0.0074 | 0.9100 | 0.0261 | 0.8700 | 0.0273 |
| 7 | 0.650 | 3.25 | 0.9374 | 0.0148 | 0.8953 | 0.0220 | 0.8623 | 0.0194 |
| 8 | 0.650 | 3.00 | 0.9391 | 0.0118 | 0.9019 | 0.0319 | 0.8747 | 0.0218 |
| 9 | 0.650 | 6.00 | 0.9408 | 0.0129 | 0.9027 | 0.0160 | 0.8687 | 0.0289 |
| 10 | 0.650 | 3.25 | 0.9390 | 0.0147 | 0.9039 | 0.0266 | 0.8682 | 0.0245 |
| 11 | 0.650 | 3.00 | 0.9386 | 0.0163 | 0.8971 | 0.0225 | 0.8689 | 0.0322 |
| 12 | 0.650 | 4.50 | 0.9411 | 0.0148 | 0.8960 | 0.0268 | 0.8808 | 0.0289 |
| 13 | 0.650 | 3.25 | 0.9364 | 0.0253 | 0.8967 | 0.0358 | 0.8700 | 0.0311 |
| 14 | 0.650 | 3.75 | 0.9419 | 0.0167 | 0.9004 | 0.0222 | 0.8753 | 0.0319 |
| 15 | 0.650 | 5.00 | 0.9388 | 0.0164 | 0.9021 | 0.0238 | 0.8774 | 0.0408 |
| 16 | 0.650 | 3.00 | 0.9386 | 0.0089 | 0.8975 | 0.0183 | 0.8648 | 0.0188 |
| 17 | 0.650 | 3.00 | 0.9353 | 0.0195 | 0.9058 | 0.0200 | 0.8593 | 0.0260 |
| **mean** | | | **0.9393** | | **0.9006** | | **0.8706** | |
| **median** | | | **0.9390** | | **0.9005** | | **0.8696** | |
| **std** | | | **0.0022** | | **0.0040** | | **0.0066** | |

Table 2: Results for the Two-class experimentation using a TCNN classifier; Forward Falls results. The acronym ID stands for participant ID.

| **FORWARD FALLS** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **UMAFall data set** | | | | | | | | |
| **ID** | **size** | **decay** | **ROC** | | **Sens** | | **Spec** | |
| | | | **mean** | **std** | **mean** | **std** | **mean** | **std** |
| 1 | 10.0000 | 2.0000 | 0.9220 | 0.1315 | 0.7550 | 0.3122 | 0.9200 | 0.1033 |
| 2 | 20.0000 | 0.8750 | 0.9150 | 0.0782 | 0.8800 | 0.1398 | 0.8650 | 0.1180 |
| 3 | 10.0000 | 1.1250 | 0.9290 | 0.0745 | 0.8300 | 0.2003 | 0.8750 | 0.1087 |
| 4 | 10.0000 | 0.7500 | 0.9460 | 0.0962 | 0.8600 | 0.1350 | 0.8833 | 0.2139 |
| 9 | 10.0000 | 0.1250 | 0.9207 | 0.0846 | 0.9000 | 0.1054 | 0.8700 | 0.1989 |
| 16 | 15.0000 | 0.1250 | 0.9222 | 0.0960 | 0.8667 | 0.1721 | 0.8500 | 0.1610 |
| 17 | 10.0000 | 0.7500 | 0.9533 | 0.0898 | 0.8950 | 0.1462 | 0.9200 | 0.1687 |
| **mean** | | | **0.9297** | | **0.8552** | | **0.8833** | |
| **median** | | | **0.9222** | | **0.8667** | | **0.8750** | |
| **std** | | | **0.0143** | | **0.0501** | | **0.0270** | |
| **Özdemir & Barshan data set** | | | | | | | | |
| **ID** | **size** | **decay** | **ROC** | | **Sens** | | **Spec** | |
| | | | **mean** | **std** | **mean** | **std** | **mean** | **std** |
| 1 | 15.0000 | 0.1250 | 0.9534 | 0.0111 | 0.9151 | 0.0250 | 0.8630 | 0.0350 |
| 2 | 25.0000 | 0.1250 | 0.9551 | 0.0118 | 0.9127 | 0.0238 | 0.8726 | 0.0162 |
| 3 | 25.0000 | 0.1250 | 0.9508 | 0.0126 | 0.9136 | 0.0222 | 0.8587 | 0.0249 |
| 4 | 10.0000 | 0.1250 | 0.9522 | 0.0122 | 0.9183 | 0.0242 | 0.8583 | 0.0233 |
| 5 | 25.0000 | 0.2500 | 0.9519 | 0.0107 | 0.9156 | 0.0225 | 0.8613 | 0.0286 |
| 6 | 25.0000 | 0.1250 | 0.9551 | 0.0073 | 0.9253 | 0.0204 | 0.8666 | 0.0246 |
| 7 | 20.0000 | 0.1250 | 0.9502 | 0.0074 | 0.9024 | 0.0195 | 0.8636 | 0.0164 |
| 8 | 20.0000 | 0.1250 | 0.9540 | 0.0102 | 0.9134 | 0.0234 | 0.8685 | 0.0208 |
| 9 | 25.0000 | 0.2500 | 0.9515 | 0.0064 | 0.9209 | 0.0203 | 0.8542 | 0.0189 |
| 10 | 15.0000 | 0.1250 | 0.9502 | 0.0109 | 0.9193 | 0.0295 | 0.8606 | 0.0275 |
| 11 | 20.0000 | 0.1250 | 0.9525 | 0.0105 | 0.9079 | 0.0227 | 0.8648 | 0.0243 |
| 12 | 20.0000 | 0.2500 | 0.9522 | 0.0109 | 0.9080 | 0.0244 | 0.8671 | 0.0357 |
| 13 | 15.0000 | 0.1250 | 0.9502 | 0.0144 | 0.9121 | 0.0237 | 0.8576 | 0.0299 |
| 14 | 25.0000 | 0.2500 | 0.9548 | 0.0108 | 0.9148 | 0.0279 | 0.8678 | 0.0321 |
| 15 | 25.0000 | 0.1250 | 0.9524 | 0.0123 | 0.9135 | 0.0265 | 0.8603 | 0.0429 |
| 16 | 25.0000 | 0.2500 | 0.9489 | 0.0098 | 0.9151 | 0.0192 | 0.8525 | 0.0189 |
| 17 | 20.0000 | 0.1250 | 0.9463 | 0.0144 | 0.9086 | 0.0246 | 0.8551 | 0.0361 |
| **mean** | | | **0.9518** | | **0.9138** | | **0.8619** | |
| **median** | | | **0.9520** | | **0.9135** | | **0.8610** | |
| **std** | | | **0.0024** | | **0.0056** | | **0.0057** | |

the results for this one-class classifier have not been included. On the other hand, the CENTROIDS classifier performed really well, especially for the forward fall type (see Table 3). The performance of the CENTROIDS is far from ideal but might work well in this anomaly detection: high Sensitivity and low Specificity induces high percentage of false alarms. Nevertheless, the low rate filtering can be enough to improve the overall performance of the solution. This will be analyzed in the next subsection.

### *4.3. Results for the solution experimentation*

This last section focuses on the evaluation of the complete solution; the main question is whether the one-class stage collaborates in an ulterior enhancement of the two-class classifier. The obtained results will be compared with the base-line solution proposed in [45, 48], referred as on-wrist Abbate. Because of the poor results of the OCSVM and the better performance of the TCNN, only the results for the combination of the CENTROIDS and the TCNN are included. Table 4 and Table 5 depict the figures obtained with the UMA Fall and the Özdemir & Barshan data sets, respectively.

What is most remarkable is that the one-class filtering does not improve the general performance of the two-class method but for the forward falls. In this case, the CENTROIDS + TCNN is better than the TCNN alone. In general, the outcome from the combination of these two techniques is similar to that of the on-wrist Abbate, but with a small reduction of the number of false alarms but at the price of a worse alarm detection. This finding, a so simple classifier as the CENTROIDS helps the TCNN, suggests that the filter might be specific to each type of fall, and that this stage would help the final classifier filtering some conflicting cases. As shown in [50], the introduction of online learning to enhance the performance of the anomaly filters or the two-class classifiers can also empower the final results.

Moreover, it is probable that if the learning of the two-class classifier uses data from participants that behave similarly to the current user, the classification results will be enhanced. This idea has been also proposed in [50]; however, the difficulty relies on determining the participants with similar behavior, that is, how to measure similarities in the behavior of two participants.

Table 3: Performance of the One-Class classifiers. Results for all the types of fall obtained from learning the CENTROIDS classifier. ID stands for the participant ID within each data set.

| **UMAFall data set** | | | | | | |
|---|---|---|---|---|---|---|
| **ID** | **FORWARD FALLS** | | **BACKWARD FALLS** | | **LATERAL FALLS** | |
| | **Sens** | **Spec** | **Sens** | **Spec** | **Sens** | **Spec** |
| | **mean / std** | **mean / std** | **mean / std** | **mean / std** | **mean / std** | **mean / std** |
| 1 | 1.000 / 0.000 | 0.000 / 0.000 | 1.000 / 0.000 | 0.000 / 0.000 | 1.000 / 0.000 | 0.000 / 0.000 |
| 2 | 1.000 / 0.000 | 0.000 / 0.000 | 1.000 / 0.000 | 0.000 / 0.000 | 1.000 / 0.000 | 0.000 / 0.000 |
| 3 | 1.000 / 0.000 | 0.200 / 0.447 | 1.000 / 0.000 | 0.200 / 0.447 | 1.000 / 0.000 | 0.200 / 0.447 |
| 4 | 1.000 / 0.000 | 0.000 / 0.000 | 1.000 / 0.000 | 0.000 / 0.000 | 1.000 / 0.000 | 0.000 / 0.000 |
| 9 | 1.000 / 0.000 | 0.000 / 0.000 | 1.000 / 0.000 | 0.000 / 0.000 | 1.000 / 0.000 | 0.000 / 0.000 |
| 16 | 1.000 / 0.000 | 0.000 / 0.000 | 0.955 / 0.015 | 0.000 / 0.000 | 1.000 / 0.000 | 0.000 / 0.000 |
| 17 | 1.000 / 0.000 | 0.000 / 0.000 | - / - | - / - | 1.000 / 0.000 | 0.000 / 0.000 |
| **mn** | **1.0000** | **0.0286** | **0.9925** | **0.0333** | **1.0000** | **0.0286** |
| **mdn** | **1.0000** | **0.0000** | **1.0000** | **0.0000** | **1.0000** | **0.0000** |
| **std** | **0.0000** | **0.0756** | **0.0184** | **0.0816** | **0.0000** | **0.0756** |
| **Özdemir & Barshan data set** | | | | | | |
| **ID** | **FORWARD FALLS** | | **BACKWARD FALLS** | | **LATERAL FALLS** | |
| | **Sens** | **Spec** | **Sens** | **Spec** | **Sens** | **Spec** |
| | **mean / std** | **mean / std** | **mean / std** | **mean / std** | **mean / std** | **mean / std** |
| 1 | 0.936 / 0.023 | 0.156 / 0.142 | 0.980 / 0.007 | 0.156 / 0.142 | 0.967 / 0.011 | 0.156 / 0.142 |
| 2 | 0.992 / 0.005 | 0.099 / 0.111 | 0.980 / 0.007 | 0.099 / 0.111 | 0.916 / 0.000 | 0.099 / 0.111 |
| 3 | 0.981 / 0.008 | 0.073 / 0.098 | 1.000 / 0.000 | 0.073 / 0.098 | 1.000 / 0.000 | 0.073 / 0.098 |
| 4 | 0.984 / 0.021 | 0.141 / 0.159 | 0.950 / 0.062 | 0.141 / 0.159 | 0.989 / 0.016 | 0.141 / 0.159 |
| 5 | 0.952 / 0.003 | 0.068 / 0.079 | 0.974 / 0.005 | 0.068 / 0.079 | 0.947 / 0.005 | 0.068 / 0.079 |
| 6 | 0.819 / 0.063 | 0.166 / 0.204 | 0.740 / 0.092 | 0.166 / 0.204 | 0.950 / 0.017 | 0.166 / 0.204 |
| 7 | 0.979 / 0.000 | 0.072 / 0.091 | 0.998 / 0.004 | 0.072 / 0.091 | 1.000 / 0.000 | 0.072 / 0.091 |
| 8 | 0.988 / 0.006 | 0.016 / 0.052 | 0.988 / 0.004 | 0.016 / 0.052 | 1.000 / 0.000 | 0.016 / 0.052 |
| 9 | 0.951 / 0.015 | 0.107 / 0.157 | 0.912 / 0.030 | 0.107 / 0.157 | 0.878 / 0.012 | 0.107 / 0.157 |
| 10 | 0.990 / 0.000 | 0.034 / 0.056 | 1.000 / 0.000 | 0.034 / 0.056 | 0.988 / 0.006 | 0.034 / 0.056 |
| 11 | 1.000 / 0.000 | 0.093 / 0.070 | 1.000 / 0.000 | 0.093 / 0.070 | 1.000 / 0.000 | 0.093 / 0.070 |
| 12 | 0.986 / 0.007 | 0.131 / 0.091 | 0.990 / 0.003 | 0.131 / 0.091 | 0.906 / 0.027 | 0.131 / 0.091 |
| 13 | 0.991 / 0.004 | 0.223 / 0.170 | 0.930 / 0.025 | 0.223 / 0.170 | 0.951 / 0.016 | 0.223 / 0.170 |
| 14 | 0.932 / 0.006 | 0.062 / 0.073 | 0.978 / 0.003 | 0.062 / 0.073 | 1.000 / 0.000 | 0.062 / 0.073 |
| 15 | 0.980 / 0.004 | 0.018 / 0.038 | 1.000 / 0.000 | 0.018 / 0.038 | 1.000 / 0.000 | 0.018 / 0.038 |
| 16 | 1.000 / 0.000 | 0.077 / 0.090 | 1.000 / 0.000 | 0.077 / 0.090 | 1.000 / 0.000 | 0.077 / 0.090 |
| 17 | 0.994 / 0.006 | 0.043 / 0.072 | 0.990 / 0.000 | 0.043 / 0.072 | 1.000 / 0.000 | 0.043 / 0.072 |
| **mn** | **0.9703** | **0.0894** | **0.9646** | **0.0894** | **0.9705** | **0.0894** |
| **mdn** | **0.9855** | **0.0756** | **0.9891** | **0.0756** | **0.9949** | **0.0756** |
| **std** | **0.0455** | **0.0554** | **0.0654** | **0.0554** | **0.0404** | **0.0554** |

Table 4: Solution experimentation. Comparison results of the CENTROIDS+TCNN vs. the on-wrist Abbate solution in [45] when evaluated with the UMAFall data set. ND means the method did not detect any peak for the participant, while - means there were no falls for this type of fall and participant.

| **New Features & feature selection + CENTROIDS + TCNN** | | | | | | |
|---|---|---|---|---|---|---|
| **ID** | **FORWARD FALLS** | | **BACKWARD FALLS** | | **LATERAL FALLS** | |
| | **Sens** | **Spec** | **Sens** | **Spec** | **Sens** | **Spec** |
| | **mean / std** | **mean / std** | **mean / std** | **mean / std** | **mean / std** | **mean / std** |
| 1 | 1.000 / 0.000 | 1.000 / 0.000 | 1.000 / 0.000 | 0.857 / 0.378 | 1.000 / 0.000 | 0.714 / 0.488 |
| 2 | 1.000 / 0.000 | 1.000 / 0.000 | 0.600 / 0.000 | 0.333 / 0.577 | 0.666 / 0.000 | 0.666 / 0.577 |
| 3 | 1.000 / 0.000 | 0.800 / 0.447 | 0.428 / 0.000 | 0.800 / 0.447 | 0.500 / 0.000 | 0.800 / 0.447 |
| 4 | 1.000 / 0.000 | 1.000 / 0.000 | 0.666 / 0.000 | 1.000 / 0.000 | 0.428 / 0.000 | 0.500 / 0.500 |
| 9 | 0.750 / 0.000 | 1.000 / 0.000 | 0.428 / 0.000 | 0.250 / 0.500 | 0.857 / 0.000 | 0.500 / 0.577 |
| 16 | 0.900 / 0.000 | 0.733 / 0.335 | 0.450 / 0.000 | 0.550 / 0.416 | 0.882 / 0.000 | 0.783 / 0.333 |
| 17 | 0.714 / 0.000 | 0.750 / 0.500 | - / - | - / - | 0.750 / 0.000 | 0.750 / 0.500 |
| **mn** | **0.909** | **0.898** | **0.596** | **0.632** | **0.726** | **0.674** |
| **mdn** | **1.000** | **1.000** | **0.525** | **0.675** | **0.750** | **0.714** |
| **std** | **0.127** | **0.129** | **0.222** | **0.302** | **0.208** | **0.126** |
| **on-wrist Abbate et al solution** | | | | | | |
| **ID** | **FORWARD FALLS** | | **BACKWARD FALLS** | | **LATERAL FALLS** | |
| | **Sens** | **Spec** | **Sens** | **Spec** | **Sens** | **Spec** |
| | **mean / std** | **mean / std** | **mean / std** | **mean / std** | **mean / std** | **mean / std** |
| 1 | 0.800 / 0.000 | 0.600 / 0.547 | 0.833 / 0.000 | 0.200 / 0.447 | 0.750 / 0.000 | 0.800 / 0.447 |
| 2 | ND / ND | ND / ND | ND / ND | ND / ND | ND / ND | ND / ND |
| 3 | 0.750 / 0.000 | 0.400 / 0.418 | 0.750 / 0.000 | 0.500 / 0.500 | 0.750 / 0.000 | 0.600 / 0.547 |
| 4 | ND / ND | ND / ND | ND / ND | ND / ND | ND / ND | ND / ND |
| 9 | ND / ND | ND / ND | ND / ND | ND / ND | ND / ND | ND / ND |
| 12 | ND / ND | ND / ND | 0.481 / 0.055 | 0.888 / 0.333 | ND / ND | ND / ND |
| 16 | 0.681 / 0.000 | 0.500 / 0.500 | 0.857 / 0.000 | 0.857 / 0.378 | 0.882 / 0.000 | 0.785 / 0.393 |
| 17 | ND / ND | ND / ND | - / - | - / - | ND / ND | ND / ND |
| **mn** | **0.744** | **0.500** | **0.731** | **0.612** | **0.794** | **0.729** |
| **mdn** | **0.750** | **0.500** | **0.792** | **0.679** | **0.750** | **0.786** |
| **std** | **0.059** | **0.100** | **0.172** | **0.326** | **0.076** | **0.112** |

## 5. Conclusion

This study proposes a three-stage process to detect fall events using a 3DACC wearable. The main contributions of this research are the event detection method and the combination of user-centred models and generalized models working together in order to detect the falls. The experimentation has been focused on three types of falls: forward falls, backward falls and lateral falls. A first stage is the event detection that identifies the relevant peaks of activity; for each detected peak several features are computed. Interestingly, the thresholds in this event detection are specifically calculated

Table 5: Solution experimentation. Comparison results of the CENTROIDS+TCNN vs. the on-wrist Abbate solution in [45] when evaluated with the Özdemir & Barshan data set.

| **New Features & feature selection + CENTROIDS + TCNN** | | | | | | |
|---|---|---|---|---|---|---|
| **ID** | **FORWARD FALLS** | | **BACKWARD FALLS** | | **LATERAL FALLS** | |
| | **Sens** | **Spec** | **Sens** | **Spec** | **Sens** | **Spec** |
| | **mean / std** | **mean / std** | **mean / std** | **mean / std** | **mean / std** | **mean / std** |
| 1 | 0.765 / 0.007 | 0.817 / 0.141 | 0.778 / 0.003 | 0.882 / 0.159 | 0.763 / 0.000 | 0.950 / 0.158 |
| 2 | 0.836 / 0.005 | 0.791 / 0.115 | 0.803 / 0.005 | 0.917 / 0.073 | 0.716 / 0.000 | 0.845 / 0.087 |
| 3 | 0.834 / 0.003 | 0.978 / 0.044 | 0.845 / 0.000 | 1.000 / 0.000 | 0.800 / 0.000 | 0.990 / 0.031 |
| 4 | 0.873 / 0.005 | 0.938 / 0.068 | 0.802 / 0.052 | 0.558 / 0.194 | 0.820 / 0.016 | 0.723 / 0.206 |
| 5 | 0.801 / 0.003 | 0.915 / 0.096 | 0.854 / 0.003 | 0.899 / 0.087 | 0.828 / 0.005 | 0.909 / 0.096 |
| 6 | 0.579 / 0.035 | 0.768 / 0.128 | 0.559 / 0.014 | 0.872 / 0.154 | 0.360 / 0.000 | 0.983 / 0.052 |
| 7 | 0.867 / 0.000 | 0.922 / 0.089 | 0.903 / 0.000 | 0.885 / 0.088 | 0.880 / 0.000 | 0.981 / 0.038 |
| 8 | 0.798 / 0.006 | 0.846 / 0.118 | 0.827 / 0.000 | 0.879 / 0.097 | 0.768 / 0.000 | 0.843 / 0.117 |
| 9 | 0.751 / 0.008 | 0.956 / 0.072 | 0.838 / 0.030 | 0.959 / 0.067 | 0.802 / 0.007 | 0.956 / 0.072 |
| 10 | 0.768 / 0.000 | 0.976 / 0.049 | 0.833 / 0.000 | 1.000 / 0.000 | 0.880 / 0.000 | 0.963 / 0.082 |
| 11 | 0.860 / 0.000 | 0.885 / 0.096 | 0.837 / 0.000 | 0.990 / 0.031 | 0.819 / 0.000 | 0.911 / 0.112 |
| 12 | 0.920 / 0.007 | 0.776 / 0.072 | 0.901 / 0.003 | 0.703 / 0.051 | 0.788 / 0.022 | 0.794 / 0.092 |
| 13 | 0.946 / 0.004 | 0.950 / 0.085 | 0.891 / 0.022 | 0.980 / 0.063 | 0.733 / 0.005 | 0.980 / 0.063 |
| 14 | 0.756 / 0.003 | 0.861 / 0.086 | 0.786 / 0.000 | 0.791 / 0.104 | 0.762 / 0.000 | 0.821 / 0.142 |
| 15 | 0.857 / 0.004 | 0.933 / 0.161 | 0.946 / 0.000 | 0.938 / 0.080 | 0.879 / 0.000 | 0.924 / 0.069 |
| 16 | 0.965 / 0.000 | 0.913 / 0.132 | 0.868 / 0.000 | 0.924 / 0.089 | 0.842 / 0.000 | 0.965 / 0.081 |
| 17 | 0.880 / 0.006 | 0.875 / 0.094 | 0.906 / 0.000 | 0.961 / 0.068 | 0.861 / 0.000 | 0.945 / 0.077 |
| **mn** | **0.831** | **0.893** | **0.838** | **0.891** | **0.784** | **0.909** |
| **mdn** | **0.847** | **0.914** | **0.842** | **0.921** | **0.811** | **0.935** |
| **std** | **0.092** | **0.068** | **0.086** | **0.119** | **0.124** | **0.080** |

| **on-wrist Abbate et al solution** | | | | | | |
|---|---|---|---|---|---|---|
| **ID** | **FORWARD FALLS** | | **BACKWARD FALLS** | | **LATERAL FALLS** | |
| | **Sens** | **Spec** | **Sens** | **Spec** | **Sens** | **Spec** |
| | **mean / std** | **mean / std** | **mean / std** | **mean / std** | **mean / std** | **mean / std** |
| 1 | 0.611 / 0.000 | 0.796 / 0.151 | 0.833 / 0.000 | 0.808 / 0.117 | 0.816 / 0.000 | 0.929 / 0.079 |
| 2 | 0.847 / 0.000 | 0.929 / 0.086 | 0.804 / 0.000 | 0.957 / 0.060 | 0.806 / 0.000 | 0.991 / 0.026 |
| 3 | 0.744 / 0.000 | 0.916 / 0.090 | 0.791 / 0.000 | 0.976 / 0.055 | 0.887 / 0.000 | 0.964 / 0.060 |
| 4 | 0.858 / 0.000 | 0.900 / 0.179 | 0.826 / 0.000 | 0.842 / 0.217 | 0.783 / 0.000 | 0.954 / 0.108 |
| 5 | 0.804 / 0.000 | 0.851 / 0.111 | 0.728 / 0.000 | 0.873 / 0.040 | 0.766 / 0.000 | 0.895 / 0.076 |
| 6 | 0.777 / 0.000 | 0.773 / 0.127 | 0.844 / 0.000 | 0.912 / 0.083 | 0.850 / 0.000 | 0.938 / 0.078 |
| 7 | 0.694 / 0.000 | 0.949 / 0.045 | 0.781 / 0.000 | 0.980 / 0.041 | 0.869 / 0.000 | 0.960 / 0.069 |
| 8 | 0.648 / 0.000 | 0.877 / 0.106 | 0.689 / 0.000 | 0.833 / 0.102 | 0.777 / 0.000 | 0.904 / 0.092 |
| 9 | 0.800 / 0.000 | 0.864 / 0.067 | 0.863 / 0.000 | 0.870 / 0.056 | 0.824 / 0.000 | 0.880 / 0.066 |
| 10 | 0.814 / 0.000 | 0.978 / 0.046 | 0.903 / 0.000 | 0.978 / 0.046 | 0.847 / 0.000 | 0.954 / 0.049 |
| 11 | 0.847 / 0.000 | 0.988 / 0.035 | 0.895 / 0.000 | 1.000 / 0.000 | 0.718 / 0.000 | 0.977 / 0.070 |
| 12 | 0.800 / 0.000 | 0.848 / 0.114 | 0.771 / 0.000 | 0.892 / 0.095 | 0.783 / 0.000 | 1.000 / 0.000 |
| 13 | 0.875 / 0.005 | 0.904 / 0.083 | 0.854 / 0.000 | 0.935 / 0.118 | 0.666 / 0.000 | 0.929 / 0.063 |
| 14 | 0.833 / 0.000 | 0.877 / 0.089 | 0.833 / 0.000 | 0.910 / 0.064 | 0.683 / 0.000 | 0.912 / 0.103 |
| 15 | 0.858 / 0.000 | 0.945 / 0.048 | 0.895 / 0.000 | 0.962 / 0.048 | 0.758 / 0.000 | 0.960 / 0.074 |
| 16 | 0.759 / 0.000 | 0.949 / 0.075 | 0.903 / 0.000 | 0.971 / 0.065 | 0.902 / 0.000 | 0.956 / 0.077 |
| 17 | 0.827 / 0.000 | 0.971 / 0.048 | 0.953 / 0.000 | 0.971 / 0.048 | 0.932 / 0.000 | 0.961 / 0.051 |
| **mn** | **0.780** | **0.908** | **0.834** | **0.929** | **0.804** | **0.946** |
| **mdn** | **0.810** | **0.910** | **0.839** | **0.947** | **0.795** | **0.956** |
| **std** | **0.063** | **0.058** | **0.071** | **0.054** | **0.076** | **0.034** |

for each user. The second stage includes a one-class classifier that aims to filter the peaks that come from ADLs, while diverting those uncommon patterns to a two-class classifier. This latter stage is the responsible of labelling the fall events.

The experimentation included two publicly available data sets. The results show that for forward falls the one-class filtering enhances the performance of the TCNN. This finding suggests that each type of fall would eventually need a very specific filter, reinforcing the conclusion from [50] about the need of online learning solutions for both the one-class methods and the two-class classifiers: the more adapted to the current user the better the solution is. The results also suggest that it might be interesting to obtain a more accurate two-class classifier using data only from subjects with similar activity levels; the use of online learning might also improve the general performance in the classification stages.

## Acknowledgements

This research has been funded by the Spanish Ministry of Science and Innovation, under project MINECO-TIN2017-84804-R, and by the Grant FC-GRUPIN-IDI/2018/000226 project from the Asturias Regional Government.

## References

## References

[1] L. Rubenstein, Falls in older people: epidemiology, risk factors and strategies for prevention, Age Ageing 35 (2006) 37–41.

[2] S. Chaudhuri, H. Thompson, G. Demiris, Fall detection devices and their use with older adults: A systematic review, J Geriatr Phys Ther 37 (2014) 178–196.

[3] HelpLine, Why is a quick response so important when the elderly fall, `https://www.helpline.co.uk/blog/why-is-a-quick-response-so-important-when-the-elderly-fall/`, 2019. [Online; accessed 28-June-2019].

[4] WalaBot, Walabot home is the next-generation personal response system, `https://walabot.com/walabot-home`, 2019. [Online; accessed 28-June-2019].

[5] VAYYAR, Make yourself at home, https://vayyar.com/smart-home, 2019. [Online; accessed 28-June-2019].

[6] SENSIFALL, Sensifall, the smart floor fall detection system, http://www.sensifall.com/index.html, 2019. [Online; accessed 28-June-2019].

[7] D. Alert, Our medical alert systems: The perfect fit for any lifestyle, https://directalert.ca/, 2019. [Online; accessed 28-June-2019].

[8] A. Support, Use fall detection with apple watch series 4, https://support.apple.com/en-us/HT208944, 2018. [Online; accessed 28-June-2019].

[9] G. A. Fowler, The apple watch faces its toughest challenge yet: Grandma and grandpa, Washington Post, "https://www.washingtonpost.com/technology/2018/10/03/apple-watch-faces-its-toughest-challenge-yet-grandma/?noredirect=on&utm_term=.553f6258bdc1", 2018. [Online; accessed 28-June-2019].

[10] MobileReviewEh, Does it work? apple watch series 4 fall detection test, https://www.youtube.com/watch?v=qDlBX5JmTxw, 2019. [Online; accessed 28-June-2019].

[11] A. Leone, G. Diraco, P. Siciliano, Detecting falls with 3d range camera in ambient assisted living applications: a preliminary study, Medical Engineering & Physics 33 (2011) 770–781.

[12] W. Y. Shieh, J. C. Huang, Falling-incident detection and throughput enhancement in a multi-camera video-surveillance system, Medical Engineering & Physics 34 (2012) 954–963.

[13] E. E.Geertsema, G. H.Visser, M. A.Viergever, S. N.Kalitzin, Automated remote fall detection using impact features from video and audio, Journal of Biomechanics 88 (2019) 25–32.

[14] C. Ma, A. Shimada, H. Uchiyama, H. Nagahara, R. Taniguchi, Fall detection using optical level anonymous image sensing system, Optics & Laser Technology (2019) 44–61.

[15] Y. Peng, J. Peng, J. Li, P. Yan, B. Hu, Design and development of the fall detection system based on point cloud, Procedia Computer Science 147 (2019) 271–275.

[16] L. Yang, Y. Ren, W. Zhang, 3d depth image analysis for indoor fall detection of elderly people, Digital Communications and Networks 2 (2016) 24–43.

[17] M. Alwan, P. J. Rajendran, S. Kell, D. Mack, S. Dalal, M. Wolfe, R. Felder, A smart and passive floor-vibration based fall detector for elderly, in: 2nd International Conference on Information and Communication Technologies, volume 1, 2006.

[18] H. Rimminen, J. Lindström, M. Linnavuo, R. Sepponen, Detection of falls among the elderly by a floor sensor using the electric near field, IEEE Transactions on Information Technologies and Biomedicine 14 (2010) 1475–1476.

[19] E. Principi, DiegoDroghini, S. Squartinia, P. Olivetti, F. Piazza, Acoustic cues from the floor: A new approach for fall classification, Expert Systems with Applications 60 (2016) 51–61.

[20] R. Igual, C. Medrano, I. Plaza, Challenges, issues and trends in fall detection systems, BioMedical Engineering OnLine 12 (2013). URL: http://www.biomedical-engineering-online.com/content/12/1/66.

[21] Y. S. Delahoz, M. A. Labrador, Survey on fall detection and fall prevention using wearable and external sensors, Sensors 14 (2014) 19806–19842. URL: http://www.mdpi.com/1424-8220/14/10/19806/htm. doi:doi:10.3390/s141019806.

[22] S. S. Khan, JesseHoey, Review of fall detection techniques: A data availability perspective, Medical Engineering and Physics 39 (2017) 12–22. URL: http://www.sciencedirect.com/science/article/pii/S1350453316302600. doi:https://doi.org/10.1016/j.medengphy.2016.10.014.

[23] E. Casilari-Pérez, F. García-Lagos, A comprehensive study on the use of artificial neural networks in wearable fall detection systems, Expert Systems with Applications 138 (2019). doi:https://doi.org/10.1016/j.eswa.2019.07.028.

[24] A. Ngu, Y. Wu, H. Zare, A. Polican, B. Yarbrough, L. Yao, Fall detection using smartwatch sensor data with accessor architecture, in: H. Chen, D. Zeng, E. Karahanna, B. I. (Eds.), Proceedings of the International Conference on Smart Health ICSH 2017, Lecture Notes in Computer Science, volume 10347, Springer, 2017, pp. 81–93.

[25] J. R. Villar, E. de la Cal, V. González, J. Sedano, Using ensembles for improving fall detection, in: IV Jornadas de Fusión de la Información y Ensemble Learning (IV FINO), 2018.

[26] F. Wu, H. Zhao, Y. Zhao, H. Zhong, Development of a wearable-sensor-based fall detection system, International Journal of Telemedicine and Applications 2016, Article ID 576364 (2015) 11 pages. URL: https://www.hindawi.com/journals/ijta/2015/576364/. doi:http://dx.doi.org/10.1155/2015/576364.

[27] H. Cao, S. Wu, Z. Zhou, C.-C. Lin, C.-Y. Yang, S.-T. Lee, C.-T. Wu, A fall detection method based on acceleration data and hidden Markov model, in: 2016 IEEE International Conference on Signal and Image Processing (ICSIP), IEEE, 2016, pp. 684–689. URL: http://ieeexplore.ieee.org/document/7888350/. doi:10.1109/SIPROCESS.2016.7888350.

[28] S. Yu, H. Chen, R. A. Brown, Hidden Markov Model-Based Fall Detection With Motion Sensor Orientation Calibration: A Case for Real-Life Home Monitoring, IEEE Journal of Biomedical and Health Informatics 22 (2018) 1847–1853. URL: https://ieeexplore.ieee.org/document/8171718/. doi:10.1109/JBHI.2017.2782079.

[29] P. Jatesiktat, W. T. Ang, An elderly fall detection using a wrist-worn accelerometer and barometer, in: 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2017, pp. 125–130. doi:10.1109/EMBC.2017.8036778.

[30] A. M. Sabatini, G. Ligorio, A. Mannini, V. Genovese, L. Pinna, Prior-to- and post-impact fall detection using inertial and barometric altimeter measurements, IEEE Transactions on Neural Systems and Rehabilitation Engineering 24 (2016) 774–783. URL: http://ieeexplore.ieee.org/abstract/document/7173441/. doi:10.1109/TNSRE.2015.2460373.

[31] A. K. Bourke, J. Klenk, L. Schwickert, K. Aminian, E. A. F. Ihlen, S. Mellone, J. L. Helbostad, L. Chiari, C. Becker, Fall detection algorithms for real-world falls harvested from lumbar sensors in the elderly population: A machine learning approach, in: 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2016, pp. 3712–3715. doi:10.1109/EMBC.2016.7591534.

[32] Q. T. Huynh, U. D. Nguyen, L. B. Irazabal, N. Ghassemian, B. Q. Tran, Optimization of an accelerometer and gyroscope-based fall detection algorithm, Journal of Sensors 2015, Article ID 452078 (2015) 8 pages. doi:http://dx.doi.org/10.1155/2015/452078.

[33] A. Sorvala, E. Alasaarela, H. Sorvoja, R. Myllyla, A two-threshold fall detection algorithm for reducing false alarms, in: Proceedings of 2012 6th International Symposium on Medical Information and Communication Technology (ISMICT), 2012, p. 10.1109/ISMICT.2012.6203028.

[34] G. Rescio, A. Leone, P. Siciliano, Supervised machine learning scheme for electromyography-based pre-fall detection system, Expert Systems with Applications 100 (2018) 95–105.

[35] Y. Wu, Y. Su, R. Feng, N. Yu, X. Zang, Wearable-sensor-based pre-impact fall detection system with a hierarchical classifier, Measurement 140 (2019) 283–292.

[36] F. Bianchi, S. J. Redmond, M. R. Narayanan, S. Cerutti, N. H. Lovell, Barometric pressure and triaxial accelerometry-based falls event detection, IEEE Transactions on Neural Systems and Rehabilitation Engineering 18 (2010) 619–627. doi:10.1109/TNSRE.2010.2070807.

[37] E. Boutellaa, O. Kerdjidj, K. Ghanemc, Covariance matrix based fall detection from multiple wearable sensors, Journal of Biomedical Informatics 94 (2019) 103189.

[38] T. Zhang, J. Wang, L. Xu, P. Liu, Fall detection by wearable sensor and one-class svm algorithm, in: I. G. Huang DS., Li K. (Ed.), Intelligent Computing in Signal Processing and Pattern Recognition, volume 345 of *Lecture Notes in Control and Information Systems*, Springer Berlin Heidelberg, 2006,

pp. 858–863. URL: https://link.springer.com/chapter/10.1007%2F978-3-540-37258-5_104?LI=true#citeas. doi:https://doi.org/10.1007/978-3-540-37258-5_104.

[39] A. Bourke, J. O'Brien, G. Lyons, Evaluation of a threshold-based triaxial accelerometer fall detection algorithm, Gait and Posture 26 (2007) 194–199.

[40] A. Bourke, P. van de Ven, M. Gamble, R. O'Connor, K. Murphy, E. Bogan, E. McQuade, P. Finucane, G. Olaighin, J. Nelson, Evaluation of waist-mounted tri-axial accelerometer based fall-detection algorithms during scripted and continuous unscripted activities, Journal of Biomechanics 43 (2010) 3051–3057.

[41] F. Bagala, C. Becker, A. Cappello, L. Chiari, K. Aminian, J. M. Hausdorff, W. Zijlstra, J. Klenk, Evaluation of accelerometer-based fall detection algorithms on real-world falls, PLoS ONE 7 (2012) e37062. doi:https://doi.org/10.1371/journal.pone.0037062.

[42] M. Kangas, A. Konttila, P. Lindgren, I. Winblad, T. Jämsaä, Comparison of low-complexity fall detection algorithms for body attached accelerometers, Gait and Posture 28 (2008) 285–291.

[43] M. Kangas, I. Vikman, L. Nyberg, R. Korpelainen, J. Lindblom, T. Jamsa, Comparison of real-life accidental falls in older people with experimental falls in middle-aged test subjects, Gait and Posture 35 (2012) 500–505.

[44] E. Casilari, M. A. Oviedo-Jiménez, Automatic fall detection system based on the combined use of a smartphone and a smartwatch, PLOS ONE 10 (2015) 1–11. URL: https://doi.org/10.1371/journal.pone.0140929. doi:10.1371/journal.pone.0140929.

[45] S. Abbate, M. Avvenuti, F. Bonatesta, G. Cola, P. Corsini, AlessioVecchio, A smartphone-based fall detection system, Pervasive and Mobile Computing 8 (2012) 883–899.

[46] S. Abbate, M. Avvenuti, P. Corsini, J. Light, A. Vecchio, Wireless Sensor Networks: Application - Centric Design, Intech, 2010, p. 22. doi:10.5772/13802.

[47] P. Tsinganos, A. Skodras, A smartphone-based fall detection system for the elderly, in: Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis, 2017, p. 10.1109/ISPA.2017.8073568.

[48] S. B. Khojasteh, J. R. Villar, C. Chira, V. M. González, E. de la Cal, Improving fall detection using an on-wrist wearable accelerometer, Sensors 18 (2018) 1350. doi:https://doi.org/10.3390/s18051350.

[49] M. Fáñez, J. R. Villar, E. de la Cal, V. M. González, J. Sedano, Feature clustering to improve fall detection: a preliminary study, in: 14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019), Advances in Intelligent Systems and Computing, Springer, volume 950, 2019, pp. 219–228.

[50] J. R. Villar, E. de la Cal, M. Fáñez, V. M. González, J. Sedano, User-centered fall detection using supervised, online learning and transfer learning, in press in Progress in Artificial Intelligence (2019).

[51] I. P. E. S. Putra, J. Brusey, E. Gaura, R. Vesilo, An event-triggered machine learning approach for accelerometer-based fall detection, Sensors 18 (2018) 2034. doi:https://doi:10.3390/s18010020.

[52] P. Kostopoulos, T. Nunes, K. Salvi, M. Deriaz, J. Torrent, F2d: A fall detection system tested with real data from daily life of elderly people, in: 2015 17th International Conference on E-health Networking, Application Services (HealthCom), 2015, pp. 397–403.

[53] H. Gjoreski, J. Bizjak, M. Gams, Using smartwatch as telecare and fall detection device, in: 2016 12th International Conference on Intelligent Environments (IE), 2016, pp. 242–245. doi:10.1109/IE.2016.55.

[54] A. Hakim, M. S. Huq, S. Shanta, B. Ibrahim, Smartphone based data mining for fall detection: Analysis and design, Procedia Computer Science 105 (2017) 46–51. URL: http://www.sciencedirect.com/science/article/pii/S1877050917302065. doi:https://doi.org/10.1016/j.procs.2017.01.188.

[55] R. Igual, C. Medrano, I. Plaza, A comparison of public datasets for acceleration-based fall detection, Medical Engineering and Physics

37 (2015) 870–878. URL: http://www.sciencedirect.com/science/article/pii/S1350453315001575. doi:https://doi.org/10.1016/j.medengphy.2015.06.009.

[56] M. Deutsch, H. Burgsteiner, Health Informatics Meets eHealth, volume 223 of *Studies in Health Technology and Informatics*, IOS Press, 2016, pp. 259–266. doi:10.3233/978-1-61499-645-3-259.

[57] C. Medrano, I. Plaza, R. Igual, Á. Sánchez, M. Castr0, The effect of personalization on smartphone-based fall detectors, Sensors 16, Article ID 117 (2016) 117. URL: http://www.mdpi.com/1424-8220/16/1/117/htm. doi:10.3390/s16010117.

[58] T. Mauldin, M. Canby, V. Metsis, A. Ngu, C. Rivera, Smartfall: A smartwatch-based fall detection system using deep learning, Sensors 18 (2018) 3363. URL: http://dx.doi.org/10.3390/s18103363. doi:10.3390/s18103363.

[59] L. Chen, R. Li, H. Zhang, L. Tian, N. Chen, Intelligent fall detection method based on accelerometer data from a wrist-worn smart watch, Measurement 140 (2019) 215–226.

[60] E. Casilari, J.-A. Santoyo-Ramón, J.-M. Cano-García, Analysis of public datasets for wearable fall detection systems, Sensors 17 (2017) 4324 – 4338. URL: http://www.mdpi.com/1424-8220/17/7/1513. doi:https://10.3390/s17071513.

[61] E. Casilari, J. A. Santoyo-Ramón, J. M. Cano-García, Umafall: A multisensor dataset for the research on automatic fall detection, Procedia Computer Science 110 (2017) 32 – 39. URL: http://www.sciencedirect.com/science/article/pii/S1877050917312899. doi:https://doi.org/10.1016/j.procs.2017.06.110, 14th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2017) / 12th International Conference on Future Networks and Communications (FNC 2017) / Affiliated Workshops.

[62] A. T. Özdemir, B. Barshan, Detecting falls with wearable sensors using machine learning techniques, Sensors 14 (2014) 10691–10708.

[63] S. Gasparrini, E. Cippitelli, E. Gambi, S. Spinsante, J. Wahslen, I. Orhan, T. Lindh, Proposal and experimental evaluation of

fall detection solution based on wearable and depth data fusion, in: ICT Innovations 2015, Advances in Intelligent Systems and Computing, volume 399, Springer, 2016, pp. 99–108. URL: http://www.tlc.dii.univpm.it/blog/databases4kinectandhttps://ieee-dataport.org/documents/tst-fall-detection-dataset-v2. doi:10.1007/978-3-319-25733-4_11.

[64] S. B. Khojasteh, J. R. Villar, E. de la Cal, V. M. González, J. Sedano, Fall detection analysis using a real fall dataset, in: International Joint Conference SOCO'18-CISIS'18-ICEUTE'18. SOCO'18-CISIS'18-ICEUTE'18 2018. Advances in Intelligent Systems and Computing, volume 771, 2018, pp. 334–343.

[65] D. Droghini, D. Ferretti, E. Principi, S. Squartini, 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 2017 (2017) 1–13. doi:10.1155/2017/1512670.

[66] M. Kuhn, The caret package, http://topepo.github.io/caret/index.html, 2017. Last checked 15-1-2018.

[67] e1071: Misc functions of the department of statistics, probability theory group (formerly: E1071), tu wien, https://cran.r-project.org/web/packages/e1071/index.html, 2019.

[68] M. Yu, Y. Yu, A. Rhuma, S. Mohsen, R. Naqvi, L. Wang, J. A. Chambers, An online one class support vector machine-basedperson-specific fall detection system for monitoringan elderly individual in a room environment, IEEE Journal of Biomedical and Health Informatics 17 (2013) 1002–1014.

[69] nnet: Feed-forward neural networks and multinomial log-linear models, https://cran.r-project.org/web/packages/nnet/index.html, 2019.