Digital Signal Processing 125 (2022) 103129

Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

Classification of fall directions via wearable motion sensors

Mustafa Şahin Turan [a], Billur Barshan [b,*]

[a] Institute of Mechanical Eng., École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
[b] Dept. of Electrical and Electronics Eng., Bilkent University, Bilkent, TR-06800 Ankara, Turkey

ARTICLE INFO

Article history:
Available online 7 June 2021

Keywords:
Wearables
Wearable sensors
Motion sensors
Fall-direction classification
Machine learning classifiers
Assistive technology

ABSTRACT

Effective fall-detection and classification systems are vital in mitigating severe medical and economical consequences of falls to people in the fall risk groups. One class of such systems is based on wearable sensors. While there is a vast amount of academic work on this class of systems, not much effort has been devoted to the investigation of effective and robust algorithms and like-for-like comparison of state-of-the-art algorithms using a sufficiently large dataset. In this article, fall-direction classification algorithms are presented and compared on an extensive dataset, comprising a total of 1600 fall trials. Eight machine learning classifiers are implemented for fall-direction classification into four basic directions (forward, backward, right, and left). These are, namely, Bayesian decision making (BDM), least squares method (LSM), *k*-nearest neighbor classifier (*k*-NN), artificial neural networks (ANNs), support vector machines (SVMs), decision-tree classifier (DTC), random forest (RF), and adaptive boosting or AdaBoost (AB). BDM achieves perfect classification, followed by *k*-NN, SVM, and RF. Data acquired from only a single motion sensor unit, worn at the waist of the subject, are processed for experimental verification. Four of the classifiers (BDM, LSM, *k*-NN, and ANN) are modified to handle the presence of data from an unknown class and evaluated on the same dataset. In this robustness analysis, ANN and *k*-NN yield accuracies above 96.2%. The results obtained in this study are promising in developing real-world fall-classification systems as they enable fast and reliable classification of fall directions.

1. Introduction

Wearable technology is based on smart electronic devices affixed to different parts of the body to detect, analyze, and transmit information regarding body signals such as physiological and vital signs or motion data. Advancements in multiple enabling technologies including embedded systems, wireless sensor networks, mobile and edge computing have contributed to the development of wearables which take part as key elements in the Internet of Things (IoT) [1,2]. The connectivity between sensors, electronics, and software enables objects to process and exchange data through the internet with other connected devices and systems. Such a network of computing intelligence and communicating smart sensors allows the extraction of valuable information about the user state and well being.

Developing context-aware systems that can reliably monitor, interpret, and categorize activities of daily living (ADLs) is of paramount importance to improve the user's life quality. Monitoring ADLs and detecting abnormal behavior or high-risk events, such as falls, to support those with particular needs are challenging research issues that have received growing attention recently [3,4]. A fall is defined as an unstable event where a person unintentionally ends up on the ground or other lower level [5]. Typically, fall events occur in between ADLs.

Falls are often dangerous and might lead to serious injury or even death if medical attention is not provided rapidly. Serious medical conditions can arise due to either direct injury from the contact with the ground or the extended period of lying on the ground. Although the elderly face the direst danger related to falls, disabled people, patients with visual, gait, balance, orthopedic, and neurological problems, workers, athletes, mountain climbers, and children are also in the fall risk group. Falls may have different consequences for people in different age, gender, and profession groups: While they may result in serious and even life-threatening injuries to the elderly and special disease groups, children may experience trauma, and the professional performance of workers and athletes may be completely tarnished. Considering that more than 66% of people who have fallen once have a tendency to fall again [6], fallers are likely to suffer from the long-term physical, psychological, and social consequences of falls. Regardless of the nature of the faller, falling

* Corresponding author.
E-mail addresses: mustafa.turan@epfl.ch (M.Ş. Turan), billur@ee.bilkent.edu.tr (B. Barshan).

https://doi.org/10.1016/j.dsp.2021.103129
1051-2004/© 2021 Elsevier Inc. All rights reserved.

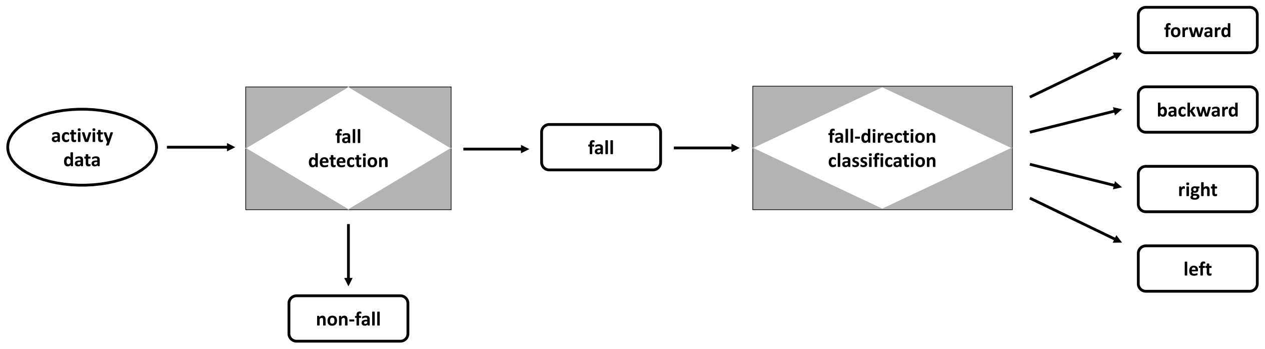

Fig. 1. The overall system consisting of a fall-detection and a fall-direction classification module.

is a serious, costly, and life-threatening public health problem. Developing reliable and effective fall-monitoring systems to mitigate the serious consequences of falls would improve the quality of life of those in the fall risk group and reduce the medical costs resulting from fall-related injuries [7]. This would have a positive effect on people, society, and even economies of countries.

Detection and classification of falls is relevant in areas such as ambient intelligence, assistive technology, healthcare, and sports science. The user's personal safety and comfort must be maintained without restricting their independence and mobility, invading their privacy, and being detrimental to their welfare.

A considerable amount of academic and commercial work has focused on recognizing fall events as accurately and rapidly as possible. Numerous academic attempts have addressed the problem of fall detection, each shedding light on some aspects of this extensive and difficult problem [8–10].

Having successfully detected falls with over 98.5% accuracy in [11], in this article, we consider fall-direction classification to identify the direction of a fall effectively, should a fall be detected. This allows more accurate emergency first response. A fall-direction classification module is developed that operates after the detection of a fall. The overall system with the fall-detection and fall-direction classification modules is illustrated in Fig. 1. The fall-detection module classifies the activity data that it receives into a fall or ADL, employing either heuristics or machine learning (ML) techniques [11,12]. If a fall is detected, the fall-direction classification module is activated. This module extracts features from the raw data and classifies falls into one of four basic directions: forward, backward, right, and left. A total of eight fall-direction classifiers are implemented and a comparative evaluation of their performances is made based on data acquired from a single motion sensor unit worn on the user's waist. The robustness of the module in handling data from falls with undefined directions is investigated. The main contributions of this article are, thus, providing valuable insight to the relative performances of the state-of-the-art fall-direction classification algorithms over a common fall dataset as well as investigating their robustness to falls occurring in undefined directions.

The remainder of this article is organized as follows: Section 2 provides a literature survey on fall classification. The methodology proposed in this work as well as the results of the comparative evaluation of the ML classifiers in terms of confusion matrices, classification metrics, and run times are provided in Section 3. Robustness analysis of the classifiers is provided in Section 4 where the classifiers are modified to handle test instances from falls whose directions are not well defined and do not belong to any one of the four basic fall directions. Finally, Section 5 provides a summary and concluding remarks with possible future research directions.

2. Motivation and related work

We believe that fall-detection systems could offer more than solely detecting falls which involves a binary decision process. Once a fall is detected, identifying the direction of the fall effectively could diminish the likelihood of harm that people from different fall risk groups may receive. The direction of a fall can contain various cues regarding the nature of the fall. Forward falls are mostly caused by tripping, whereas lateral falls can be due to loss of consciousness, and backward falls can be a result of slipping. The direction of a fall can also be of value in an emergency scenario to assess the situation and apply appropriate treatment. That is, knowing the direction of a fall can lead the emergency response team to a more accurate diagnosis of the state of the subject. Indeed, lateral falls are more likely to cause limb and neck injuries whereas backward falls are often associated with the head coming into contact with the ground and, therefore, are more dangerous. A forward fall is usually more controlled than a lateral or backward fall because people are more experienced to use their arms in the forward position. A wearable fall-detection system usually already includes motion sensor units which can supply adequate information for recognizing the directions of falls. Therefore, once a fall is detected, a fall-detection system can provide invaluable fall-direction information without requiring any additional hardware. As an example, typical acceleration, angular velocity, and magnetic field signals of falls corresponding to the four basic directions are provided in Fig. 2. The fall-direction classification algorithms employed in this study rely on the data acquired from a motion sensor unit worn on the subject's waist, which are sufficient for effective fall-direction classification.

Research has been conducted on classifying the type of falls in both the ambient assisted living field [13,14] and wearable sensors field [15–24] albeit much limited in quantity compared to the vast literature on fall detection. Almost all of these fall-classification algorithms are activated after detecting a fall, except that Choi et al. [18] integrate fall-direction classification into the fall-detection algorithm. They classify the activities of the subjects into seven activity types: three ADLs and four falls towards the four basic directions — forward, backward, right, and left. Using a novel feature selection method and naïve Bayesian (NB) decision making, they are able to classify the activity types of 670 data instances in their dataset with 99.4% accuracy.

Although most wearable-sensor-based fall-direction classification studies classify falls into the four basic directions mentioned above, references [16] and [25] classify falls into three directions: forward, backward, and lateral. Utilizing a sensor node comprising a tri-axial accelerometer and a barometric pressure sensor at the waist of the subject and using a heuristic algorithm employing the azimuth angle of the subject, they were able to achieve a 94.12% fall-direction classification accuracy on a dataset consisting of 119 fall instances. Dinh and Struck [19] also consider the same three

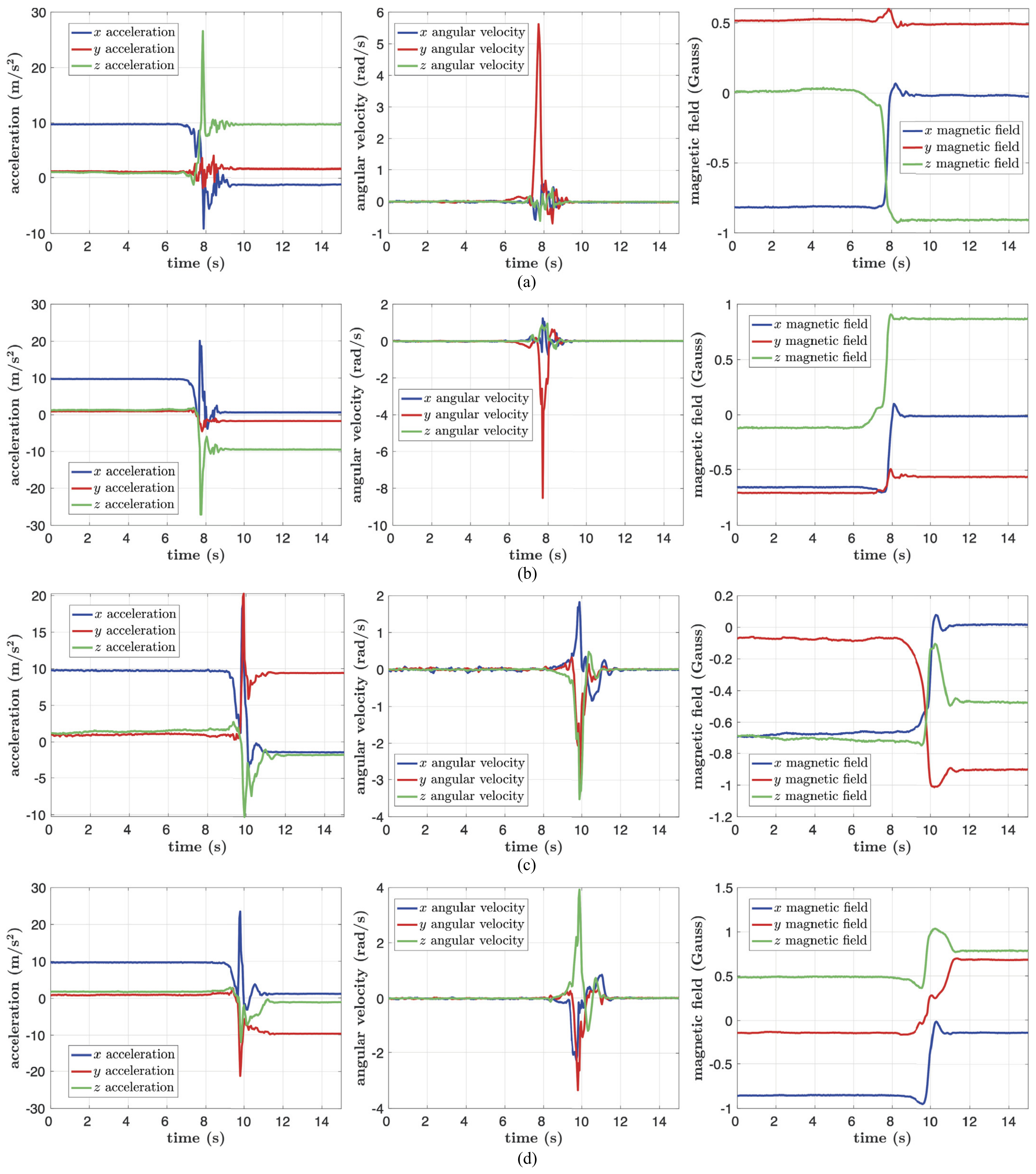

Fig. 2. Acceleration, angular velocity, and magnetic field signals of a) forward, b) backward, c) right, and d) left falls. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

fall types with the addition of a collapse type. Attempting to capture four types of falls, they use accelerometer data collected from the waist of the subject, and transformed from Cartesian to spherical coordinates. With an algorithm that utilizes fuzzy logic and ANNs, they achieve an average fall-direction classification sensitivity of 94% over all types of falls, on a dataset collected from five subjects and comprising a total of 100 fall instances.

Reference [26] categorizes falls into a broader set: forward, backward, right-side, left-side, blinded-forward, and blinded-backward. The authors of [27] group fall types into six more specific categories which are fall forward and lie on the floor, fall backward and lie on the floor, fall lateral right and lie on the floor, fall lateral right and sit up from the floor, fall lateral left and lie on the floor, fall lateral left and sit up from the floor.

Majority of the studies on wearable-sensor-based fall classification use motion sensor units worn on the waist, chest, or the back of the subject; however, Tao et al. [20] propose a fall-direction classification system with four force-sensor resistors in both the right and left shoe insoles of the subjects. They attempt to classify falls into four directions as Choi et al. [18] do and achieve a classification rate of 75% on a very limited dataset of only 12 fall instances.

The study by Albert et al. [17] attempts to classify falls into the four basic directions mentioned above using the accelerometer data from a mobile phone affixed to the back of the subjects. Their dataset for fall classification consists of a total of 223 fall instances collected during laboratory experiments involving 15 participants. They compare five different classifiers — SVM, sparse multi-nomial logistic regression (SMLR), NB, *k*-NN, and DTC — with a large feature set of 178 features in total, without using any feature reduction technique. They evaluate these classifiers on their own dataset with two different cross-validation techniques: 10-fold and subject-based cross validation. SMLR yields the best classification performance with a classification accuracy of 99.6% with both types of cross validation.

Pannurat et al. [21] present a hybrid fall-monitoring algorithm that can detect different phases of a fall as well as classify ADLs using a wearable accelerometer. In particular, they implement a rule-based algorithm to detect the pre-impact, impact, and post-impact phases of a fall and a ML-based ADL classifier to confirm falls as well as detect the occurrence of a particular type of fall: syncope. On a dataset comprising 16 healthy subjects performing 14 types of fall and 12 types of ADL, their method is able to achieve up to 99% accuracy.

The authors of [14] use depth camera data to detect and classify falls of subjects with a walking support system. Employing a Hidden Markov Model (HMM)-based technique, they are able to classify the states of the user, which correspond to three types of ADLs and five fall directions: the four basic directions and downward (collapse). Upon detection and classification of a fall, they control the motion of the walking support system to achieve fall prevention. On a dataset collected from four healthy subjects, their method is able to reach a state classification accuracy of 81.0%.

Kwon et al. [22] address the problem of fall classification after the detection of a fall, using a chest-worn inertial measurement unit (IMU). Using a temporal signal angle measurements algorithm alongside three ML-based algorithms, their method can classify five different types of falls: incorrect weight shifting, trip, bump, loss of support, and collapse. The authors evaluate their method on a dataset collected from seven volunteers to show that their method presents accuracy values of up to 93.3%.

In [23], the authors present a comparative study of using different cumulant features extracted from waist-worn accelerometers as well as four ML methods for fall detection and fall-direction classification into four basic directions. Processing a dataset collected from six healthy subjects performing five ADLs and four types of falls, they show that with the use of cumulant features, SVM classifier yields better results than DTC, NB, and ANN.

Andò et al. [24] propose a system that classifies various ADLs as well as falls using accelerometer and gyroscope data from a waist-worn smartphone. Employing a threshold-based algorithm and multisensor data fusion, their method is able to achieve almost perfect sensitivity and specificity values on a dataset collected from 10 healthy subjects.

In recent years, the availability of a new publicly available dataset of ADL and fall activities has sped up the development of fall-classification systems. The multimodal UP-Fall Detection Dataset [28] includes data collected from 11 healthy subjects via multiple wearable sensors, cameras, and context-aware sensors while performing five types of falls and six types of ADLs. The availability of this dataset led to the organization of "2019 Challenge UP — Multimodal Fall Detection" competition [29]. Indeed, this competition stirred significant interest in fall detection and classification, as evident in the recently published book [30]. For instance, Espinosa et al. [31] employ convolutional neural networks (CNNs) to detect and classify falls via video data in this dataset. Upon optimizing the CNN architecture through cross validation, their method classifies different activities in the dataset with 82% accuracy.

Despite abovementioned work, there is still need for further research in the fall-classification field, especially in comparison to the broader field of fall detection. Existing literature not only lacks in providing effective fall-classification systems, but would also benefit from an exhaustive and fair evaluation process for various classification methods. In this study, eight state-of-the-art ML classifiers are implemented for classifying falls into four basic directions. These classifiers are then evaluated on a common extensive dataset comprising 1600 fall instances using subject-based cross validation (which is more challenging than using randomized folds [32,33]) with parameter optimization. Their performances are then compared in terms of confusion matrices, performance metrics, and run times. Furthermore, this study contributes to the fall-direction classification area by conducting robustness analysis and modifying four of the eight ML classifiers to avoid making decisions in the presence of data from an unknown class. This way, robustness to unknown classes is achieved. To our knowledge, previous studies on fall-direction classification do not consider such evaluation of their fall-direction classification algorithms and do not address the problem in such depth to achieve as high performance. Moreover, the datasets used in previous studies are rather limited. In summary, this study adds on the existing work in the literature by thoroughly evaluating and comparing eight state-of-the-art ML classifiers on a large dataset and conducting a robustness analysis.

3. Fall-direction classification system

In this section, first the acquisition of the original dataset that contains both fall and non-fall activity types is described. Only the fall-type activities are considered for fall-direction classification. Experimental methodology and the feature extraction process are outlined. State-of-the-art ML classifiers that are employed for fall-direction classification are presented. The classifiers are evaluated and compared in terms of five performance metrics and their run times.

3.1. Description of the dataset

The dataset used in this study is made publicly available [34] and was originally acquired by our research team [12] to evaluate different ML algorithms for effective and reliable fall detection. It was also used in the studies reported in [35–37]. The dataset was

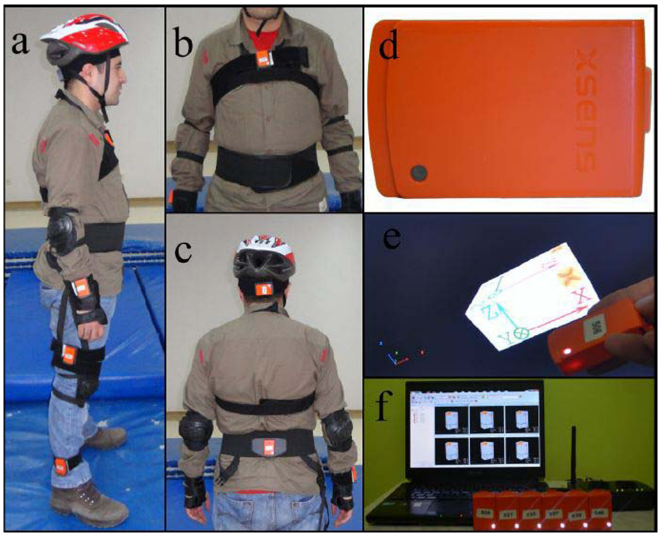

Fig. 3. a)-c) Configuration of the sensor units on the subject's body, d) MTw sensor unit, e) axes of a sensor unit, f) connection to a PC and the interface [12].

collected by following the protocols proposed by Abbate et al. [38] with the approval of Erciyes University Ethics Committee to conduct experiments with human participants. A total of 16 young and healthy subjects performed the scripted activities with their informed written consent. The seven female subjects had an average age of 21.5, an average weight of 58.5 kg, and an average height of 169.5 cm, whereas the nine male subjects averaged to 24 years in age, 67.5 kg in weight, and 172 cm in height. The subjects performed the activities on a soft floor mat, with protective equipment on their head, wrists, elbows, and knees in order to avoid injuries (Fig. 3 a)-c)).

Six wireless MTw sensor units from Xsens Technologies [39] were affixed to the head, waist, chest, right wrist, right thigh, and right ankle of each subject. Each of these units consists of three tri-axial devices (an accelerometer, a gyroscope, and a magnetometer with respective working ranges of ± 120 m/s^2, $\pm 1200°$/s, and ± 1.5 Gauss) and an atmospheric pressure sensor with an operating range of 300–1100 hPa. Fig. 3 illustrates the configuration of the sensor units on the subject's body as well as the axes of each sensor unit. Data were collected at a sampling frequency of 25 Hz and sent to a PC via a ZigBee connection for storage.

Each subject performed five different executions of 16 non-fall activities (ADLs) and 20 fall activities. A broad span of activities was selected, in agreement with the guidelines in [38], to capture most of the real-world activities so that the evaluation of developed algorithms can produce realistic outcomes. Non-fall and fall activities are listed separately, with a brief description of each activity, in Tables 1 and 2. Common ADLs as well as near-fall activities were included in the dataset in order to construct a genuine representation of real-life activities. Included fall activities also embrace a wide variety of fall types that can be encountered in real-life scenarios. A vast dataset comprising 2880 trials was obtained: 1280 non-fall and 1600 fall trials. Each trial of duration 10–15 s was recorded and stored separately.

Although the dataset contains data from six different sensor units worn by the subjects, only the data recorded by the waist sensor unit are used throughout this study. This results from an attempt to render the proposed algorithms more feasible to embed in a hardware system, because not only the cost of a system requiring six sensor units would be considerably high, such a system would also be obtrusive to the user. The waist is shown to be a commonly used region of the body, as well as the chest, to capture motion signals from the subjects and yields the highest accuracy for fall detection compared to the head and the limbs [35,36].

Table 1
Brief descriptions of the non-fall activities (ADLs) in the dataset.

No.	Brief description	Type
1	walking forward	non-fall
2	walking backward	non-fall
3	running	non-fall
4	squatting and then standing	non-fall
5	bending at about 90°	non-fall
6	bending to pick up an object	non-fall
7	walking with a limp	non-fall
8	stumbling with recovery	non-fall
9	ankle sprain	non-fall
10	coughing/sneezing	non-fall
11	standing to sitting on a hard surface (chair)	non-fall
12	standing to sitting on a medium surface (sofa)	non-fall
13	standing to sitting on air	non-fall
14	standing to sitting on a soft surface (bed)	non-fall
15	standing to lying on bed	non-fall
16	lying on bed to standing	non-fall

Table 2
Brief descriptions of the fall activities in the dataset, with their directions.

No.	Brief description	Type/Direction
17	standing to falling forward to the floor	fall/forward
18	standing to falling forward to the floor with arm protection	fall/forward
19	standing to falling on knees	fall/undefined
20	standing to falling on knees and then lying down	fall/forward
21	standing to falling forward with quick recovery	fall/forward
22	standing to falling forward with slow recovery	fall/forward
23	standing to falling forward, ending in right lateral position	fall/forward
24	standing to falling forward, ending in left lateral position	fall/forward
25	standing to falling down on the floor, ending sitting	fall/undefined
26	standing to falling backward, ending lying	fall/backward
27	standing to falling backward, ending in right lateral position	fall/backward
28	standing to falling backward, ending in left lateral position	fall/backward
29	standing to falling on the right side, ending lying	fall/right
30	standing to falling on the right side with recovery	fall/right
31	standing to falling on the left side, ending lying	fall/left
32	standing to falling on the left side with recovery	fall/left
33	from lying, rolling out of bed and falling on the floor	fall/undefined
34	standing on a podium to forward fall on the floor	fall/forward
35	syncope — standing to falling vertically	fall/undefined
36	syncope fall, slowly slipping off a wall on the side	fall/undefined

It must be noted that the dataset was collected mostly from young and healthy subjects (as well as some middle-aged ones), performing simulated falls in a laboratory setting with protective equipment. Although it would have been preferable to have real fall data from elderly subjects or subjects from certain disease groups, current subject profiles and fall categories were chosen with the aim of gathering an extensive dataset [12]. Real-life fall data or data involving elderly subjects are extremely limited and difficult to collect because of the fragility of the elderly and the long waiting times. We believe this situation should not hinder the results obtained in this article because it has been reported in [40] that real-life falls by older people bear similar characteristics to those of simulated falls.

A data instance is labeled as non-fall if the index of the activity is less than or equal to 16 (Table 1) or as fall if the index of the activity is greater than 16 (Table 2). Because only fall-type activities can be classified into four basic directions, only the activities with index greater than 16 (Table 2) are included in this evaluation. In other words, assuming that the fall-detection algorithm detected a fall, only the fall-type activities in the dataset will be used for fall-direction classification. The 20 fall types considered within the scope of this article (Table 2) can be grouped into five classes corresponding to the four basic directions and a class with undefined directions. In this part of the study, we only use the 15 fall types, corresponding to the four well-defined directions, in the comparative evaluation. Specifically, there are eight forward, three backward, two right, and two left fall types. Remaining five fall types, with undefined directions, are reserved to be used in Section 4.

Since each of the 15 types of falls, with well-defined directions, is performed five times by each of the 16 subjects, this sums up to a total of 1200 directional fall instances: 640 forward, 240 backward, 160 right, and 160 left fall instances.

3.2. Preprocessing

Note that the sizes of the data belonging to the four classes are not comparable; that is, the size of the forward fall class is almost three times as large as the size of the backward one and even four times as large as the sizes of the right and left fall classes. This imbalance in class sizes would corrupt the performance of certain classifiers. For instance, such a distribution of class sizes is likely to cause ANN to learn the weights with a bias towards the forward fall class because there are more instances from that class and every data instance updates the weights once in each epoch in online backpropagation. To avoid this problem, every data instance in backward, right, and left fall classes are replicated and added to the dataset until the sizes of the classes become comparable; that is, the data from the backward fall class are copied twice while the data from the right and left classes are replicated thrice and these replications are added to the dataset to make the sizes of the four classes more or less even (640, 720, 640, and 640, respectively). Finally, the order of the data instances is rearranged with a random permutation.

As part of the preprocessing, raw data acquired from the motion sensor unit attached to the waist of the subject are filtered using three-point median filtering in order to eliminate the high-frequency noise components of the signals.

3.3. Feature extraction

After preprocessing, 27 simple features are extracted from the whole duration of the recording of each trial at the sensor unit worn on the waist: minima, maxima, and the mean values of the accelerometer, gyroscope, and magnetometer data in the x, y, and z directions. The feature set consisting of these $27\,(= 3$ sensor types $\times\ 3$ axes $\times\ 3$ features per axis) features is normalized to have zero mean and unit standard deviation.

3.4. Description of the ML classifiers considered

Eight state-of-the-art ML classifiers are implemented for comparative evaluation of their performances based on the dataset described above for fall-direction classification. These are, namely, Bayesian decision making (BDM), least squares method (LSM), k-nearest neighbor classifier (k-NN), artificial neural networks (ANNs), support vector machines (SVMs), decision-tree classifier (DTC), random forest (RF), and adaptive boosting or AdaBoost (AB). Some of these classifiers have parameters that need to be tuned for the best performance of the classifier [41]. These parameters are often optimized via cross validation. In this study, we employ subject-based cross validation, which is described in the next section. The classifiers BDM, LSM, and DTC do not have any parameters to be optimized whereas the parameters of the remaining five classifiers (k-NN, ANN, SVM, RF, and AB) are optimized through a grid search. Separate grid searches for each parameter are conducted and the best results are provided and used. Brief descriptions of the ML classifiers are given below, together with information on the parameter selection of five of the classifiers, indicating the intervals of the grid searches for the optimization of their parameters. Ranges of optimal parameter values for various folds of cross validation are also provided. More detail on these classifiers can be found in [41–43].

BDM BDM is based on fitting multi-variate Gaussian distributions to the data from each class — in our case, four classes — and obtaining the mean vectors and covariance matrices of these Gaussian distributions. Once these parameters are obtained, the training process is completed and the testing phase continues with calculating the *a posteriori* probabilities of each test data for each class and assigning the test data to the class that gives the largest *a posteriori* probability. No parameters need to be selected for BDM.

LSM In the training phase of classification with the LSM, the mean vectors of the data from each of the four classes are calculated and stored. In testing, sums of squared distances of a test instance to the mean vectors of each class are calculated and the test instance is assigned to the class whose mean is the closest to it in the feature space. LSM has no parameters to be selected.

***k*-NN** Training of the k-NN classifier comprises the storage of the training feature vectors. In the testing phase, the Euclidean distances of a test feature vector to every training feature vector are calculated and the nearest k training feature vectors are selected. The test feature vector is then assigned to the most frequently occurring class among these k training feature vectors. The parameter k is optimized through a grid search, taking integer values from 1 to 50 and an optimal k value of 1 is obtained.

ANN ANNs are networks of units called neurons, arranged in multiple layers [44]. In this study, an ANN with only a single hidden layer of neurons is implemented for fall-direction classification. The input layer (the very first layer) consists of the input neurons, each of which takes the value of a feature in the feature vector as input; therefore, the number of neurons in the input layer is exactly the same as the number of features used (27). Each input-layer neuron is connected to the neurons in the hidden layer (the layer in the middle) with unique weights. At each neuron in the hidden layer, a non-linear activation function is applied to the weighted sum of the input neurons using the connection weights from each of the input neurons to that hidden-layer neuron. A similar connection is then made from the hidden layer to the output layer (the third layer). In this study, the activation function used in all hidden and output neurons is a sigmoid function of the form $g(x) = (1 + e^{-x})^{-1}$, where x is the weighted sum of the outputs of the neurons in the previous layer.

The ANN implemented in this study has four output neurons which use the same sigmoid activation function as the hidden-layer neurons and can display output values ranging from 0 to 1. When a data instance is given as input to the ANN and the non-linear activation function is applied to the weighted sum of the outputs of all neurons in the hidden layer, the output of the ANN is obtained. The values obtained at each one of these four output neurons indicate the confidence levels of that data instance belonging to each one of the classes. Finally, the feature vector is assigned to the class with the highest confidence level. The connection weights are initialized with a uniform random distribution between 0 and 0.2, and training is performed using iterative online (stochastic) backpropagation algorithm with a learning rate of 0.3. The algorithm is terminated when there is not a significant reduction in the average errors of all training data. While learning the weights corresponds to determining the weights of the ANN, testing is equivalent to multiplying the feature values of the test data with the obtained weights to identify its label. The number of hidden-layer neurons is optimized through a grid search where integer values from 1 to 50 are considered. Optimal values for this parameter range between 19 to 46.

SVM SVMs were originally designed for binary classification and are based on creating a hyperplane in the feature space that has the maximum margin between two classes in the training data. However, here, we have a multi-class problem with four classes. It has been shown in [45] that the performances of the one-against-one and one-against-all variants of multi-class SVM are comparable but the training time of the former is shorter. Therefore, we use the one-against-one approach. A kernel function can be utilized to transform the original feature space to another space, which can prove to be useful when the training data are not linearly separable in the original feature space. In this study, Gaussian Radial Basis Function kernel, $f_{\mathrm{GRBF}}(\vec{x}, \vec{y}) = e^{-\gamma||\vec{x}-\vec{y}||^2}$ is used, where γ is the kernel parameter and $\vec{x}$ and $\vec{y}$ are two feature vectors. After the transformation to another space, an optimization process is conducted to find the hyperplane that maximizes the margin between the data of each pair of classes with the penalty parameter C. MATLAB's LIBSVM toolbox is used for the implementation of SVM. The parameters γ and C are optimized through a grid search where both are ranged from 10^{-5} to 10^{5} on a logarithmic scale. Optimal values in the intervals $[10^{-2}, 10^{1}]$ and $[10^{-3}, 10^{-1}]$ are obtained for these two parameters, respectively.

DTC DTC is an ML classifier based on decision stumps: a simple structure applying a threshold to one feature, that is, comparing the value of that feature to a threshold to determine if it is higher or lower. At each node of a DTC, there is a decision stump to produce two branches which are then connected to two other nodes. Final nodes of a DTC, which do not utilize stumps (a feature and a threshold value) and thus do not lead to any more nodes, are called leafs. At the leaf nodes of a DTC, the decision about the estimated class of the data instance is made. The features and the thresholds to be used in the nodes of a DTC are selected using a splitting criterion in a greedy fashion; that is, the split that yields the optimal value of a specific criterion is selected at each node without considering the optimality of the overall tree structure. In this study, Gini impurity criterion is selected as the splitting criterion, which is a measure of the frequency of incorrectly classifying a randomly chosen data instance when it is classified according to the distribution of the classes in that split. The value of this criterion is zero when the nodes created by that split are pure, that is, all training instances in one leaf belong to a specific class. Building the tree structure from the training set until all leafs are pure results in overfitting to the training set. Therefore, different pruning techniques are employed to prevent trees having an excessive depth and number of nodes. In this study, prepruning technique is employed to prevent DTC from overfitting, which involves rejecting to add any more nodes when building the tree if the size of a node is sufficiently small, even though the node is not pure. DTC does not have any parameters to optimize.

RF RF is a special classifier employing bagging (bootstrap aggregating) technique: an ensemble learning technique which involves taking a large number of subsets from the training set and combining the results of the classifiers that are trained by each of these subsets to obtain the final decision. As the name suggests, RF is based on training a large number of DTCs with the randomly generated subsets of the training set and combining the results of each classifier to end up with a final decision. Because training a large number of DTCs is computationally intensive, a randomly selected subset of the features is used in their training. Using the decisions of a large number of decision trees trained on subsets of the training set, RF eliminates the problem of overfitting that is usually pronounced with DTCs. Two parameters are optimized: the number of trees to be trained is ranged from 40 to 240 to observe that the optimal parameter values are between 120 and 200, and the number of features to be used in the training of trees is varied from 1 to 27 to obtain optimal values between 1 and 10.

AdaBoost AdaBoost is a boosting algorithm, that is, it utilizes the whole training set to iteratively train a large number of weak learners. At the first iteration, a weak learner is trained where all instances in the training set have equal weights. Afterwards, at each iteration, the weights of the training instances that are incorrectly classified by the previous weak learner are increased to train a new weak learner with the updated weight distribution of the training set. The AB algorithm utilizes decision stumps as weak learners, and the number of weak learners to be used is a parameter that needs to be optimized. The number of weak learners is varied from 50 to 250 to obtain optimal parameter values between 100 and 190.

3.5. Description of subject-based cross validation

Subject-based multi-fold cross validation with parameter optimization is employed in evaluating the performances of fall-direction classification algorithms. This is preferred over randomly partitioned multi-fold cross validation in this work because it is attempted to make a more realistic evaluation of the algorithms where the data of the test subjects are not used in the determination of parameters. It is highly likely that any user of such a fall-direction classification system will not have contributed to this dataset, if any algorithm presented here were to be embedded in a hardware system for real-world use. Therefore, by excluding all of the data of the test subjects from the training data, we seek to avoid optimistic results that may be caused by the correlation among the data from the same subject. This selection, in turn, causes relatively high variations in the performance metrics of the eight different training and test set combinations.

Using the record of the subject index for each data instance, the fall dataset is partitioned into eight folds in a subject-based fashion: seven folds of data from one male and one female subject each, and one fold of data from two male subjects. A pseudocode of the subject-based cross validation with parameter optimization is given in Algorithm 1. In an outer loop, each one of these eight folds is separated to be used as the test set whilst the remaining seven folds are combined to form the training set, which will be used for parameter optimization and training. At each iteration of an intermediate loop, a grid mesh of parameter values is swept and specific parameter value(s) are considered. While performing parameter optimization, sufficiently large intervals are considered for the parameters to avoid obtaining optimal parameter values on the limits of the intervals; furthermore, the intervals of parameter sweep are updated, should such a situation be encountered. At every iteration of the intermediate loop, training data are divided into seven sub-folds of data from the same combinations of subjects that formed it in the first place. In an inner loop, each one of the seven sub-folds is taken as the test set for the parameter selection (called sub-test set to avoid confusion) and the remaining six sub-folds are combined to obtain the training set for the parameter selection (similarly, called sub-training set). In every iteration of the inner loop, the classifier is trained using the corresponding parameter value(s) and sub-training set combination and tested using the corresponding parameter value(s) and sub-test set combination. After the inner loop is completed, the average of the

Algorithm 1 Pseudocode of the cross validation with parameter optimization.

```
Divide the whole dataset into eight folds with respect to the subjects
for Each Fold {Outer Loop} do
    This fold is the test set
    Remaining seven folds are combined to obtain the training set
    Best Accuracy = 0
    for Each Value of the Parameter(s) {Intermediate Loop} do
        Divide the training set into seven sub-folds with respect to the subjects
        for Each Sub-Fold {Inner Loop} do
            This sub-fold is the sub-test set
            Remaining six sub-folds are combined to obtain the sub-training set
            Train the classifier with the sub-training set and the parameter value(s)
            Test the classifier with the sub-test set and the parameter value(s)
            Store accuracy
        end for
        Find the average accuracy over all seven iterations of the Inner Loop
        if Average Accuracy > Best Accuracy then
            Optimal Parameter Value(s) = This Parameter Value(s)
            Best Accuracy = Average Accuracy
        end if
    end for
    Train the classifier with the training set and the optimal parameter value(s)
    Test the classifier with the test set and the optimal parameter value(s)
    Store the confusion matrix
end for
```

sub-test accuracies of all seven runs is calculated and stored. At the end of the intermediate loop, the parameter value(s) that yield the highest average sub-test accuracy is selected. At the end of one iteration of the outer loop, the classifier is trained using the corresponding training set (seven of eight folds) and the selected parameter value(s) and tested with the corresponding test set (one of eight folds). The confusion matrix for each of these training and test combinations obtained in every iteration of the outer loop is stored. At the very end of the subject-based cross validation with parameter optimization, eight confusion matrices are obtained.

3.6. Comparative evaluation of the ML classifiers

In this section, some basic definitions and the methodology of the comparative study that is undertaken are given, followed by its results.

Fall-direction classification is a multi-class decision problem that requires a decision on whether a detected fall is in one of the considered directions. While testing the algorithms, for each fall direction, we may encounter four different cases:

- true positive (TP): the fall is in a particular direction and the classifier correctly detects that direction
- false positive (FP): the fall is not in a given direction but the classifier incorrectly detects that direction
- true negative (TN): the fall is not in a given direction and the classifier correctly does not detect that direction
- false negative (FN): the fall is in a particular direction but the algorithm incorrectly does not detect that direction

A FP corresponds to a *false alarm* and a FN corresponds to a *missed detection* in radar terminology which is a two-class (binary) classification scenario. Since fall-direction classification is a multi-class decision problem, one needs to define TP, FP, TN, FN for each class separately. Thus, we denote these four cases for specific classes with subscripts. For instance, TP_F, TP_B, TP_R, and TP_L denote the TP cases for forward (F), backward (B), right (R), and left (L) fall classes, respectively.

Having described these four types of outcomes for each class, a confusion matrix can be constructed as in Fig. 4, where F, B, R, and L represent the total numbers of true falls in the forward, backward, right, and left directions, respectively. Furthermore, F′, B′, R′, and L′ represent the total numbers of detected falls in the same

		classified direction				
		forward	**backward**	**right**	**left**	**total:**
true direction	**forward**	FF′	FB′	FR′	FL′	F
	backward	BF′	BB′	BR′	BL′	B
	right	RF′	RB′	RR′	RL′	R
	left	LF′	LB′	LR′	LL′	L
	total:	F′	B′	R′	L′	

Fig. 4. A 4 × 4 confusion matrix for fall-direction classification.

four directions, respectively. Specifically, based on the definitions of TP, FP, TN, FN and the confusion matrices in Fig. 4, the following relationships are derived for the forward fall (F) class:

$$
\begin{aligned}
TP_F &= FF' \\
FP_F &= F' - FF' = BF' + RF' + LF' \\
TN_F &= B - BF' + R - RF' + L - LF' \\
FN_F &= F - FF' = FB' + FR' + FL'
\end{aligned} \tag{1}
$$

These can similarly be defined for the other fall classes.

Based on the aforementioned fundamental definitions, five performance metrics (accuracy, precision, sensitivity, specificity, and F-measure) are calculated. In particular, accuracy of a fall-direction classifier is calculated as the ratio of correctly classified falls to all fall instances:

$$
\text{Accuracy} = \frac{FF' + BB' + RR' + LL'}{F + B + R + L} \tag{2}
$$

Moreover, precision, sensitivity, and specificity metrics are calculated separately for each class using the definitions of the four outcomes above. Below, the definitions of these metrics are given for the forward fall (F) class and their definitions for the other three classes follow similarly:

$$
\begin{aligned}
\text{Precision}_F &= \frac{TP_F}{TP_F + FP_F} = \frac{FF'}{F'} \\
\text{Sensitivity}_F &= \frac{TP_F}{TP_F + FN_F} = \frac{FF'}{F} \\
\text{Specificity}_F &= \frac{TN_F}{TN_F + FP_F} = \frac{B - BF' + R - RF' + L - LF'}{B + R + L}
\end{aligned} \tag{3}
$$

Obviously, there is an inverse relationship between sensitivity and specificity metrics. The FP and FN ratios can be easily obtained in terms of specificity and sensitivity: FP ratio = 1 − Specificity and FN ratio = 1 − Sensitivity. This indicates a compromise between an *aggressive* algorithm that may incorrectly classify many falls into a particular direction, and a *finicky* algorithm that may mistakenly *not* classify many falls into a particular direction.

After computing the class-based precision, sensitivity, and specificity values, the values of these metrics for the overall fall-direction classification algorithm are calculated as the arithmetic

Table 3
Confusion matrices in fall-direction classification.

BDM:
$$\begin{array}{cc} & \text{classified} \\ \text{true} & \begin{bmatrix} 640 & 0 & 0 & 0 \\ 0 & 720 & 0 & 0 \\ 0 & 0 & 640 & 0 \\ 0 & 0 & 0 & 640 \end{bmatrix} \end{array}$$

SVM:
$$\begin{array}{cc} & \text{classified} \\ \text{true} & \begin{bmatrix} 629 & 0 & 9 & 2 \\ 0 & 720 & 0 & 0 \\ 0 & 4 & 636 & 0 \\ 0 & 0 & 0 & 640 \end{bmatrix} \end{array}$$

LSM:
$$\begin{array}{cc} & \text{classified} \\ \text{true} & \begin{bmatrix} 616 & 0 & 15 & 9 \\ 0 & 696 & 3 & 21 \\ 0 & 8 & 632 & 0 \\ 0 & 0 & 0 & 640 \end{bmatrix} \end{array}$$

DTC:
$$\begin{array}{cc} & \text{classified} \\ \text{true} & \begin{bmatrix} 621 & 5 & 1 & 13 \\ 3 & 687 & 12 & 18 \\ 4 & 16 & 620 & 0 \\ 12 & 32 & 0 & 596 \end{bmatrix} \end{array}$$

k-NN:
$$\begin{array}{cc} & \text{classified} \\ \text{true} & \begin{bmatrix} 628 & 0 & 9 & 3 \\ 0 & 720 & 0 & 0 \\ 0 & 0 & 640 & 0 \\ 0 & 0 & 0 & 640 \end{bmatrix} \end{array}$$

RF:
$$\begin{array}{cc} & \text{classified} \\ \text{true} & \begin{bmatrix} 640 & 0 & 0 & 0 \\ 0 & 717 & 0 & 3 \\ 0 & 4 & 636 & 0 \\ 0 & 0 & 0 & 640 \end{bmatrix} \end{array}$$

ANN:
$$\begin{array}{cc} & \text{classified} \\ \text{true} & \begin{bmatrix} 616 & 4 & 13 & 7 \\ 0 & 717 & 0 & 3 \\ 0 & 4 & 636 & 0 \\ 4 & 0 & 0 & 636 \end{bmatrix} \end{array}$$

AB:
$$\begin{array}{cc} & \text{classified} \\ \text{true} & \begin{bmatrix} 619 & 0 & 1 & 20 \\ 0 & 717 & 0 & 3 \\ 4 & 24 & 612 & 0 \\ 4 & 4 & 0 & 632 \end{bmatrix} \end{array}$$

mean of their class-based values. Therefore, the overall precision, sensitivity, and specificity of the fall-direction classification algorithm are given as:

$$\begin{aligned} \text{Precision} &= \frac{1}{4}\left(\text{Precision}_\text{F} + \text{Precision}_\text{B} + \text{Precision}_\text{R} + \text{Precision}_\text{L}\right) \\ \text{Sensitivity} &= \frac{1}{4}\left(\text{Sensitivity}_\text{F} + \text{Sensitivity}_\text{B} + \text{Sensitivity}_\text{R} \right. \\ &\quad \left. + \text{Sensitivity}_\text{L}\right) \\ \text{Specificity} &= \frac{1}{4}\left(\text{Specificity}_\text{F} + \text{Specificity}_\text{B} + \text{Specificity}_\text{R} \right. \\ &\quad \left. + \text{Specificity}_\text{L}\right) \end{aligned} \tag{4}$$

As the last performance metric, F-measure of the fall-classification algorithm is obtained by multiplying the harmonic mean of precision and sensitivity metrics by two:

$$F\text{-measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}} \tag{5}$$

Note that we have used class-based precision, sensitivity, and specificity metrics averaged over the four classes since the sizes of the classes are comparable, albeit not exactly the same. The F-measure metric in Equation (5) provides a combined performance measure of a system using the two metrics that have an inverse relationship.

Eight 4×4 confusion matrices that are obtained at each iteration of the outer loop of the cross-validation process are summed up and shown in Table 3 to reflect the overall performance of each classifier. Upon inspecting this table, one can see that forward falls are mostly confused with right and left falls by LSM, k-NN, ANN, SVM, DTC, and AB classifiers. Furthermore, LSM, ANN, RF, and AB can classify falls from the backward direction class into the left direction class. DTC further confuses falls in the backward direction with those to the right and to the left. Although some misclassifications occur, no major problems are observed in Table 3 other than those mentioned.

Performance metrics of the eight ML classifiers are calculated separately for each of the four classes as described above and are averaged over all four classes. The values of the five performance metrics for the eight ML classifiers are tabulated in Table 4 in mean plus/minus one standard deviation format. It is evident from this table that, with the selected feature set, BDM achieves perfect classification on the fall dataset with subject-based cross validation. This result is highly satisfactory, especially considering the size of the dataset; that is, BDM classifies the directions of all fall instances correctly without needing any parameter optimization and any complex features. The performances of three other classifiers are also very good. Namely, the k-NN classifier follows BDM with the average values of all performance metrics being above 99.5%. Similarly, SVM and RF yield performance metrics above 99.4%, only within 0.1% of k-NN. Although their performances are not comparable to the top three ML classifiers, ANN and LSM produce performance metrics above 98.6% and 97.8%, respectively. Among the eight classifiers, the use of DTC results in the lowest average value and the largest standard deviation for each of the performance metrics. The overall high performance of the ML classifiers considered here indicates that the extracted features bear adequate information for reliable and accurate fall-direction classification.

3.7. Run-time analysis of the ML classifiers

All eight ML classifiers are trained and tested on MATLAB version R2015a installed on a computer with Intel® Core™ i5-3230M CPU running at 2.60 GHz, 4.00 GB RAM, and Windows 7 Home Premium 64-bit operating system while no other external application or program is running.

To determine the run times of the algorithms, subject-based cross validation of each algorithm is executed without parameter optimization. First, the whole dataset is partitioned into eight folds of data based on subject pairs as before, and in a loop, each one of these eight folds is used as test data while the remaining seven are combined to obtain the training data. At each iteration of this loop, the classifiers are trained using the corresponding training data and fixed parameter value(s) and then tested with the corresponding test data and the same parameter value(s). This process is equivalent to executing only the outer loop in Algorithm 1. Both the training and testing times of the classifiers are recorded at each iteration. Training, testing, and total run times of the eight ML classifiers run for fall-direction classification are given in Table 5 in mean plus/minus one standard deviation format for the classification of a single data instance. The total run time is the sum of the training and testing times.

It can be observed from Table 5 that the total run time of the LSM classifier is significantly lower than those of the other algorithms. In fact, it is nearly as low as one tenth of the next fastest algorithm, k-NN. Note that k-NN is the only classifier that does not require any training. Average total run times of k-NN and SVM are on a par with each other and significantly lower than the total run times of BDM, ANN, DTC, RF, and AB. Although the total run time of ANN is over 70 times more than that of LSM, it corresponds to an average of 2.72 ms for the classification of a single data instance. We believe that this run time is feasible for real-world operation, especially considering that such an algorithm will be pre-trained in such a scenario. The training times are acceptable for all classifiers since training will be executed only once. However, the training times would increase if more classes are added. Despite displaying a high degree of performance, RF displays a total run time that is nearly as much as four times that of ANN. Furthermore, AB requires even more time to train and test a single data instance, because it uses all 27 features in each weak classifier that it trains. Considering the relatively large total run time of DTC, the run-time results of these two algorithms that use a large number of DTCs are expected to be slow.

The authors of a fall-detection and localization study [46] compare the average training and testing times (as well as the performance metrics) of k-NN, SVM, and Gaussian Process (GP) classifiers implemented on Raspberry Pi programmable boards. In that study, 80% of the data from six subjects is used for training and 20% used for testing. The evaluation process is conducted on five different Raspberry Pis for each classifier. Our results confirm those in [46]

Table 4
Results of the comparison for fall-direction classification (Acc: accuracy, Pr: precision, Se: sensitivity, Sp: specificity, F-m: F-measure).

Classifier	Acc (%)	Pr (%)	Se (%)	Sp (%)	F-m (%)
BDM	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
LSM	97.88 ± 1.85	98.04 ± 1.66	97.92 ± 1.83	97.88 ± 1.84	97.98 ± 1.74
k-NN	99.55 ± 1.17	99.57 ± 1.10	99.53 ± 1.20	99.54 ± 1.17	99.55 ± 1.15
ANN	98.68 ± 2.10	98.77 ± 1.91	98.65 ± 2.17	98.67 ± 2.11	98.71 ± 2.04
SVM	99.43 ± 1.20	99.47 ± 1.12	99.41 ± 1.23	99.43 ± 1.20	99.44 ± 1.18
DTC	95.61 ± 3.67	96.10 ± 3.20	95.61 ± 3.72	95.61 ± 3.68	95.86 ± 3.45
RF	99.43 ± 0.89	99.47 ± 0.83	99.43 ± 0.92	99.43 ± 0.90	99.45 ± 0.87
AB	97.73 ± 2.52	97.94 ± 2.37	97.67 ± 2.61	97.72 ± 2.53	97.80 ± 2.48

Table 5
Run-time results for fall-direction classification.

Classifier	Training (μs)	Testing (μs)	Total (μs)
BDM	3.05 ± 2.58	1004.39 ± 66.21	1007.44 ± 65.79
LSM	**0.51 ± 0.19**	**36.53 ± 1.88**	**37.04 ± 1.98**
k-NN	–	327.93 ± 374.67	327.93 ± 374.67
ANN	2605.01 ± 84.74	115.50 ± 14.51	2720.51 ± 92.19
SVM	210.43 ± 27.69	135.92 ± 261.27	346.35 ± 288.18
DTC	362.95 ± 316.11	587.93 ± 1127.23	950.89 ± 1438.55
RF	10170.05 ± 300.71	230.55 ± 378.49	10400.59 ± 671.46
AB	41504.24 ± 4736.56	24810.02 ± 42173	66314.32 ± 5063.93

showing that the training time of k-NN is significantly shorter than that of SVM; however, it suffers from a considerably larger testing time.

4. Robustness analysis of fall-direction classification system

In the previous section, the performances of eight fall-direction classification algorithms are evaluated in classifying detected falls into four basic directions. Such an evaluation implicitly makes the assumption that all fall types are associated with one of these four well-defined directions. However, falls may not always have such well-defined directions in the way they occur, for instance, syncope (fainting) and falling out of bed. A fall-direction classification system that only classifies a fall instance into one of these four directions would erroneously classify such falls into one of the four directions regardless. An effective fall-direction classification system should be able to assign the falls that do not belong to any of the four basic directions to the unknown (zero or null) class. However, the standard implementations of the eight ML classifiers that are considered in this study do not provide such functionality.

Thus, we next address the problem of fall-direction classification in the presence of test data from an unknown class. To our knowledge, such an investigation has not been considered previously in the fall-direction classification realm. The standard implementations of four of the eight ML classifiers are modified to handle feature vectors from an unknown class and included in a robustness analysis where test instances from falls whose directions do not belong to any one of the four basic directions are also used.

4.1. Descriptions of the modified ML classifiers

Modifications to the standard implementations of BDM, LSM, k-NN, and ANN classifiers are briefly described below. Note that BDM was the best performing classifier in Section 3 followed by k-NN in terms of the accuracy criterion. Therefore, two out of the selected four classifiers had top-ranking performance. ANN was the fifth best performing classifier. Including the remaining four classifiers in this part of the study would have required extensive modifications to their training and testing routines and, therefore, not considered within the scope of this study.

BDM Modifications are made both to the training and the test phases of the BDM classifier. After calculating the mean vectors and the covariance matrices of the classes, *a posteriori* probabilities of each training instance for all classes are obtained in a loop. For each class, the minimum *a posteriori* probability among all training instances is stored to obtain a 4×1 threshold vector of minimum *a posteriori* probability values for each class. In the test phase, if the maximum *a posteriori* probability of a test instance to one of the four classes does not exceed the average of the 4×1 threshold vector over the four classes, the test instance is assigned to the unknown class.

LSM Similar to the BDM classifier, both the training and the test phases of the LSM classifier are modified. Upon calculating the mean vectors of the classes in the training phase, the sum of the squared Euclidean distance of every training instance to the mean vector of the class that it belongs to is calculated. A 4×1 threshold vector is constructed where each element of the vector is the maximum of the sum of squared distances for a particular class. In the test phase, after the classification of each test instance, the sum of the squared Euclidean distances from the test instance to the mean vector of the class that it is assigned to is compared to the minimum value in the 4×1 threshold vector. If the obtained sum of squared distance is greater than this threshold value, the test instance is assigned to the unknown class.

***k*-NN** A distance threshold similar to the modified LSM is also employed in the k-NN classifier. After storing the training instances in the training phase, Euclidean distances from every training instance to the others of the same class are calculated. The maximum distance from one training instance to another of the same class is then stored for each class. A 4×1 distance threshold vector consisting of these maximum distances for all classes is stored to terminate the training process. In the standard test procedure of the k-NN classifier, after the label of the test instance is estimated as the most frequent class among its k nearest neighbors, the distance from the test instance to the nearest neighbor is calculated and compared to the average of the four 4×1 distance threshold vectors. If the nearest distance is greater than this distance threshold, the test instance is assigned to the unknown class. A grid search for k from 1 to 50 is conducted for parameter optimization in the modified subject-based cross-validation process that will be explained in Section 4.2, and the optimum k value of 1 is obtained for each fold.

ANN Recall from Section 3.4 that the values obtained at each one of the four output neurons indicate the confidence levels of that data instance belonging to each one of the four classes. To make the ANN classifier capable of recognizing an unknown fall type, a threshold on the confidence level is set. After obtaining the confidence levels of class memberships of each test data through the standard test procedure of ANN, the maximum of these confidence levels is compared to the confidence threshold. If it is less than the confidence threshold, the data instance is assigned to the unknown class. Given that the output values range between 0 and 1, an empirical confidence threshold value of 0.85 is employed. The number of hidden-layer neurons is considered as a parameter of the modified ANN and optimized in the subject-based cross-validation process. This parameter is varied from 1 to 50 and the optimal parameter values are found to be in the interval of $[2, 22]$ for the eight folds.

These four modified ML classifiers are evaluated on the fall dataset that is described in Section 3.1. In that section, only 15 of a total of 20 fall types are used and the remaining five types of falls are excluded from the evaluation of the standard fall-direction classification algorithms. The remaining five fall types that are labeled as "undefined" in Table 2 do not have well-defined directions unlike the other fall types that belong to one of the four basic directions. These five fall types are, therefore, used here as an unknown (zero or null) fall class to evaluate the robustness of the four modified fall-direction classification algorithms. Excluded from any training or validation set in parameter selection, these fall types are to be used only in the test procedure of already trained classifiers with already optimized parameters. Since there is a total of 400 fall instances in this unknown class, the whole set of $1600\ (= 1200 + 400)$ fall instances is used in this part of the study.

4.2. Description of the modified subject-based cross validation

The same subject-based cross validation with parameter optimization that is employed in Section 3.5 is also used here with a minor modification. After dividing the folds and sub-folds of data and separating the training, test, sub-training, and sub-test sets corresponding to every iteration of the three (inner, intermediate, and outer) loops, data from the unknown class are also segmented into eight folds based on the same pair-wise subject combinations used previously. Every fold of data from the unknown class is then added to the test set of the corresponding iteration. At the end, every *training* set in the outer loop consists of a total of $1050\ (= 15$ fall types $\times$ 14 subjects $\times$ 5 trials) data instances whilst every *test* set in the outer loop consists of a total of $200\ (= 20$ fall types $\times$ 2 subjects $\times$ 5 trials) data instances.

4.3. Comparative evaluation of the modified ML classifiers

Here, we compare the classification performance of the modified ML classifiers in the presence of data from an unknown class in terms of confusion matrices and performance metrics described previously. The 5×5 confusion matrices are calculated in the same way as in Section 3.6 with the addition of an unknown direction class to the fifth row and column. Consequently, the definitions and calculations of the five performance metrics described in that section are valid as well, with the addition of the new class.

The overall confusion matrices of the four modified classifiers that are obtained by summing up the eight 5×5 confusion matrices acquired in the outer loop of the cross-validation process are provided in Table 6. This table suggests that there is an obvious trade-off to be made between detecting the falls in the four basic directions correctly and rejecting to make a decision when a data instance from the unknown class is encountered. Modified BDM, k-NN, and ANN classifiers exhibit a strict approach in rejecting to make a decision; that is, they classify a considerable number of falls that actually belong to one of the four basic directions into the unknown class. However, these three classifiers correctly classify the data that belong to the unknown class with high accuracies. Modified LSM algorithm, on the other hand, displays a less strict

Table 6
Confusion matrices in the robustness analysis.

BDM:
$$\begin{array}{cc} & \text{classified} \\ \text{true} & \begin{bmatrix} 592 & 0 & 0 & 0 & 48 \\ 0 & 690 & 0 & 0 & 30 \\ 0 & 0 & 580 & 0 & 60 \\ 0 & 0 & 0 & 624 & 16 \\ 6 & 7 & 0 & 0 & 387 \end{bmatrix} \end{array}$$

k-NN:
$$\begin{array}{cc} & \text{classified} \\ \text{true} & \begin{bmatrix} 572 & 0 & 0 & 0 & 68 \\ 0 & 711 & 0 & 0 & 9 \\ 0 & 0 & 620 & 0 & 20 \\ 0 & 0 & 0 & 632 & 8 \\ 0 & 7 & 0 & 2 & 391 \end{bmatrix} \end{array}$$

LSM:
$$\begin{array}{cc} & \text{classified} \\ \text{true} & \begin{bmatrix} 615 & 0 & 0 & 1 & 24 \\ 0 & 693 & 3 & 15 & 9 \\ 0 & 8 & 616 & 0 & 16 \\ 0 & 0 & 0 & 636 & 4 \\ 11 & 95 & 0 & 28 & 266 \end{bmatrix} \end{array}$$

ANN:
$$\begin{array}{cc} & \text{classified} \\ \text{true} & \begin{bmatrix} 599 & 0 & 0 & 0 & 41 \\ 0 & 708 & 0 & 0 & 12 \\ 0 & 0 & 612 & 0 & 28 \\ 0 & 0 & 0 & 620 & 20 \\ 1 & 2 & 2 & 3 & 392 \end{bmatrix} \end{array}$$

approach in rejecting to make a decision. It shows high accuracy in classifying fall instances that belong to one of the four basic directions; however, it does not show acceptable performance in rejecting to make a decision in the presence of data from an unknown class.

After obtaining the confusion matrices for each of the eight folds, the five performance metrics are calculated in the same way as in Section 3.6. The results of the robustness analysis for the four modified fall-direction classification algorithms are provided in Table 7 in mean plus/minus one standard deviation format.

The performance metrics of the modified fall-direction classification algorithms in the presence of data from an unknown class are all above 90.8%. ANN achieves the highest average values in all performance metrics, followed by k-NN, which is within only 0.4% of ANN in all of the performance metrics. BDM and LSM classifiers follow the performances of these two classifiers with performance values above 94.5% and 90.9%, respectively. Comparing the results in this table with those in Table 4, one can observe a significant decline in the overall performances of the classifiers. This decline is caused by not only the fall instances with a well-defined direction that are assigned to the unknown class but also the fall instances from the unknown class that are classified into one of the four directions. This performance degradation, however, is not dire considering the difficult nature of detecting data from the unknown class with simple modifications to the standard ML classifiers.

It is also evident in Table 7 that the variations (standard deviations) in the results of the performance metrics over different combinations of training and test data are larger than those in the previous case. Overall, satisfactory results are obtained in the presence of test data from an unknown class, where the average values of all performance metrics are above 90.9%.

4.4. Run-time analysis of the modified ML classifiers

The run times of the modified classifiers are also measured and recorded. Because of the additions in the training and test phases of the classifiers, we expect their training, testing, and total times to be considerably larger than those in Table 5.

The same procedure as the one in Section 3.7 is followed to obtain the training, testing, and total times of the four modified fall-direction classification algorithms. The MATLAB version and the specifications of the computer used for running the algorithms are the same. The same subject-based cross validation without parameter optimization as described in Section 3.7 is employed here, albeit with a minor modification: the test data belonging to the unknown class is also segregated into eight folds according to the same two-subject combinations as used previously and each of these folds of additional test data from the unknown class is then added to the test data of the corresponding iteration of the loop. Training, testing, and total run times of the four modified fall-direction classification algorithms are presented in Table 8 in

Table 7
Results of the robustness analysis (Acc: accuracy, Pr: precision, Se: sensitivity, Sp: specificity, F-m: F-measure).

Classifier	Acc (%)	Pr (%)	Se (%)	Sp (%)	F-m (%)
BDM	94.51 ± 4.68	94.70 ± 3.73	94.64 ± 4.48	94.51 ± 4.67	94.67 ± 4.10
LSM	92.96 ± 2.05	92.83 ± 2.62	90.89 ± 2.11	92.87 ± 2.05	91.85 ± 2.32
k-NN	96.25 ± 2.78	95.84 ± 2.55	96.30 ± 2.78	96.25 ± 2.78	96.07 ± 2.66
ANN	**96.41 ± 3.11**	**96.20 ± 3.05**	**96.62 ± 2.95**	**96.42 ± 3.11**	**96.41 ± 3.00**

Table 8
Run-time results for the robustness analysis.

Classifier	Training (ms)	Testing (ms)	Total (ms)
BDM	1.438 ± 0.016	1.359 ± 0.017	2.797 ± 0.033
LSM	**0.054 ± 0.001**	**0.061 ± 0.002**	**0.144 ± 0.003**
k-NN	2.554 ± 0.010	2.223 ± 0.032	4.777 ± 0.039
ANN	4.734 ± 0.183	0.181 ± 0.039	4.915 ± 0.188

mean plus/minus one standard deviation format and for the classification of a single data instance.

It is observed in Table 8 that there are significant changes in the run times and especially in the training times of the fall-direction classification algorithms when compared to the run times in Table 5. The total run time of the ANN classifier in this table is almost twice as much as that in Table 5, whereas the increase is almost three times in BDM, four times in LSM, and more than 14 times in k-NN. These differences are mainly caused by the fact that every training instance needs to be considered in the training phase in order to obtain the thresholds. For instance, the significant increase in the run times of k-NN when compared to the former case is a result of calculating the distance from every training instance to every other training instance of the same class, meaning that it considers all combinations of pairs within all classes in the training set. On the whole, the rankings of the total run times of the modified fall-direction classification algorithms are the inverse of the rankings of their performances in the robustness analysis. That is, the best performing algorithm, ANN, is the slowest in Table 8, which is followed by k-NN, BDM, and LSM, which are the second, third, and fourth best performing algorithms in the robustness analysis, respectively. Even with a total run time of nearly 5 ms for the classification of a single data instance, ANN yields satisfactory results in the robustness analysis and it can be implemented in a real-world fall-detection and classification system.

5. Summary and conclusions

In the first part of this fall-direction classification study, eight different ML classifiers are implemented to classify fall actions (with well-defined directions) into four basic directions. A set of 27 simple features extracted from the motion sensor unit data acquired from the waist of the subject is used for this purpose. After optimizing the parameters of the classifiers through a grid search, the classifiers are evaluated over 1200 directional fall instances using subject-based cross validation.

BDM classifies all test instances correctly, achieving 100% classification rate, followed by k-NN, SVM, and RF with all of their average performance metrics being above 99.4%. A comparison of the run times of the considered classifiers indicates that the majority of the selected classifiers can train and test a single data instance in about 1 ms time. Besides, the four best performing algorithms (BDM, k-NN, SVM, and RF) achieve smaller total average run times than AB.

The results indicate that BDM, k-NN, SVM, and RF can be used for superior fall-direction classification in real-world scenarios where it is necessary to consider the robustness of classifiers to data from an unknown class.

In the second part of this study, BDM, LSM, k-NN, and ANN classifiers are modified to handle the presence of data from an unknown class. Robustness analysis is conducted where 400 test instances belonging to an unknown fall-direction class are included in the test set to evaluate the performance of the modified classifiers. The highest average classification accuracies of 96.4% and 96.3% are achieved by the ANN and k-NN classifiers, respectively. The modified algorithms attain robustness to test data from an unknown class at the expense of considerably larger run times than those in the first part of the study. Besides, the achieved performance levels in the robustness analysis are not comparable with the perfect classification obtained in the classification of falls into four basic directions; however, the results obtained in this robustness analysis bear considerable importance in the evaluation of the fall-direction classification algorithms in a realistic scenario. This robustness analysis is especially valuable since to our knowledge, this is the first study in the fall-classification area to consider the presence of data from an unknown class, modify the classifiers as needed, and analyze their robustness. Such an analysis is not necessary in a fall-detection study where binary decisions are made between fall and non-fall activities without the involvement of unknown classes. Classifying the direction of a fall, however, is a type of multi-class activity recognition process where this kind of investigation happens to be crucial.

Further research can be conducted towards benchmarking various fall-classification methods on assorted datasets. Such a study would considerably increase the real-world applicability of the implemented methods. Although the dataset used in this study is extensive in size, it consists of segmented data collected during laboratory experiments from young and healthy subjects. Evaluating the algorithms based on real-world data acquired from middle-aged and/or elderly subjects would be highly beneficial. Embedding the algorithms in hardware for real-world use would facilitate the comparative evaluation based on such real-world data.

Although the fall-direction classification system developed in this article exhibits satisfactory performance, it is susceptible to the misalignment of the sensor unit. Attaining invariance to the position and orientation of the sensor units has been investigated in [1,2] for daily activity recognition and can be extended to cover fall detection and classification as well. Another promising future direction in fall classification is combining information from the other sensory elements of IoT which are not body-worn but embedded in a smart environment.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. Yurtman, B. Barshan, S. Redif, Position invariance for wearables: interchangeability and single-unit usage via machine learning, IEEE Int. Things J. 8 (10) (15 May 2021) 8328–8342, https://doi.org/10.1109/JIOT.2020.3044754.

[2] B. Barshan, A. Yurtman, Classifying daily and sports activities invariantly to the positioning of wearable motion sensor units, IEEE Int. Things J. 7 (6) (June 2020) 4801–4815, https://doi.org/10.1109/JIOT.2020.2969840.

[3] L. Chen, C.D. Nugent (Eds.), Human Activity Recognition and Behaviour Analysis for Cyber-Physical Systems in Smart Environments, Springer Nature Switzerland AG, Cham, Switzerland, 2019.

[4] A. Patel, J. Shah, Sensor-based activity recognition in the context of ambient assisted living systems: a review, J. Ambient Intell. Smart Environ. 11 (4) (July 2019) 301–322.

[5] World Health Organization, Falls, http://www.who.int/mediacentre/factsheets/fs344/en/. (Accessed 15 June 2021).

[6] L.J. Baraff, R. Della Penna, N. Williams, A. Sanders, Practice guideline for the ED management of falls in community-dwelling elderly persons, Ann. Emerg. Med. 30 (4) (October 1997) 480–492.

[7] N. Pannurat, S. Thiemjarus, E. Nantajeewarawat, Automatic fall monitoring: a review, Sensors 14 (7) (July 2014) 12900–12936.

[8] A. Singh, S. Ur Rehman, S. Yongchareon, P.H. Joo Chong, Sensor technologies for fall detection systems: a review, IEEE Sens. J. 20 (13) (July 2020) 6889–6919.

[9] A. Ramachandran, A. Karuppiah, A survey on recent advances in wearable fall detection systems, BioMed Res. Int. (January 2020) 2167160.

[10] X. Wang, J. Ellul, G. Azzopardi, Elderly fall detection systems, Front. Robot. AI 7 (71) (June 2020), https://doi.org/10.3389/frobt.2020.00071.

[11] M.Ş. Turan, Fall Detection and Classification Using Wearable Motion Sensors, M.Sc. Thesis, Department of Electrical and Electronics Engineering, Bilkent University, Bilkent, Ankara, Turkey, September 2017.

[12] A.T. Özdemir, B. Barshan, Detecting falls with wearable sensors using machine learning techniques, Sensors 14 (6) (June 2014) 10691–10708.

[13] E. Principi, D. Droghini, S. Squartini, P. Olivetti, F. Piazza, Acoustic cues from the floor: a new approach for fall classification, Expert Syst. Appl. 60 (October 2016) 51–61.

[14] S. Taghvaei, K. Kosuge, Image-based fall detection and classification of a user with a walking support system, Front. Mech. Eng. 13 (3) (2018) 427–441.

[15] O. Ojetola, E.I. Gaura, J. Brusey, Fall detection with wearable sensors—SAFE (SmArt Fall dEtection), in: Proc. IEEE 7th Int. Conf. on Intelligent Environments, IE, Nottingham, U.K., 25–28 July, 2011, pp. 318–321.

[16] M. Tolkiehn, L. Atallah, B. Lo, G.-Z. Yang, Direction sensitive fall detection using a triaxial accelerometer and a barometric pressure sensor, in: Proc. 33rd Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society, Boston, MA, U.S.A., 30 August–3 September, 2011, pp. 369–372.

[17] M.V. Albert, K. Kording, M. Herrmann, A. Jayaraman, Fall classification by machine learning using mobile phones, PLoS ONE 7 (5) (May 2012) e36556–e36561.

[18] Y. Choi, A.S. Ralhan, S. Ko, A study on machine learning algorithms for fall detection and movement classification, in: Proc. IEEE Int. Conf. on Information Science and Applications, ICISA, Jeju Island, South Korea, 26–29 April, 2011.

[19] C. Dinh, M. Struck, A new real-time fall detection approach using fuzzy logic and a neural network, in: Proc. IEEE 6th Int. Workshop on Wearable Micro and Nano Technologies for Personalized Health (pHealth), Oslo, Norway, 24–26 June, 2009, pp. 57–60.

[20] Y. Tao, H. Qian, M. Chen, X. Shi, Y. Xu, A real-time intelligent shoe system for fall detection, in: Proc. IEEE Int. Conf. on Robotics and Biomimetics, ROBIO, Phuket, Thailand, 7–11 December, 2011, pp. 2253–2258.

[21] N. Pannurat, S. Thiemjarus, E. Nantajeewarawat, A hybrid temporal reasoning framework for fall monitoring, IEEE Sens. J. 17 (6) (2017) 1749–1759.

[22] S.B. Kwon, J.-H. Park, C. Kwon, H.J. Kong, J.Y. Hwang, H.C. Kim, An energy-efficient algorithm for classification of fall types using a wearable sensor, IEEE Access 7 (2019) 31321–31329, https://doi.org/10.1109/ACCESS.2019.2902718.

[23] S.S. Kambhampati, V. Singh, M.S. Manikandan, B. Ramkumar, Unified framework for triaxial accelerometer-based fall event detection and classification using cumulants and hierarchical decision tree classifier, Healthc. Technol. Lett. 2 (4) (2015) 101–107.

[24] B. Andò, S. Baglio, C.O. Lombardo, V. Marletta, A multisensor data-fusion approach for ADL and fall classification, IEEE Trans. Instrum. Meas. 65 (9) (2016) 1960–1967.

[25] N. El-Bendary, Q. Tan, F.C. Pivot, A. Lam, Fall detection and prevention for the elderly: a review of trends and challenges, Int. J. Smart Sens. Intell. Syst. 6 (3) (June 2013) 1230–1266, https://doi.org/10.21307/ijssis-2017-588.

[26] I. Putra, J. Brusey, E. Gaura, R. Vesilo, An event-triggered machine learning approach for accelerometer-based fall detection, Sensors 18 (1) (December 2017) 20, https://doi.org/10.3390/s18010020.

[27] K.-H. Chen, Y.-W. Hsu, J.-J. Yang, F.-S. Jaw, Evaluating the specifications of built-in accelerometers in smartphones on fall detection performance, Instrum. Sci. Technol. 46 (2) (2018) 194–206, https://doi.org/10.1080/10739149.2017.1363054.

[28] L. Martínez-Villaseñor, H. Ponce, J. Brieva, E. Moya-Albor, J. Núñez Martínez, C. Peñafort Asturiano, UP-fall detection dataset: a multimodal approach, Sensors 19 (9) (April 2019) 1988.

[29] H. Ponce, L. Martínez-Villaseñor, Approaching fall classification using the UP-fall detection dataset: analysis and results from an international competition, in: H. Ponce, L. Martínez-Villaseñor, J. Brieva, E. Moya-Albor (Eds.), Challenges and Trends in Multimodal Fall Detection for Healthcare, Springer Nature Switzerland AG, Cham, Switzerland, 2020, pp. 121–133.

[30] H. Ponce, L. Martínez-Villaseñor, J. Brieva, E. Moya-Albor (Eds.), Challenges and Trends in Multimodal Fall Detection for Healthcare, Springer Nature Switzerland AG, Cham, Switzerland, 2020.

[31] R. Espinosa, H. Ponce, S. Gutiérrez, L. Martínez-Villaseñor, J. Brieva, E. Moya-Albor, Application of convolutional neural networks for fall detection using multiple cameras, in: H. Ponce, L. Martínez-Villaseñor, J. Brieva, E. Moya-Albor (Eds.), Challenges and Trends in Multimodal Fall Detection for Healthcare, Springer Nature Switzerland AG, Cham, Switzerland, 2020, pp. 97–120.

[32] L. Wang, L. Cheng, G. Zhao, Machine Learning for Human Motion Analysis: Theory and Practice, IGI Glob., Hershey, PA, U.S.A., 2010.

[33] B. Barshan, A. Yurtman, Investigating inter-subject and inter-activity variations in activity recognition using wearable motion sensors, Comput. J. 59 (9) (September 2016) 1345–1362.

[34] A.T. Özdemir, B. Barshan, Simulated Falls and Daily Living Activities Data Set, UCI Mach. Learn. Repository, School Inf. Comput. Sci., Univ. California at Irvine, Irvine, CA, U.S.A., June 2018 [Online]. Available, http://archive.ics.uci.edu/ml/datasets/Simulated+Falls+and+Daily+Living+Activities+Data+Set.

[35] A.T. Özdemir, An analysis on sensor locations of the human body for wearable fall detection devices: principles and practice, Sensors 16 (8) (July 2016) 1161.

[36] P. Ntanasis, E. Pippa, A.T. Özdemir, B. Barshan, V. Megalooikonomou, Investigation of sensor placement for accurate fall detection, in: P. Perego, G. Andreoni, G. Rizzo (Eds.), Proc. 6th EAI Int. Conf. on Wireless Mobile Communication and Healthcare, MobiHealth, Milan, Italy, 14–16 November 2016, in: Lecture Notes of the Institute for Computer Sciences, Social Informatics, and Telecommunications Engineering (LNICST), vol. 192, Springer International Publishing AG, Cham, Switzerland, June 2017, pp. 225–232.

[37] E. Pippa, E.I. Zacharaki, A.T. Özdemir, B. Barshan, V. Megalooikonomou, Global vs local classification models for multi-sensor data fusion, in: Proc. 10th Hellenic Conf. on Artificial Intelligence, Patras, Greece, 9–12 July, 2018.

[38] S. Abbate, M. Avvenuti, P. Corsini, J. Light, A. Vecchio, Monitoring of human movements for fall detection and activities recognition in elderly care using wireless sensor network: a survey, Wireless Sensor Networks: Application-Centric Design, InTech, Rijeka, Croatia, 2010.

[39] MTw Awinda User Manual and Technical Documentation, Xsens Technologies B.V., Enschede, The Netherlands, 2021 [Online]. Available, https://www.xsens.com/hubfs/Downloads/Manuals/MTw_Awinda_User_Manual.pdf. (Accessed 15 June 2021).

[40] M. Kangas, I. Vikman, L. Nyberg, R. Korpelainen, J. Lindblom, T. Jämsä, Comparison of real-life accidental falls in older people with experimental falls in middle-aged test subjects, Gait Posture 35 (3) (March 2012) 500–505.

[41] E. Alpaydın, Introduction to Machine Learning, 2nd ed., MIT Press, Cambridge, MA, U.S.A., 2010.

[42] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd ed., John Wiley & Sons, New York, NY, U.S.A., 2000.

[43] A.R. Webb, Statistical Pattern Recognition, 2nd ed., Wiley, West Sussex, U.K., 2002.

[44] S. Haykin, Neural Networks: A Comprehensive Foundation, 3rd ed., Prentice Hall, Upper Saddle River, New Jersey, NJ, U.S.A., 2007.

[45] C.-J. Lin, A comparison of methods for multi-class support vector machines, IEEE Trans. Neural Netw. 13 (2) (March 2002) 415–425.

[46] J. Clemente, F. Li, M. Valero, W. Song, Smart seismic sensing for indoor fall detection, location, and notification, IEEE J. Biomed. Health Inform. 24 (2) (February 2020) 524–532.

Mustafa Şahin Turan received the B.S. degree in Mechatronics Engineering from Sabancı University, Istanbul, Turkey in 2015 and the M.Sc. degree in Electrical and Electronics Engineering from Bilkent University, Ankara, Turkey, in 2017.

Currently, he is a Ph.D. candidate at École Polytechnique Fédérale de Lausanne (EPFL), Institute of Mechanical Engineering, Lausanne, Switzerland. He was a Research and Teaching Assistant with Bilkent University between 2015 and 2017. His current research interests include safe and adaptive networked control of large-scale systems with applications to microgrids.

Billur Barshan received the B.S. degrees in electrical engineering and in physics from Boğaziçi University in Istanbul, Turkey, and the M.Sc., M.Phil., and Ph.D. degrees all in electrical engineering from Yale University, New Haven, CT, U.S.A.

After working as a post-doctoral researcher in the Robotics Research Group, Univ. of Oxford, Oxford, U.K., she joined the Faculty of Bilkent University, Ankara, Turkey, where she is currently a Professor with the Department of Electrical and Electronics Engineering. Her current research interests include wearable sensing, wearable robots and mechanisms, intelligent sensing, motion capture and analysis, detection and classification of falls, machine learning, pattern classification, and multi-sensor data fusion.

Dr. Barshan received the TÜBİTAK Young Scientist Award (1998), METU Mustafa Parlar Foundation Research Award (1999), and two best paper awards. She served on the Management Committee of the COST-IC0903 Action MOVE between 2010 and 2013.