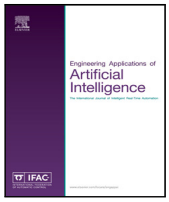
Engineering Applications of Artificial Intelligence 114 (2022) 105170

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



On the feature extraction process in machine learning. An experimental study about guided versus non-guided process in falling detection systems

Elena Escobar-Linero [a], Francisco Luna-Perejón [a], Luis Muñoz-Saavedra [a], José Luis Sevillano [a,b], Manuel Domínguez-Morales [a,b,*]

[a] Architecture and Computer Technology department (ATC), Robotics and Technology of Computers Lab (RTC), E.T.S. Ingeniería Informática, Avda. Reina Mercedes s/n, Universidad de Sevilla, Seville, 41012, Spain

[b] Computer Engineering Research Institute (I3US), E.T.S. Ingeniería Informática, Avda. Reina Mercedes s/n, Universidad de Sevilla, Seville, 41012, Spain

ARTICLE INFO

Keywords:
Deep learning
Recurrent neural networks
Discrete wavelet transform
Fall detection

ABSTRACT

Falls are current events that can lead to severe injuries and even accidental deaths among the population, especially the elderly. Since them usually live alone and their contact with other people has decreased since pandemic, recent years studies have focused on automatic fall detection systems with wearable devices using machine learning algorithms. Overall, and according to other works, these systems can be classified as non-guided, if the machine learning model directly uses raw data without feature extraction, or as guided systems, if a previous step of feature extraction is needed to reduce complexity of the algorithm. However, no recommendations are made in the literature on which system could be more advantageous for detecting fall events. Therefore, in this work, a detailed comparison between both types of systems is carried out, using the same process for different machine learning models in order to obtain an accurate classification of activities of daily living, falling risks, and falls. This process includes the optimization of models' hyperparameters to obtain the best classifiers, followed by an assessment using common evaluation metrics, confusion matrices, ROC curves and execution times. Results show a better classification of models' three classes for the non-guided models. However, the guided models show more stable metrics and lower computational load.

1. Introduction

Fall detection is one of the most common events related to gait study. This kind of event can lead to serious injuries; but the most dangerous consequences are that these injuries sometimes persist, becoming chronic diseases, and, in the worst cases, they can even cause death. Falls are more dangerous in older people and, according to the last study conducted by the World Health Organization (WHO), they are one of the main causes of morbidity and mortality among this group of people (World Health Organization, 2021).

In this study, the W.H.O. estimates that more than 680 thousand mortal falls happen every year worldwide; therefore, after traffic accidents, falls are the main cause of death from unintentional injuries. Furthermore, falls are the cause of more than 38 million permanent life disabilities each year, more than transport injuries, drowning, burns, and poisoning combined.

Among the entire population, older people are at the highest risk of serious injury problems or death caused by falls, and this risk increases year by year. For example, in the United States of America, between 20 and 30% of the elderly who fall suffer more than light injuries (concussions, hip fractures, and head injuries are very common in these cases). Psychological aspects are also very important in falls (such as fear of falling again), not only physical problems. All of these variables cause variations in the gait pattern, increasing the risk of falling.

Regardless of the age group of the population, after a fall occurs, a quick response is essential to prevent the consequences from escalating. However, in exceptional situations such as the COVID-19 pandemic (when mobility and personal contact must be reduced) or inaccessibility to a caregiver or a family member who can attend quickly (due to living in unpopulated areas or alone), automatic falling detection systems (FDS) become essential.

The usual response of FDSs occurs after the user has fallen, acting as a quick intervention mechanism. However, to avoid falls-related injuries, it is interesting to detect not only falls itself, but also the risks of falls to detect gait problems and prevent future falls (although these situations are much more difficult to detect) (Toraman et al., 2010; Jackson et al., 2017; Cuevas-Trisan, 2019). Therefore, the effectiveness of FDS must be analyzed in terms of these three common states (or classes): activity of daily living (ADL), fall, and risk of falling.

* Corresponding author at: Architecture and Computer Technology department (ATC), Robotics and Technology of Computers Lab (RTC), E.T.S. Ingeniería Informática, Avda. Reina Mercedes s/n, Universidad de Sevilla, Seville, 41012, Spain.
E-mail address: mjdominguez@us.es (M. Domínguez-Morales).

https://doi.org/10.1016/j.engappai.2022.105170
Received 24 March 2022; Received in revised form 22 June 2022; Accepted 30 June 2022
Available online 6 July 2022

There are multiple types of FDS that obtain good detection results: user's home integrated solutions (within the scope of the "smart home") by means of cameras (Huang et al., 2018; Lotfi et al., 2018), telecare solutions with user interaction (Do et al., 2018; Sarabia-Jácome et al., 2020), mobile applications or wearable systems (Lee et al., 2018; González-Cañete and Casilari, 2021), among others; but only a few of them include the detection of falling risks.

Although many people tend to spend most of their time at home, it is also important that these systems are able to detect falls outside it. This is why, in recent years, wearable device-based systems have become increasingly important, mainly using the inertial sensors (such as the accelerometer) to detect falls based on sudden variation in acceleration. In this field, this research group has extensive experience handling acceleration sensors in embedded systems, obtaining promising results in previous work (Luna-Perejón et al., 2021b, 2019).

However, analyzing the accelerometer signal is not a trivial task, since multiple activities of daily living can be mistaken for possible falls if rapid movement is required to perform them (such as sitting in a chair, running, or walking down a staircase). Therefore, in recent years, machine learning (ML) techniques have been applied to implement a fall detection system with acceptable results. However, within this field, there are multiple possibilities to extract the characteristics that correctly define a fall and differentiate it from other activities(such as a falling risk or ADL).

In general, there are two clearly differentiated approaches to machine learning-based systems according on how the data is previously treated and then processed by the AI algorithm: On the one hand, there are systems that are trained with the raw data and are able to automatically extract the features that best define the provided data (at the expense of increasing the complexity of these systems), which follow a **non-guided** process of feature extraction; and, on the other hand, there are those systems in which the developer performs a previous process of manual feature extraction in order to facilitate the task of the classifier (at the expense of having the responsibility of properly selecting the features), which follow a **guided** process of feature extraction.

Typically, non-guided systems for time-dependent information use recurrent neural networks with a prefixed window size, while guided systems for time-dependent information typically use features obtained from the frequency components of the time window (usually using the Fast Fourier Transform or the Discrete Wavelet Transform).

There is no global consensus on which of these mechanisms is preferable, and it is up to the researcher to use whichever one he or she deems appropriate. Likewise, a comparison of the two types of systems is difficult to carry out because many different casuistics are involved: dataset used, classes classified, hyperparameters of the neural networks, training process, evaluation metrics, etc. Therefore, to make an accurate comparison, the same process must be followed in both systems, and both development teams must be coordinated.

Therefore, in this work, the main novelty is to take well-known machine learning mechanisms and propose a reliable framework which compares the two type of systems mentioned earlier and suggest a consensus when choosing between one system or another. For this to be achieved, we intend to carry out a double parallel process of design, implementation, and evaluation of two independent classifiers to detect falls, falling risks, and ADLs using the same dataset and following the same classifier elaboration and optimization process. Each of these systems will follow a different methodology to extract the system features: one of them will follow a non-guided process using recurrent neural networks (RNNs), while the second one will follow a guided process using the DWT (Discrete Wavelet Transform) for the extraction of frequency components. The purpose of this study is, starting from the same premises and following a coordinated evolution, to compare these two types of systems in the most rigorous way possible: first, in terms of classification accuracy and, additionally, regarding the computational efficiency (execution time).

Once this internal comparison has been made, the final results will be compared with previous work carried out in recent years.

This work is structured as follows: In the next section, a search of previous works related to FDS implementations based on wearable device sensors is carried out; in the third section, the tools and methodology used to develop both machine learning systems for fall detection are presented; then, in the fourth section, the results of the process detailed in the previous section are shown, and both systems are compared with each other and with previous works, and the results obtained are discussed in detail. Finally, the conclusions obtained from this work are presented.

2. Related works

In this section, a search for similar works is performed in order to compare this work at the end of the manuscript. For this purpose, a global search is performed in the most commonly used search engines (IEEExplorer, ScienceDirect, and Google Scholar) using the following keywords: "wearable", "fall detection", and "machine learning". The resulting set of works is filtered by year, restricting this parameter to those published from 2018 to 2021 (last four years, not taken into account year 2022 as it does not have been finished yet); and taking into account only those works published in international journals with at least five citations. Preprints or arXiv/bioRxiv works waiting for acceptance are not selected.

An initial filtering of works that necessarily include a falling risk classification has not been considered appropriate, as there are only a few works that include this third class. In the filtered related works, only two of them include a third class in their classification system. The results after the search process and the filtering of articles that eliminate those that were not focused on a classifier design reflect the total number of 16 works. The final selected works after the search process are briefly presented and summarized below, but their detailed results are included in the comparison table placed in the results section of this work:

- Howcroft et al. (2018): in this work, a study was carried out with a custom dataset of 75 people (28 previously classified as fallers and 47 as non-fallers) placing acceleration sensors on the head, pelvis, and leg. This is a guided process of extraction of 29 features by accelerometer (87 in total) with Artificial Neural Networks (ANN) and Support Vector Machines (SVM). Although they also used an instrumented insole with pressure sensors, the results of this device have not been taken into account compared to this work. It is important to note that this study considered only a single activity repeated multiple times over six months (walking down a straight corridor while pronouncing a series of words).
- Yoo et al. (2018): In this work, a custom dataset is used again, which only includes five men performing seven activities (of which only one is a fall). Using a non-guided process (raw data) in an artificial neural network with five layers and a large number of neurons (input layer of 525, output layer of 2, and hidden layers of 500, 500 and 2000 neurons), the work classifies between fall and activity of daily living (ADL) using an accelerometer placed on the wrist.
- Putra et al. (2018): In this third article, the information contained in two public datasets (Cogent and Sisfall) is used to differentiate between fall and ADL using two accelerometers located in the chest and waist. A guided feature extraction process (27 features in total) is followed from the pre-impact (9), impact (9) and post-impact (9) phases of the falls. The system is used with a k-Nearest-Neighbor classification algorithm (k-NN), Linear Regression (LR), and SVM to train and test the system. It is important to note the large imbalance between the fall and ADL samples, which results in not providing information on the accuracy of the system but having to resort to other metrics.

- Khojasteh et al. (2018): Using various public datasets, a guided temporal feature extraction mechanism (8 features) from an accelerometer located on the wrist is used to classify between fall and non-fall events. Several mechanisms are used in this work, including ANN, Decision Tree (DT), Rule-Based System (RBS) and SVM.
- Chen et al. (2019): In this work, an assembly of 4 ANNs is used to distinguish between a fall event and ADL, following a guided feature extraction process with 24 temporal and 6 frequential features (30 in total) from the signal of an accelerometer located on the wrist. The dataset used is collected for this work with 11 participants performing 18 activities (only three of them are falls), causing a large imbalance with only 13 falls in total.
- Rivolta et al. (2019): In this work, a study were carried out on 79 elderly people with an accelerometer placed on the chest. Due to the danger of falling in this population, this type of activity is not carried out but 8 ADLs are performed. Therefore, this system performs a classification on a single class (ADL), and any event that escapes the trained parameters is considered an anomalous event (which could include falls). They used 21 features extracted from the accelerometer to train two types of system: one based on a Linear Regression Model (LRM) and the other on artificial neural networks (ANN).
- Hassan et al. (2019): Using the combination of a Recurrent Neural Network (RNN) and a Convolutional Neural Network (CNN), called ConvLSTM (because of the use of LSTM recurrent units), in this work the training and classification of a fall detection system is carried out with data from the MobiAct dataset (collected from the mobile phone accelerometer of 67 subjects). The input data to the network come from the extraction of 58 temporal features of the accelerometer. In this work, two classifications are carried out: a two-class classification (fall and no fall) and another classification of three classes, in which a difference is made between falling, standing up, and lying down.
- Santos et al. (2019): In this work, three public datasets (URFD, SmartWatch, and Notch) are used with data collected from the accelerometer of a smartphone located in the pocket. These data are used without extracting features (raw data) as input to a convolutional neural network (CNN) to distinguish between fall and ADL. It is important to mention that due to the imbalance of the falling samples (<20%) and the small amount of data in these datasets, this work performs a data augmentation process that multiplies the original samples of the URFD dataset by more than 100, and by more than 10 in the case of the remaining datasets.
- Wang et al. (2020): Using data from the SisFall dataset (which contains information from the accelerometer located at the waist), in this work, a CNN is evaluated to distinguish between falls and ADLs by previously extracting 13 features per axis (39 in total).
- Yu et al. (2020): In this work, a classifier based on the combination of a RNN and a CNN (ConvLSTM) is used again, but classifications are made not only with the combination, but also with each network independently. Using raw data from the Sisfall dataset, it is classified into three classes: fall, non-fall, and pre-impact. Although this last class is similar to fall risk events, the SisFall dataset does not include activities with such events, but in this work they are generated manually as a result of the accelerometer variation prior to the fall event.
- Jansi et al. (2020): Using the URFD dataset (with data from six men), a simple system is carried out to distinguish between falls and ADLs based on a threshold detection algorithm. As in other works that use this dataset, the number of samples is very low, including only 40 activities of daily living and 30 falls among all participants.
- Althobaiti et al. (2020): Here, a custom dataset collected from signals from an accelerometer located in the chest of 35 participants is used. Through a guided extraction mechanism of 72 features (between temporal and frequency), four activity classifiers are developed that distinguish between drop and ADL: These classifiers are based on Linear Discriminant Analysis (LDA), Decision Tree (DT), SVM, and ANN. It should be noted that the collected dataset is formed by seven activities, of which only one consists of a fall (it includes only 525 samples of falls among all participants).
- Meyer et al. (2020): On this occasion, a proprietary dataset made up of 37 patients with multiple sclerosis (18 labeled as fallers and 17 as non-fallers) is created, in which information is collected for six months from two accelerometers located on the chest and thigh. With this information, a classifier system is developed that distinguishes between falls and ADL using raw information from accelerometers and a bidirectional recurrent neural network (BiLSTM).
- Alarifi et al. (2021): In this work, a custom dataset is used, in which 14 participants wear accelerometers in various parts of the body (chest, waist, head, both wrists, and both ankles) and perform 36 activities (of which 20 are falls). To classify the activity between falling and ADL, a guided feature extraction process is performed until a total of more than 1,400 features are reached, which are part of the input of a classifier system based on convolutional neural networks (CNN).
- Galvão et al. (2021): In this work, a classification system between fall and ADL is used, using information from two public datasets (URFD and UP-Fall). To do this, they carried out an extraction process of 165 characteristics that are used to distinguish between fall and ADL using two mechanisms: CNN and RNN. Although this work includes, in addition to the above, a vision system to improve fall detection, only results without the vision sensor are used to make the comparison.
- Waheed et al. (2021): In this last work, information from the SisFall and UP-Fall datasets is again used to design an activity classification system that distinguishes between fall and ADL. On this occasion, RAW information from the accelerometer and a bidirectional recurrent neural network (BiLSTM) are used.

It can be observed that there are works that use raw information from the accelerometers (named as unguided process), while others extract characteristics from the accelerometer data (named as guided process). Although this is not related to the results, it is commonly related to the classifier used: normally, systems that use recurrent neural networks do not include a feature extraction process, since the recursion of the network is capable of automatically extracting features that it deems appropriate.

For this reason and to thoroughly compare a guided system with an unguided one, in this work, a system based on recurrent neural networks (unguided without prior feature extraction) is used with another based on classical artificial neural networks (guided, with prior extraction of temporal and frequency characteristics).

The following section will detail the process performed to obtain the classifiers presented in this work. The inner comparison between both classifiers and the comparison with the previous works' quantitative results are detailed in depth in the final part of the Results section.

3. Materials and methods

This section presents the components and methodologies involved in the development of this work. We will start by detailing the data used to train and test the classifiers; then, we will detail the elaboration and optimization process followed for both classifiers; and finally, the process followed to evaluate them.

Table 1
AnkFall dataset characteristics obtained for each participant.

Participants	Activities		
	ADL	Risk	Fall
21	4	3	4

Table 2
Final distribution for each subset using Hold-Out methodology.

Subsets	Participants	Activities	Samples
Train	1–4, 6–7, 11–20	463	6504
Test	5, 8, 9, 10, 21	152	2219

3.1. Dataset

The dataset used in this comparative work contains data from the 3-axis accelerometer obtained from the ADXL345 sensor from 21 participants who performed 11 different activities (4 ADL activities, 3 falling risk activities, and 4 fall activities). It was collected in the Robotics and Technology of Computers lab. during 2020 and published in early 2021 under the name "AnkFALL", because the data recorder device was located in the ankle. This dataset is publicly available online (https://github.com/mjdominguez/AnkFall) (Luna-Perejón et al., 2021a). The content of this dataset is detailed in Table 1.

To evaluate the classifiers with the information provided by this dataset, the Hold-Out technique is applied, splitting the dataset into two subsets: the first is used for training and the second for testing purposes. After this distribution, in order to avoid bias, each subset contains data from different users: among all the participants, the data from 16 of them are randomly selected for training, leaving the data from the remaining 5 users for testing. A 64-sample temporal window is used for segmentation purposes, corresponding to approximately 1.28 s acquired at a frequency of 50 Hz, and a 25% sample size shift (corresponding to 16 samples), and this approach is applied as a data augmentation technique: So, after this process, there is a 75% overlap of information between two consecutive time windows. The distribution of the data is shown in Table 2.

3.2. Classification

As previously mentioned, the main focus of this work is the development and comparison of two classifiers based on different feature extraction techniques for fall detection using an accelerometer signal: on the one hand, a classifier is developed that follows a non-guided feature extraction process and whose implementation consists of a recurrent neural network (RNN), capable of automatically extracting temporal information from the input data; on the other hand, a classifier is developed that follows a guided feature extraction process and whose implementation consists of a classical artificial neural network (ANN) whose input features are previously obtained from the frequency components of the temporal data window (extracted using the discrete wavelet transform or DWT), and also from statistical values of the temporal information.

The choice of these two deep learning models is because the analysis of time series information needs a system that can treat a high number of samples. In this case, RNN models, which have a more complex architecture, are able to analyze these amounts of data by looking for relationships between values of the information and comparing it with the previous time window temporally stored. On the other hand, less complex models, as classical ANN, are a great option if the input data is previously simplified, by analyzing the raw information and extracting informative frequency and temporal features. Additionally, both families of algorithms have been extensively studied and their implementation has been highly optimized in recent years in various frameworks, which allow their optimal integration in systems suitable for different contexts and for different purposes, including embedded systems of low consumption that allows portability and increased autonomy. This fact provides a preliminary advantage over other algorithms framed within classical machine learning. The final goal of this work can be observed in Fig. 1.

Table 3
Hyperparameter values.

Batch size	[16, 32, 48]
Dropout rate	[0.1, 0.2, 0.3]
Neurons and layers	[8, 16, 32, 8:4, 16:8, 32:16]

As can be seen, one network (RNN) has a much higher complexity than the other; although in the classical ANN network the process of converting to frequency components and extracting features from their coefficients is computationally expensive. It is important to clarify that this comparison will be carried out on a level playing field:

- The same dataset will be used.
- The same split between the training set and test set shall be used.
- The same temporal window width shall be used.
- The same overlap of consecutive time windows shall be used.
- The same hyperparameters' values shall be evaluated.
- The same number of layers shall be assessed in both architectures.
- The same optimizer shall be used for the training and using the same optimizer metric.

Finally, the process followed to obtain the best candidate for each type of network (which will ultimately be used in the comparison) is as follows for both networks:

- Phase 1 - Hyperparameters adjustment: multiple trainings are performed by varying the batch size and dropout values, as well as various alternatives of the architecture based on the number of layers and the number of neurons. In more detail, batch size is the parameter that indicates the number of samples used in each iteration of the training process before updating any parameter. As more samples are used in each iteration, the parameters will suffer less updates. However, very low samples used can lead to overfitting the model. Another parameter adjusted is the dropout value, which indicates the rate of zeroing randomly selected neurons in each layer of the network. This is done to prevent the model to overfit. Other hyperparameters, like learning rate, which sets the rate of updates of the weights in each neuron, will be set to a fixed value of 0.0005. Finally, the same alternatives of architecture for both types of network are applied. Taking into account the size of the original dataset, combinations of one and two hidden layers are evaluated, with number of nodes from 8 to 32. Table 3 shows the summary of the hyperparameter values used in this grid search process. 57 combinations will be evaluated for each model to obtain an optimal hyperparameter adjustment.
- Phase 2 - Best candidates selection: the previously obtained results are discussed and the best cases are selected, which will be used in the following phases.
- Phase 3 - Exhaustive candidate assessment: more exhaustive tests are performed over the previously selected candidates using different techniques to assess the robustness by using cross-validation, and the final results are detailed and discussed using different evaluation metrics. Moreover, the execution times are compared too.

3.2.1. Non-guided feature extraction

We refer to non-guided extraction methods as the use of Machine Learning algorithms that fully automate the process of extracting useful features for classification. Thus, these models receive raw data as input,

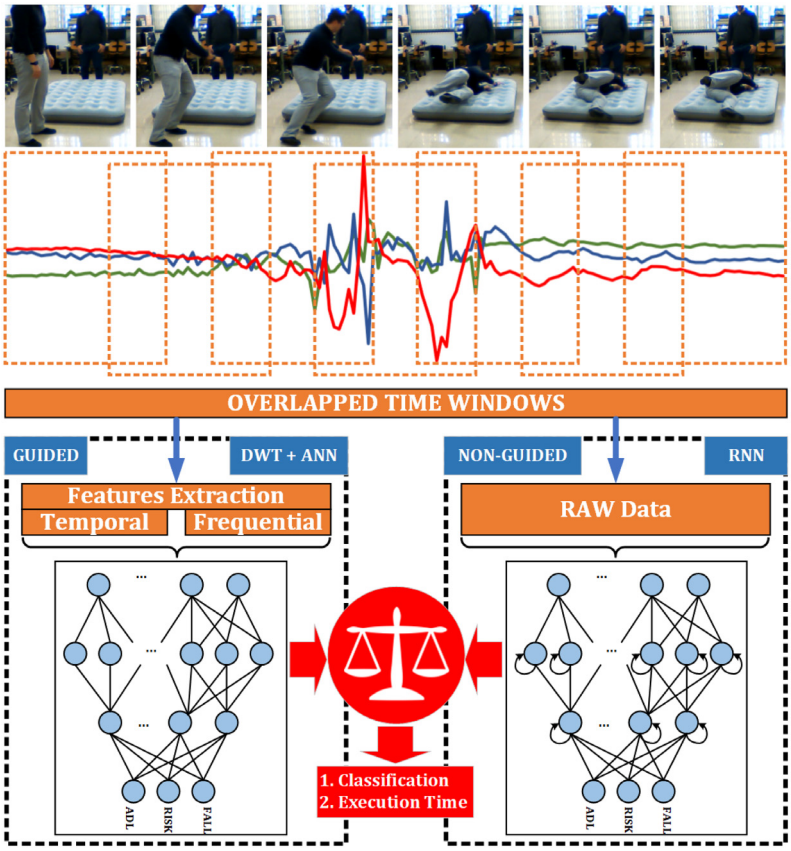


Fig. 1. Work's Graphical Abstract. Fall images obtained from AnkFall dataset (Luna-Perejón et al., 2021a).

that is, as it could be registered by an acquisition device (i.e., the three acceleration axes taken by an accelerometer). This kind of algorithms are included under the term of Deep Learning, which have a large number of adjustable parameters during the training process, and this allows the identification of complex characteristics that are not easy to identify using traditional methods.

These types of algorithms have gained special popularity in the field of artificial vision (Daneshjou et al., 2021; Fu et al., 2021), as well as in the recognition and interpretation of natural language (Chatterjee et al., 2019; Stylianou and Vlahavas, 2021), but there are also numerous studies that use them to extract information from other types of data such as signals (Casal et al., 2021; Buongiorno et al., 2021). Its application in complex signals whose patterns are varied and difficult to cover by extracting specific features is especially interesting, since they can locate and take advantage of undocumented characteristics that improve accuracy by facing a regression or classification problem (Alkhodari and Fraiwan, 2021). Additionally, trained with a sufficiently large and representative set of samples, these algorithms prove to be tolerant to noise, which makes it possible to dispense with filtering phases that consume computing time and energy, and that can eliminate relevant features in the process (Azar et al., 2021).

One of the most studied families of Deep Learning models for the analysis of non-periodic signals are the RNNs, whose design was mainly conceived to extract sequential and temporal characteristics. Specifically, gated Recurrent Neural Networks (gated-RNN) are the most commonly used, as they have a design that favors retaining information extracted from medium and long length sequences. Studies have shown that these algorithms can reach high levels of precision distinguishing between activities of daily living and falls (Farsi, 2021), to the point of also facing the identification of events in which there is a high risk of culminating in a fall (Luna-Perejón et al., 2019). Their main disadvantage is their high computational complexity, although some studies have managed to reduce the complexity of the models to the point that they can be executed in real time on low-power microcontrollers.

RNNs have layers of neurons that receive as input information about a sequence at an instant t and previous information obtained by analyzing the sequence at previous instants. To implement these networks, each recurrent layer is replicated as many times as the length of the input sequences. Classical RNNs, due to their length, suffer from the so-called vanishing gradient problem (Hochreiter, 1998), which does not allow the training of the first layers during the application of the backpropagation through time algorithm (Williams and Zipser, 1995). Gated-RNNs face this problem by including memory cells in their architecture. The memory cells store information separated, saving it over the sequence analysis procedure. The updating and use of the information that is stored during model inference is managed by activation functions. In the context of recurrent neural networks, these activation functions are called gates, due to the function they perform. Their parameters are adjustable during the training process, so that they learn what information is important to retain to extract appropriate characteristics from the analyzed sequence.

The main exponents of this family of RNNs are Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). LSTM units (Hochreiter and Schmidhuber, 1997) includes three gateways in its architecture to manage the stored information. To update the stored information, two of the three activation functions are used, which are called input and forget gates. When the input gate is activated, it introduces new information to the memory of the cell, while the forget gate deletes information that is no longer considered relevant for the analysis of the rest of the sequence. The third activation function, called the output gate, manages the information that is relevant to use for the inference process together with the next element of the analyzed sequence.

The set of components that a classic LSTM unit (or LSTM layer in an specific time t) has are modeled by the following mathematical

expressions:

$$\mathbf{h}^t = \mathbf{o}^t \circ \tanh(\mathbf{c}^t) \tag{1}$$

$$\mathbf{c}^t = \mathbf{f}^t \circ \mathbf{c}^{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}^t \tag{2}$$

$$\tilde{\mathbf{c}}^t = \tanh(\mathbf{W}_{xc}\mathbf{x}^t + \mathbf{W}_{hc}\mathbf{h}^{t-1} + \mathbf{b}_c) \tag{3}$$

$$\mathbf{f}^t = \sigma(\mathbf{W}_{xf}\mathbf{x}^t + \mathbf{W}_{hf}\mathbf{h}^{t-1} + \mathbf{b}_f) \tag{4}$$

$$\mathbf{i}^t = \sigma(\mathbf{W}_{xi}\mathbf{x}^t + \mathbf{W}_{hi}\mathbf{h}^{t-1} + \mathbf{b}_i) \tag{5}$$

$$\mathbf{o}^t = \sigma(\mathbf{W}_{xo}\mathbf{x}^t + \mathbf{W}_{ho}\mathbf{h}^{t-1} + \mathbf{b}_o) \tag{6}$$

In equations above, $\mathbf{h}^t$ (Eq. (1)) indicates the final state values that returns the LSTM unit (or layer) in the specific time t. The cell memory, that is, the information stored, is represented by $\mathbf{c}^t$ (Eq. (2)). The new information coming from the RNN (that is denoted by $\tilde{\mathbf{c}}^t$ in Eq. (3)) is merged by using a hyperbolic tangent function.

Regarding the LSTM gates, both $\mathbf{f}^t$ (Eq. (4)) and $\mathbf{i}^t$ (Eq. (5)) represent the forget and input gates, respectively. Finally, the output gate ($\mathbf{o}^t$ in Eq. (6)) calculates what stored information should be used as the output or state of the cell. These gates use the new input ($\mathbf{x}^t$) and the previous cell state to update the new cell memory information that should be stored.

The $\circ$ symbol denotes vectorial pointwise multiplication, while σ and tanh represent the classic sigmoid and hyperbolic tangent activation functions, respectively. The described equations use different parameters whose values are adjusted during the learning process. These are called weights, and are as follows:

- Bias weights: $b_o, b_c, b_f, b_i \in R^N$
- Input weights: $W_{xo}, W_{xf}, W_{xc}, W_{xi} \in R^{N\times M}$
- Recurrent weights: $W_{ho}, W_{hc}, W_{hf}, W_{hi} \in R^{N\times N}$

where N and M represents the number of cell units and inputs, respectively.

The second Gated-RNN considered, called GRU (Cho et al., 2014), contain two gates, named update and reset. This kind of gated-RNNs differ from LSTMs primarily in that they lack an output gate, and therefore what the cell stores in memory is completely dumped into the neural network during the entire training process. The update cell stores new input information and the rset gate erases data stored from previous iterations.

The equations that rule this gated-RNN are:

$$\mathbf{h}^t = (1 - \mathbf{z}^t) \circ \mathbf{h}^{t-1} + \mathbf{z}^t \circ \tilde{\mathbf{h}}^t \tag{7}$$

$$\mathbf{z}^t = \sigma(\mathbf{W}_{xz}x^t + \mathbf{W}_{hz}\mathbf{h}^{t-1} + \mathbf{b}_z) \tag{8}$$

$$\tilde{\mathbf{h}}^t = tanh(\mathbf{W}_{xc}\mathbf{x}^t + \mathbf{W}_{hc}(\mathbf{r}^t \circ \mathbf{h}^{t-1})) \tag{9}$$

$$\mathbf{r}^t = \sigma(\mathbf{W}_{xr}\mathbf{x}^t + \mathbf{W}_{hr}\mathbf{h}^{t-1} + \mathbf{b}_r) \tag{10}$$

where $\mathbf{z}^t$, $\mathbf{r}^t$ represents the update and reset gates results, respectively. It can be seen that these cells have fewer weight parameters.

In this work, four different main RNN architectures are proposed, differentiated by the type of RNN layers used and the number of layers. Thus, the established architectures contain either one LSTM layer, one GRU layer, two LSTM layers, or two GRU layers (see Fig. 2). For each combination considered, a varied number of neurons were assessed (see Table 3). For those architectures with a single recurrent layer, variations with 8, 16, and 32 neurons in those layers were tested. For those architectures with two recurrent layers, the number of neurons that were analyzed in the first layer were also 8, 16 and 32, while the number of neurons in the second layer was half that of the previous

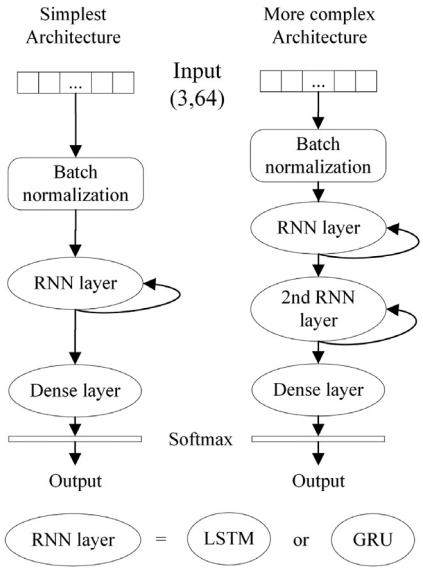


Fig. 2. Summary diagram with the RNN architectures considered in this study.

layer. To combine two consecutive recursive layers, the partial results of the first layer, that is, the resulting states of each recursive unit at each instant t ($\mathbf{h}^t_{first\_layer}$), must be used as inputs of each recursive unit of the second layer $\mathbf{x}^t_{second\_layer}$, for the corresponding instant t.

To complete each architecture it is necessary to include additional layers. First, the inputs layers consist of sequences of 64 × 3 samples, corresponding to the 64 temporary samples, each one composed of the values of the three acceleration axes. Second, they all have a dense or fully connected layer of three neurons as output layer for the classification of three classes (that is, ADL, RISK and FALL) combined with a softmax activation function to normalize the outputs.

Finally, we provide each architecture with two complementary operational layers. In first place, an initial batch normalization layer is used, which normalizes the value of the samples with respect to each batch used during training. This technique has been tested in studies and has been found to facilitate the convergence of trained model (Ioffe and Szegedy, 2015). In second place, dropout was applied in each recurring layer, which randomly drops a random percentage of input connections to the layer (Hinton et al., 2012). In this study three dropout values were analyzed (Table 3), as well as three batch size values. Both techniques are used only during the training of each model, the first to improve convergence and the second to combat overtraining. In addition, a loss function weighted inversely proportional to the number of samples of each class is used. With the use of this function during the training of the models, the error in the classification of a sample belonging to minority classes (that is, FALL or RISK) in the dataset has greater relevance than an error in the classification of a sample of the most abundant class (ADL). Previous studies have verified the effectiveness of these techniques in this gated-RNN architectures (Torti et al., 2019; Luna-Perejon et al., 2019).

3.2.2. Guided feature extraction

Guided extraction attempts to simplify the task of the classifier by adding a previous step to the implementation of Machine Learning models. In this way, the key step in the guided process is the feature extraction procedure. Features are the variables that form the input of the computational model, and they can be obtained from the raw data, but it is essential to extract the correct set of features so that a successful classification can be obtained (Jia et al., 2022). Through this process of feature extraction, most of the information of raw data is preserved, while the algorithm have a lighter processing. Moreover, by extracting

features, the effects of redundant or irrelevant information is overcome, obtaining a new feature space that can effectively be used to establish a reliable correlation between input data and the output classes. Besides, by adding this previous step of extraction, the computational load of the Machine Learning model can be reduced significantly (Syed et al., 2021).

As stated above, it is essential to extract adequate features, since they are the main factor conditioning the success of any subsequent Machine Learning endeavour (Guyon and Elisseeff, 2006). In the last years, different methods have been developed to extract the best features from raw data, obtaining time-domain features, frequency-domain features or even time-frequency domain features. Some previous works have shown that frequency features can achieve better performance than those extracted from the time domain (Braccesi et al., 2015; Hertel et al., 2016; Too et al., 2017). To obtain frequency-domain features, the original data has to be decomposed in its frequency information, which is usually obtained using Discrete Fourier Transform (DFT). However, with physiological signals, as in the case of the acceleration signal used in this work, the periodicity is non-stationary, so it is more accurate to obtain the frequency information in time. The Discrete Wavelet Transform (DWT) is often used as a signal decomposition in time-frequency domain, conducting a feature extraction from the raw data in time domain and frequency domain (Varuna Shree and Kumar, 2018; Ji et al., 2019). On the other hand, several studies have also shown that using only temporal features for acceleration signals can achieve good classification results (Yang, 2009; Arif, 2015).

Having these two possibilities for the acceleration data, the possibility of combining both frequency and time-domain features seems to be the right solution according to previous works. In particular, in the scope of gait analysis and activity recognition, where the original data consists of acceleration signals, previous works have studied the combination of both types of features, achieving even more accurate results when classifying between different events (Althobaiti et al., 2020; Zhao et al., 2020). Following this purpose, in this work we extracted features in both domains. On the one hand, frequency-domain features are the fraction of energy of the DWT decomposition coefficients versus total energy. This characteristic is also called energy distribution, and with this feature some studies have obtained high accuracy classification results (Ayrulu-Erdem and Barshan, 2011; Moran et al., 2015), but also by combining it with the normalized variance of each DWT coefficient (Mitchell et al., 2013). On the other hand, time-domain features are common spatial statistical metrics obtained from acceleration data (Althobaiti et al., 2020). In the following point, the signal decomposition with DWT and the feature extraction process will be detailed.

A wavelet is a wave oscillation which has two main properties: scale and location. The scale is related to the wave frequency, while the location defines the temporal position of the wave (Sekine et al., 1998). Thus, the wavelet decomposition obtains the frequency information of the signal as well as its spatial location. As mentioned previously, this is very useful in non-stationary signals, like acceleration.

Going deeper in the explanation, the wavelet decomposition procedure consists of two main steps. On the one hand, it performs a scalar product between the original signal and a discrete wavelet basis, known as the mother wavelet, obtaining the approximation coefficients of the decomposition. On the other hand, it performs the scalar product between the original data and a father wavelet, which results in the detail coefficients (Chakraborty and Nandy, 2020). In practice, this explanation is implemented as a filter bank composed of levels of low-pass filters, to obtain approximation coefficients, and high-pass filters, which obtain detail coefficients. After the first level of decomposition, approximation coefficients are downsampled through the same process obtaining wavelet coefficients of the second level, and this consecutively until the last level of decomposition is reached. Finally, the DWT obtains one vector of approximation coefficients, A_i, and a set of detail coefficient vectors, $D_1, D_2, \ldots, D_i$, where i is the chosen decomposition level (Songra et al., 2011).

In this way, the mother wavelet is going to establish the properties of the DWT, so it is necessary to choose the correct one, as well as the order of the decomposition. In the literature, there are no specific methods to choose the mother function. In this work, we have selected Daubechies wavelet of third-order level decomposition (db3), since it has a finite number of parameters, it enables fast implementation and high compressibility (Bruce et al., 2002).

The signal is now transformed in time-frequency domain and the next step is the reduction of the wavelet coefficients dimension. This is done through the feature extraction process, which are the energy distribution of each coefficient vector. The energy distribution is obtained as the ratio of the coefficient vector to the total energy of all coefficient vectors. Thus, E_T represents the total energy at the ith level; EDR_A stands for the energy distribution of the low frequency coefficients at the ith level, and EDR_{D_j} is the energy ratio of the jth vector of high frequency coefficients, where $j = 1, \ldots, i$. The equations defining the energy ratios are the following (Ayrulu-Erdem and Barshan, 2011):

$$E_T = A_i A_i^T + \sum_{j=1}^{i} D_j D_j^T \tag{11}$$

$$EDR_A = \frac{A_i A_i^T}{E_T} \tag{12}$$

$$EDR_{D_j} = \frac{D_i D_j^T}{E_T} \tag{13}$$

Combining these energy features with the normalized variances of each wavelet coefficient vector can result in more information obtained from the original data (Mitchell et al., 2013). Since the movement of each participant is measured with a tri-axial accelerometer, the third-order DWT is applied to each axis, obtaining one approximation coefficient vector, A_3, and three detail coefficient vectors, D_1, D_2, D_3, for every axis. Thus, the energy distribution and the normalized variance are calculated for each vector, having 24 frequency features ([4 energy distributions + 4 variances] x 3 axis).

As mentioned earlier, the frequency features are also combined with time-domain features. The latter are obtained as statistical metrics calculated for each accelerometer axis, x, y and z. The metrics are the mean (μ), variance (σ^2), standard deviation (σ), root-mean-square (rms), skewness ($skew$) and kurtosis ($kurt$). They are given by the following equations (Althobaiti et al., 2020):

$$\mu(a) = \frac{1}{N}\sum_{i=1}^{N} a_i \tag{14}$$

$$\sigma^2(a) = \frac{1}{N}\sum_{i=1}^{N}(a_i - \mu(a))^2 \tag{15}$$

$$\sigma(a) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(a_i - \mu(a))^2} \tag{16}$$

$$rms(a) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} a_i^2} \tag{17}$$

$$skew(a) = \frac{1}{N\sigma^3}\sum_{i=1}^{N}(a_i - \mu(a))^3 \tag{18}$$

$$kurt(a) = \frac{1}{N\sigma^4}\sum_{i=1}^{N}(a_i - \mu(a))^4 \tag{19}$$

where $a = x, y, z$ and N is the number of samples. Finally, in the guided extraction process, we have extracted 18 temporal features (6 features x 3 axis) and 24 frequency features, having a set of 42 features that will form the input data of the Machine Learning model.

The model implemented after the process of feature extraction is going to classify this input data into three possible classes: activities of daily living (ADL), falling risk events, and fall events. Taking into account the previous steps of data preprocessing and extraction, an accurate prediction can be made with a simpler classifier. In this work,

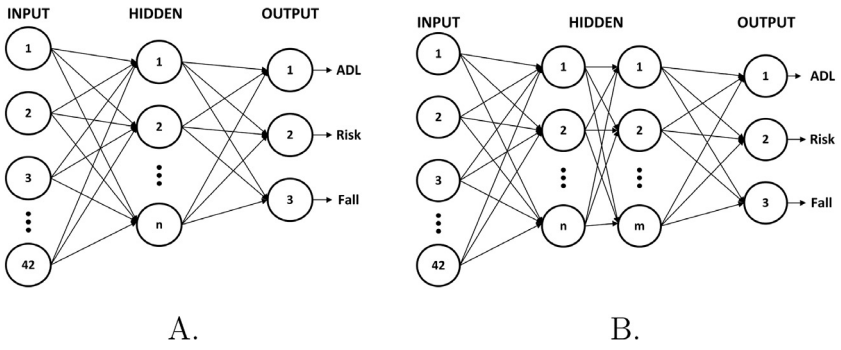


Fig. 3. MLP architecture with (A) one hidden layer and (B) two hidden layers.

we implemented a classical artificial neural network (ANN) based on a multilayer perceptron neural network (MLP).

With this classifier, we can obtain great computational efficiency, since it is simpler than other Machine Learning models, as well as a good classification accuracy. For this purpose, the best MLP architecture has to be implemented, but we need to search for the optimal parameters. This process will be detailed in the next section. Introducing this step, the MLP will have some basic components. Firstly, the architecture will be composed of one input layer, consisting of 42 frequency and temporal features; one output layer, with three neurons that are the possible events to classify; and, finally, between both layers, the hidden layer. In the latter, the number of neurons is variable, looking for the best one, both for one hidden layer and two hidden layers. The basic MLP architecture is shown in Fig. 3-A for one hidden layer and n possible neurons, and in Fig. 3-B for two possible neurons and n and m number of neurons.

The MLP classifier also has fixed hyper-parameters and functions. For the activation functions, a rectified linear function (ReLu) is applied to both the input and hidden layers; while in the output layer a Softmax function is used, which is usually applied for multi-class classification.

3.3. Evaluation

To evaluate the effectiveness in the classification results of a classifier, the most common metrics are used: accuracy (most-used metric), sensitivity (known as recall in other works), specificity, precision, and $F1_{score}$ (Sokolova et al., 2009). To this end, the classification results obtained for each class are tagged as "True Positive" (TP), "True Negative" (TN), "False Positive" (FP) or "False Negative" (FN). According to them, the high-level metrics are presented in the next equations:

$$Accuracy = \sum_{c} \frac{TP_c + TN_c}{TP_c + FP_c + TN_c + FN_c}, c \in classes \tag{20}$$

$$Sensitivity = \sum_{c} \frac{TP_c}{TP_c + FN_c}, c \in classes \tag{21}$$

$$Specificity = \sum_{c} \frac{TN_c}{TN_c + FP_c}, c \in classes \tag{22}$$

$$Precision = \sum_{c} \frac{TP_c}{TP_c + FP_c}, c \in classes \tag{23}$$

$$F1_{score} = 2 * \frac{precision * sensitivity}{precision + sensitivity}. \tag{24}$$

About those metrics:

- Accuracy: all samples classified correctly compared to all samples (see Eq. (20)).
- Sensitivity (or recall): proportion of values classified as "true positive" that are correctly classified (see Eq. (21)).
- Specificity: proportion of values classified as "true negative" that are correctly classified (see Eq. (22)).
- Precision: proportion of values classified as "true positive" in all cases that have been classified as it (see Eq. (23)).
- $F1_{score}$: It considers two of the main metrics (precision and sensitivity), calculating the harmonic mean of both parameters (see Eq. (24)).

The above metrics are common to all ML/DL systems; but there are other commonly used metrics in diagnostic systems; this is the case of the ROC curve (Receiver Operating Characteristic) (Hoo et al., 2017), because it is the visual representation of the True Positives Rate (TPR) versus the False Positives Rate (FPR) as the discrimination threshold is varied. Usually, when using the ROC curve, the area under the curve (AUC) is used as a value of the system's goodness-of-fit.

Therefore, the classifier systems developed in this work will be evaluated according to all the metrics detailed in this subsection. However, for internal benchmarking, in addition to the classification accuracy and the various metrics derived from it, a comparative study of execution times will be carried out. For this, both classifiers will be run on the same machine for multiple iterations; and the average result will be denoted as the system runtime. Once these times have been obtained, a comparison will be made based on the acceleration obtained:

$$Acceleration = \frac{Execution\ Time_{slower}}{Execution\ Time_{faster}} \tag{25}$$

The acceleration result determines the execution speed of the faster classifier with respect to the slower classifier. Subtracting 1 from the acceleration result gives the exact percentage improvement.

4. Results and discussion

This section shows the results of designing and training both classifiers following the three phases previously detailed. Then, the internal comparison between both classifiers is presented, distinguishing between the classification results and the execution times achieved by both systems. Finally, a comparison including the classifiers designed in this work and the classifiers implemented in the works detailed in Section 2 is shown and their results discussed.

4.1. Non-guided system

Starting with the classifiers developed in this work, we first present the system with a non-guided feature extraction process (RNN) in all its optimization phases.

4.1.1. Phase 1

The results obtained in this first phase with the RNN architectures considered (see Tables 4–7) show a slight improvement in accuracy with increasing the dropout value.

However, relating these results in combination with the variation of other hyperparameters, there is no evidence that it is a very influential factor in improving the model. On the other hand, small batch size values seem to have a greater impact on obtaining a more accurate value.

Table 4
Non-guided system — Phase 1. Grid search results for architectures with 1 LSTM layer.

Nodes per layer	Batch size	Dropout											
		0.1				0.2				0.3			
		Train		Test		Train		Test		Train		Test	
		Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss
8	16	83.66	0.7180	86.74	1.2899	83.66	0.7473	88.40	1.2763	78.63	1.0785	89.40	0.9831
	32	81.93	0.8727	88.53	1.0191	81.70	0.8289	88.96	0.8286	81.25	0.9087	90.13	0.8784
	48	85.84	0.6608	86.66	1.2654	81.12	0.9031	87.79	1.0108	83.15	0.8399	88.57	0.8696
16	16	86.23	0.6210	87.92	1.1044	87.39	0.5802	88.74	1.1855	81.40	0.9415	89.14	1.0661
	32	83.48	0.7075	89.92	1.4007	88.60	0.4920	90.13	0.8976	89.05	0.4930	88.83	1.1068
	48	92.94	0.3228	88.35	0.9859	90.20	0.4405	88.27	1.5447	81.03	0.8949	87.92	0.8923
32	16	94.19	0.2539	90.70	1.8703	95.52	0.2092	90.35	1.7410	94.86	0.2169	90.44	1.5402
	32	95.28	0.2089	90.44	1.8797	95.07	0.2017	89.74	1.5543	92.94	0.3150	89.96	1.5440
	48	96.30	0.1483	89.79	2.2084	96.13	0.1734	90.09	1.9165	89.25	0.4597	90.00	0.9842

Table 5
Non-guided system — Phase 1. Grid search results for architectures with 1 GRU layer.

Nodes per layer	Batch size	Dropout											
		0.1				0.2				0.3			
		Train		Test		Train		Test		Train		Test	
		Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss
8	16	80.03	0.8148	88.40	1.1944	80.58	0.8758	89.44	0.9051	81.05	0.8531	88.09	1.3508
	32	79.13	0.9975	87.79	1.0182	82.78	0.7946	87.96	1.0972	82.46	0.8136	88.66	1.4358
	48	85.60	0.6742	86.53	1.0605	81.63	0.8544	89.44	0.9601	82.00	0.8429	89.05	0.9218
16	16	90.04	0.4304	88.61	1.1488	89.78	0.4304	86.57	1.3152	83.94	0.7528	87.05	1.2589
	32	86.57	0.6021	89.00	1.0298	90.98	0.4012	89.31	1.5606	89.66	0.4682	90.57	1.1201
	48	91.86	0.3680	87.87	1.2811	82.78	0.8071	87.96	0.8329	82.30	0.8034	89.14	0.9314
32	16	95.41	0.1995	90.40	1.6722	95.89	0.1847	90.48	2.2336	93.59	0.2776	89.74	1.4615
	32	97.21	0.1238	89.66	2.0697	95.82	0.1930	88.14	1.9251	96.15	0.1803	89.66	1.8874
	48	97.03	0.1293	89.66	2.7663	95.22	0.2086	87.48	1.8410	95.61	0.2305	90.05	1.8588

Table 6
Non-guided system — Phase 1. Grid search results for architectures with 2 LSTM layers.

Nodes per layer	Batch size	Dropout											
		0.1				0.2				0.3			
		Train		Test		Train		Test		Train		Test	
		Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss
8:4	16	86.20	0.6297	86.70	1.1100	82.54	0.8135	86.61	1.2958	78.63	0.9641	89.27	1.1179
	32	88.69	0.5295	86.27	0.8911	87.32	0.6489	88.05	1.4167	78.00	1.0697	88.87	1.0304
	48	81.81	0.8389	88.35	1.0920	86.36	0.5937	87.35	1.9222	80.60	0.9328	89.14	1.2458
16:8	16	92.16	0.3392	90.79	1.6025	87.35	0.5334	90.48	1.3837	91.74	0.3635	90.96	1.5639
	32	90.13	0.3772	88.53	1.1690	89.42	0.4485	88.57	1.5561	89.01	0.5203	90.09	0.9423
	48	92.01	0.3507	87.87	1.7677	87.32	0.5894	85.61	1.4187	92.70	0.3241	86.22	1.9310
32:16	16	96.71	0.1520	90.96	2.4734	96.59	0.1284	90.87	2.3486	96.19	0.1768	91.13	1.8252
	32	97.52	0.0988	90.09	2.4961	96.21	0.1763	91.05	1.7024	96.03	0.1841	91.35	1.6952
	48	96.47	0.1591	90.92	1.8892	96.19	0.1547	90.18	1.8858	98.15	0.0853	89.74	2.3558

Table 7
Non-guided system — Phase 1. Grid search results for architectures with 2 GRU layers.

Nodes per layer	Batch size	Dropout											
		0.1				0.2				0.3			
		Train		Test		Train		Test		Train		Test	
		Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss
8:4	16	84.85	0.6403	88.18	1.7039	83.51	0.8214	88.44	1.0786	78.58	1.0225	88.92	1.1955
	32	86.14	0.6224	87.61	0.8836	81.31	0.7789	87.44	0.9486	82.15	0.8294	86.66	1.3443
	48	84.48	0.6813	87.48	1.2833	83.53	0.7775	88.01	1.6160	80.28	0.9188	90.13	0.8892
16:8	16	89.02	0.4609	89.70	1.5183	90.19	0.4214	89.35	1.6453	86.15	0.6806	89.57	1.0542
	32	86.93	0.5904	89.79	1.1317	90.58	0.4395	90.87	1.2282	89.78	0.4769	89.74	1.2994
	48	83.66	0.7212	88.61	0.8340	94.31	0.2437	90.27	1.4280	92.74	0.3574	88.40	2.1374
32:16	16	96.15	0.1661	90.44	1.6046	95.80	0.1847	91.44	1.8753	95.89	0.1785	90.53	2.1071
	32	97.85	0.1104	90.83	2.1141	96.19	0.1576	90.83	2.3292	96.34	0.1719	89.66	2.7514
	48	98.10	0.0842	90.27	1.7973	97.34	0.1268	89.14	2.1646	96.19	0.1615	90.40	2.1886

The increase in the number of nodes is the factor that most positively affects the accuracy during training. This increase in accuracy is less significant on the test set, although in general it still implies an improvement. However, the value of the loss suffers an increase.

Table 8
Non-guided system — Phase 2. Results obtained for the best candidates.

Model	Nodes	Batch size	Dropout	Train		Test	
				Acc	Loss	Acc	Loss
LSTM 1 layer	32	16	0.1	94.19	0.2539	90.70	1.8702
GRU 1 layer	16	32	0.3	89.66	0.4682	90.57	1.1201
LSTM 2 layers	32	32	0.3	96.03	0.1841	91.35	1.6952
GRU 2 layers	32	16	0.2	95.80	0.1847	91.44	1.8953

It is necessary to remember at this point that the loss function used in these models is weighted to combat the tendency of these architectures to focus on classifying the classes with the largest number of samples. Therefore, the balanced accuracy of the model seems to decrease as the number of nodes increases. Even so, given that the criterion established in this study to select the best candidate model has been focusing on the accuracy of the system, it was decided to investigate this aspect in later stages.

Regarding the models with a single LSTM layer, those with only 8 nodes reach an accuracy greater than 86% on the training set. With 16 and 32 nodes, the accuracy does increase considerably, reaching values above 96% in cases with 32 nodes. The increase in precision over the test set is not so notable, however, in general terms, a greater inference effectiveness is appreciated with the increase in the number of nodes.

The results obtained with models with a single GRU layer are similar to the LSTM models. On the training set, a slightly greater gain in precision is observed when increasing the number of nodes, but not on the test set. Like the single-layer LSTM models, the highest accuracy over the test set is over 90.5%.

Regarding the models consisting of two LSTM layers, those with the lowest number of evaluated nodes have lower accuracy results on the training set compared to the single-layer RNN models. However, more complex models in terms of the number of neurons obtain a significant improvement in accuracy, reaching values above 98%. The accuracy achieved on the test set of the more complex models is also slightly higher, exceeding 91%. As occurs with the set of previous models, although the accuracy increases, there is also an increase in the value of loss, which may lead to a reduction in the weighted accuracy.

Finally, the results obtained with models composed of two GRU layers show precision and loss values very similar to the models that consist of the other type of recurring nodes, the improvement in precision being again remarkable as the number of nodes increases but increasing the loss.

4.1.2. Phase 2

Table 8 shows the best models obtained in terms of precision with respect to the test set, which are used in the subsequent phases as candidates for each non-guided architecture considered. As already mentioned, in general, the number of nodes is the parameter that most influences the improvement of the model's accuracy. Except for the one-layer GRU model, all other candidates have 32 nodes in their first recurrent layer. Note that although this model has less precision than the rest, it is also the one with the lowest loss value among all candidates.

4.1.3. Phase 3

Cross-validation results. The results obtained when analyzing the architectures with cross-validation with the parameters chosen in the second phase are illustrated in Table 9. In general terms, the average precision obtained is less than that resulting from applying Hold-Out. With some folds, the accuracy obtained has improved, but with other combinations this metric has reduced its value considerably in relation to the results of the first phase. As a whole, the results show a high standard deviation, revealing that the effectiveness of these models is strongly influenced by the subset of data used for training. This result is probably a consequence of the use of the loss function used to obtain a better balanced precision on the classes.

Focusing on the results for each fold, we appreciate that on folds 3 and 6, greater precision is achieved on the test set than on the training set. This seems to indicate that the training set has more complex samples to be assimilated by this type of model, but it extracts common key characteristics that manage to cover a greater number of events in the test set. On fold 7, the worst results are obtained, more remarkable in models with a single recurring layer. In these two models, it has been observed that the number of true positives in the FALL and RISK classes increases, but so does the number of false negatives in the ADL class, in which there is a greater number of samples, which has a negative impact on the accuracy of the model.

Focusing on the cross-validation results for each model, we observe that those with two recurrent layers achieve a higher mean precision and a lower standard deviation, which reveals greater independence from the training data set used. The model composed of a single GRU layer presents the lowest precision values on average, which is a consequence of favoring the classification of the RISK and FALL samples, less abundant, to the detriment of the ADL class, with a greater number of samples. The model with one LSTM layer has an average precision value close to that obtained with the models with two recurring layers, but a standard deviation close to 10, which is mainly due to the low precision resulting from fold 7, in which it has been obtained a high number of False Negatives of the ADL class.

Metrics. Table 10 shows the effectiveness metrics of each candidate model. It can be seen that the models with the highest number of neurons and layers obtain the best performance in terms of effectiveness. These results agree with the theoretical foundations, since a greater complexity of the architecture is capable of extracting a greater number of characteristics from the obtained dataset. However, an excessive increase in complexity produces an overfitting of the model to the particular characteristics of the samples that make up the training set. The use of techniques that combat this overtraining favors performance. It can be seen that three of the four best models obtained have the highest dropout value used in the study, which agrees with what was previously stated.

Focusing on metric results, with respect to sensitivity (recall), all models present macro values around 0.79 and 0.81. We can see that this metric has very high values by focusing on the ADL and FALL classes, revealing that the models are effective in identifying these events and produce few false negatives for each class. However, this same metric presents very low values focusing on the RISK class, that is, the ratio of identified risk events is very low, being classified instead as daily life activity events or falls. This low value is responsible for reducing the sensitivity of the model in macro terms. These results are similar to those obtained in previous works (Luna-Perejón et al., 2019) and it is mainly due to two reasons: firstly, some of these events exert little acceleration and it is easily confused with ADL performed with greater intensity during movement, and secondly, the short duration of these events implies that segments corresponding to falls and ADL events often appear in the time windows taken, which makes model training more difficult.

Regarding the specificity measure, macro values above 0.93 are obtained for all classes. The largest error in this metric is related to the ADL class, which has values between 0.84 and 0.88 approximately, which reveals that when these models fail to classify, they tend to classify risky activities and falls as activities of daily living. On the other hand, the values of this metric are very high for the RISK class, indicating that these models do not tend to generate false positives for this class. For the FALL class, this metric also has a high value, indicating that few activities of daily living and risk events are classified as fall events. These results are consistent taking into account that for the training of this type of models a loss function has been used that gives a higher value to the success in classes with fewer samples.

Table 9
Non-guided system — Phase 3. Cross-validation study with the best candidates obtained from the previous phase: [Model1] LSTM 1 layer; [Model2] GRU 1 layer; [Model3] LSTM 2 layers; [Model4] GRU 2 layers.

	fold1		fold2		fold3		fold4		fold5		fold6		fold7		TOTAL			
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	$Mean_{train}$	$Mean_{test}$	STD_{train}	STD_{test}
Model1	94.88	84.31	92.90	88.42	93.56	92.63	93.17	91.00	92.57	90.26	88.08	93.16	83.79	63.07	91.28	86.12	3.63	9.80
Model2	80.62	79.77	80.71	84.19	87.49	92.48	84.51	83.36	89.23	87.55	82.57	90.85	82.29	76.56	83.92	84.97	3.09	5.32
Model3	95.49	83.52	93.63	84.54	91.40	93.46	93.99	91.64	93.73	89.13	93.91	94.18	79.71	88.02	91.69	89.21	5.02	3.86
Model4	96.87	84.39	92.96	85.58	93.66	94.44	95.28	91.16	95.19	89.74	91.81	94.18	80.70	84.63	92.35	89.16	5.00	4.03

Table 10
Non-guided system — Phase 3. Metrics for the best candidates (Hold- Out).

Model	Params			Metrics					
	Nodes	Batch size	Dropout	Accuracy	Class	Specificity	Precision	Sensitivity	f1-score
LSTM 1 layer	32	16	0.1	90.70	ADL	0.8414	0.9263	0.9547	0.9402
					RISK	0.983	0.7806	0.5628	0.6541
					FALL	0.9673	0.8919	0.9167	0.9041
					macro	0.9306	0.8663	0.8114	0.8329
GRU 1 layer	32	16	0.3	90.57	ADL	0.8526	0.9306	0.948	0.9392
					RISK	0.984	0.7852	0.5442	0.6429
					FALL	0.9574	0.8653	0.9306	0.8967
					macro	0.9313	0.8604	0.8076	0.8263
LSTM 2 layers	32:16	32	0.3	91.35	ADL	0.8776	0.9418	0.9493	0.9455
					RISK	0.9870	0.8030	0.4930	0.6110
					FALL	0.9399	0.8201	0.9365	0.8749
					macro	0.9349	0.8552	0.7930	0.8105
GRU 2 layers	32:16	16	0.2	91.44	ADL	0.8484	0.9297	0.9607	0.9449
					RISK	0.9905	0.8593	0.5395	0.6629
					FALL	0.9569	0.8614	0.9127	0.8863
					macro	0.9319	0.8835	0.8043	0.8314

Focusing on the results of the precision metric, the macro values are around 0.85 and 0.88. The class with the highest value in this metric is ADL, which indicates that the number of risk and fall events classified as ADL is very low in relation to the number of correctly classified daily life activity samples. This result is common when the dataset used for training is unbalanced. The larger number of samples of the ADL class reduces the influence of false positives in this class. For the RISK and FALL classes, the values obtained in this metric are more varied, but in general, better results are obtained with fall events than risk events. The best precision value for fall events is obtained by the model of an LSTM layer.

The f1-score metric is similar in all candidates, but the highest value is achieved with the one-layer LSTM model due to the fact that it obtains the most balanced values of the sensitivity and precision metrics.

The confusion matrices (Fig. 4) reveal the behavior of the trained models in more detail. As verified from the sensitivity results, the models have difficulty classifying fall risk events. However, the model with a single LSTM layer achieves a much higher success rate on risk events than other models, thus obtaining the most balanced results as a classifier of the three classes considered in the study. However, this model is also the one that obtained the lowest percentage of success in the other two classes (ADL and FALL) among all non-guided training models.

Confusion matrices also reveal that the highest percentage of unidentified risk events are instead classified as activities of daily living. On the other hand, it can be seen that there is a percentage of between 5% and 8% of fall events mistakenly classified as daily life activity events.

ROC curves (Fig. 5) show AUC values greater than 0.90 for all models and for all classes. The four candidate models exceed the AUC value of 0.97 for the FALL class, revealing a high reliability in the models classifying fall events. For the ADL class, the AUC values exceed 0.94. Regarding the ROC curve of the ADL class, compared to the FALL one, a response is perceived in the sensitivity value that is more conditioned to the increase in specificity, which may be due to the greater variety of different events that exist in daily living activities, as opposed to fall events. Finally, the ROC curves of the RISK class are much less pronounced, which reveals the greater difficulty in identifying these events, discriminating them from events of the other two categories. Even so, the reported AUC values are remarkably high.

Table 11
Non-guided system — Phase 3. Execution times (in s).

Model	CPU Intel i7-10700K @ 3.80GHz		GPU (GeForce GTX 1080Ti)	
	Mean	SD	Mean	SD
LSTM 1 layer	2.34E−02	4.13E−04	3.51E−03	2.48E−04
GRU 1 layer	2.78E−02	3.17E−04	3.31E−03	9.73E−05
LSTM 2 layers	4.67E−02	7.45E−04	5.81E−03	1.34E−04
GRU 2 layer	5.52E−02	7.96E−04	5.28E−03	1.42E−04

Execution times per inference. Models with two recurring layers double the execution times using CPU. This time difference is smaller using GPU. The results obtained show that the models based on RNN take advantage of the computational potential of the GPU to obtain shorter execution times than using the CPU. With the hardware resources used, times between five and nine times shorter are obtained and shown in Table 11.

4.2. Guided system

After presenting results of non-guided models, in this section the system with a guided feature extraction process (MLP) in all its phases is shown.

4.2.1. Phase 1

For the guided system, the results obtained in the hyperparameters adjustment phase are shown in Tables 12–13. In general, the variation in the dropout value leads to significant changes in the accuracy: for a small dropout value, more accurate results are reached, both for the train set and the test set. The combination of this hyperparameter with the number of neurons and the batch size will get higher values of

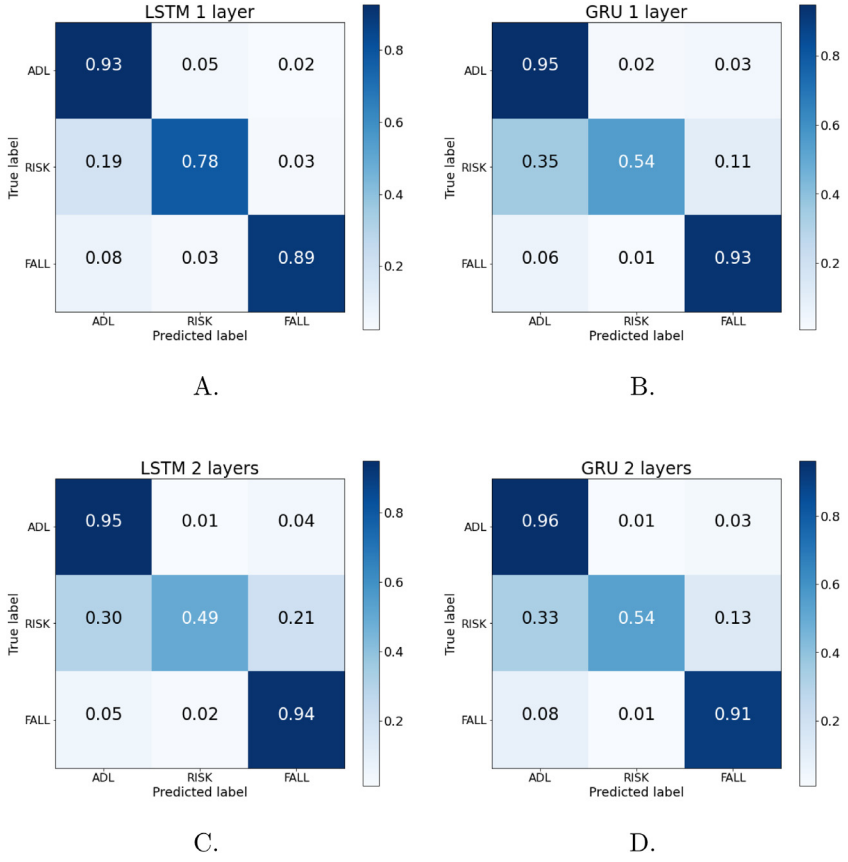


Fig. 4. Confusion matrices for best RNN models obtained from Hold-Out.

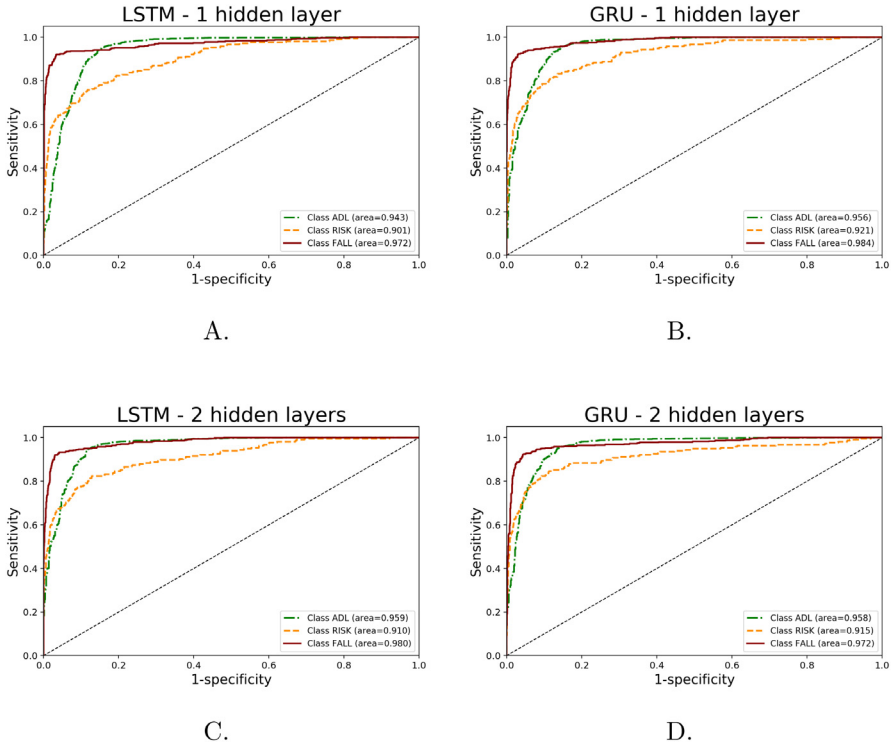


Fig. 5. ROC curves for best RNN models obtained from Hold-Out.

Table 12
Guided system — Phase 1. Grid search results for architectures with 1 hidden layer.

Nodes per layer	Batch size	Dropout											
		0.1				0.2				0.3			
		Train		Test		Train		Test		Train		Test	
		Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss
4	16	89.84	0.2851	86.26	0.537	88.08	0.3364	87.79	0.3655	87.52	0.3602	87.56	0.4174
	32	88.62	0.3257	86.84	0.4119	84.95	0.3874	87.2	0.4899	85.87	0.3833	85.53	0.4911
	48	89.74	0.2812	86.48	0.4695	72.82	0.5896	83.73	0.5582	80.52	0.4478	83.82	0.4852
8	16	90.93	0.2398	86.21	0.498	89.07	0.2969	86.93	0.4857	89.21	0.2935	86.07	0.5054
	32	91.84	0.2231	86.12	0.5328	89.82	0.2764	86.75	0.537	89.73	0.2776	86.48	0.4762
	48	91.34	0.2498	87.11	0.4159	89.18	0.2905	86.53	0.4264	90.16	0.2713	86.84	0.4953
16	16	92.2	0.2111	87.52	0.4428	92.54	0.2051	87.07	0.5765	91.9	0.2298	86.84	0.5017
	32	92.44	0.2085	87.7	0.4834	91.64	0.2244	87.29	0.4567	91.08	0.2388	86.57	0.4838
	48	92.7	0.2074	87.34	0.4685	91.59	0.2324	86.3	0.486	91.45	0.2355	86.71	0.5117
32	16	91.84	0.2269	86.66	0.4504	91.79	0.2285	87.02	0.4449	92.39	0.2052	86.57	0.6301
	32	92.56	0.2075	86.93	0.4386	92.84	0.1897	87.61	0.5279	93.27	0.1879	86.71	0.5743
	48	93.63	0.1818	87.11	0.5358	91.77	0.2262	86.8	0.4184	93.0	0.2001	86.75	0.5376

Table 13
Guided system — Phase 1. Grid search results for architectures with 2 hidden layers.

Nodes per layer	Batch size	Dropout											
		0.1				0.2				0.3			
		Train		Test		Train		Test		Train		Test	
		Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss
4:2	16	84.07	0.4939	82.06	0.5306	82.47	0.5036	84.14	0.6034	79.31	0.5475	85.44	0.5122
	32	83.67	0.4574	84.18	0.4352	72.02	0.603	84.32	0.5594	79.6	0.5695	84.45	0.5161
	48	86.96	0.401	85.26	0.4746	82.73	0.4776	83.64	0.5014	78.55	0.5551	84.23	0.4752
8:4	16	90.97	0.2666	88.01	0.5673	86.49	0.3555	87.79	0.4072	88.99	0.3512	86.53	0.9313
	32	89.87	0.2756	88.1	0.5573	89.82	0.2902	87.2	0.7472	86.09	0.4015	88.64	0.3482
	48	90.24	0.2733	87.29	0.5533	89.19	0.335	86.66	0.5064	86.36	0.3972	86.03	0.7481
16:8	16	92.56	0.2033	88.1	0.5196	90.97	0.2437	87.97	0.6454	90.56	0.2647	87.43	0.6973
	32	93.08	0.1882	87.25	0.6477	91.19	0.2423	87.43	0.553	90.11	0.2812	86.62	0.6724
	48	92.1	0.2276	86.21	0.5664	90.53	0.2607	85.89	0.5202	90.76	0.2578	86.8	0.8694
32:16	16	95.22	0.1348	87.07	1.058	91.17	0.2492	86.16	0.4556	92.91	0.1987	87.47	0.8402
	32	92.11	0.2236	87.2	0.4532	92.19	0.2135	87.25	0.4599	93.19	0.1886	87.07	0.7782
	48	91.62	0.2423	85.76	0.43	91.42	0.241	85.53	0.5052	91.45	0.2357	87.74	0.4742

accuracy. This way, as the number of neurons in the hidden layers increases, the accuracy improves. However, the batch size needs to be smaller or of an intermediate value so that this hyperparameter can have a positive impact on getting higher accuracy.

The influences of these hyperparameter values are much more significant on the accuracy of the train set, while for the test set they achieve much less changes. However, in both sets, a small dropout value, a higher number of nodes, and a smaller batch size will affect the increment of the accuracy.

On the other hand, the value of the loss on the training set tends to decrease as the number of nodes gets higher and the dropout value gets smaller. This is less remarkable on the test set, it even leads to higher loss in some cases.

Regarding the MLP models with one hidden layer, the training accuracy increases significantly as the number of nodes gets bigger, obtaining a difference of more than 3% between the accuracy of the models of lowest number of neurons and those with highest number of nodes. In the case of the test set, this influence is less significant, and in some cases this factor combined with the variation of other hyperparameters affects negatively the accuracy. However, for all single-layer MLP models, the accuracy of the test set is always between 86% and 88%. The accuracy in both sets suffers a decrease when the dropout value gets higher values. With batch sizes of 16 or 32, more accurate results on the train set can be obtained. For the test set, depending on the combination of batch size and nodes, the accuracy can improve slightly. The highest accuracy value for the test set achieved in this case is 87.7%.

Finally, the effects of increasing the number of nodes for MLP models with two hidden layers are much more remarkable than in the previous case. The training accuracy between models with the lowest and the highest number of nodes increases by more than 8% in all cases and reaches an accuracy value of 95%. The same influence occurs in the test set. With 16 and 8 nodes in both hidden layers, the accuracy of the test set is over 88%. In this MLP architecture with two hidden layers, smaller batch sizes and dropout values have a positive impact on the accuracy.

4.2.2. Phase 2

In the second phase of the guided system, the best candidates are selected among all models of the previous phase (Table 14). For both MLP models with one and two hidden layers, the best candidates have the second highest number of nodes, emphasizing the importance of more nodes on the increase of the accuracy. As explained before, the increase of the accuracy mostly occurs for lower values of dropout. In both MLP models, the Dropout value is 0.1. Note that even though they have both similar accuracy and loss values, the two-layer MLP model obtains better results.

4.2.3. Phase 3

Cross-validation results. Once the best parameters are chosen for MLP models, the two architectures are analyzed with cross-validation. Table 15 illustrates the results obtained. Unlike the non-guided system, in this system, the test accuracy is higher than the accuracy obtained in the Hold-Out. Only in one fold the accuracy in the training set is lower than the one obtained in the previous phases.

Table 14
Guided system — Phase 2. Results obtained for the best candidates.

Model	Nodes	Batch size	Dropout	Train		Test	
				Acc	Loss	Acc	Loss
MLP 1 layer	16	32	0.1	92.44	0.2085	87.7	0.4834
MLP 2 layers	16:8	16	0.1	92.56	0.2033	88.1	0.5196

In more detail, for fold 2, the precision obtained in the test set is the lowest. The reason of this result is probably because the RISK class has a higher number of samples and the number of false negatives increases significantly. It has been observed that for ADL class there is also a higher number of false negatives compared to the other folds. These two increases affect negatively the accuracy of the model.

Focusing on the results for both models, in most cases the accuracy increases for the two hidden layers. In very few results, the accuracy is higher for the model with one hidden layer. In some folds, the accuracy on the test set gets bigger values than for the training set. As mentioned earlier, this could be an indicator of more complex samples in the training set.

On the other hand, the results show a very low standard deviation both for the test and train sets, demonstrating that the guided system used is very consistent when it classifies between all three events. For the MLP model with two hidden layers, the average accuracy in both sets is higher than the model with one hidden layer. Nonetheless, the two MLP models reach higher values than those obtained previously for the Hold-Out. Moreover, the standard deviation is very low in both cases, but smaller for the two-layer MLP, which indicates that, although fold 2 has the lowest accuracy results, in general, the guided system has a great uniformity and low dependence on the set used for training.

Metrics. After evaluating the consistency of the MLP model through the process of cross-validation, the metrics for the best candidates obtained in phase 2 are calculated (see Table 16). It can be observed that, as in the case of RNN models, higher number of hidden layers leads to better performance results. However, differences in evaluation metrics between both models are significantly low, with an accuracy of 0.4% higher for two hidden layers' model. This is due to the inclusion of the guided process of extracting temporal and frequency variables, which allows the complexity of MLP models to be reduced while still achieving great effectiveness results. On the other hand, the number of neurons is the medium value of those tested in the optimization process, avoiding an excessive complexity of the network and therefore an overfitted model, but also avoiding an underfitted model since the number of neurons is not excessively low. Moreover, the number of neurons is slightly lower than for RNN models, highlighting the simplification of the MLP models by introducing the previous step of feature extraction which makes it easier for the algorithm to find learning patterns.

It can also be observed that, unlike the results of the optimal hyperparameters obtained in RNN models, in this case the dropout value is very low, both for one hidden layer and for two hidden layers. Thanks to the previous workload of extracting features, the model can easily find patterns between the input data and the output, without the need of removing a higher ratio of random neurons to avoid overfitting.

The optimized batch sizes for both models also show that the overfitting is avoided since the values are not excessively low, with which the model would perform too many times backpropagation and it would be less generalized for new data.

In more detail, it can be observed that the macro sensitivity has lower values than the rest of the metrics. This decrease is produced because of the very low sensitivity in the RISK class, where the value is lower than 0.5 in both cases. This indicates that there is a high number of false negatives in this class or, in other words, that a high number of RISK events are classified as ADL or FALL. Falling risk is the most ambiguous event out of the three classes, being complicated to classify as such, as the beginning of the stumble or the final cushioning after the trip can easily be mistaken for a fall or ADL event. Moreover, because of its short duration, the time window can be shared with falls or ADL, thus making difficult the correct classification, as stated previously in non-guided explanations. On the other hand, for ADL and FALL classes, the value of sensitivity is higher, and for ADL class it achieves values of around 0.94 in both models. These good results in sensitivity indicate a high number of true positives versus a low number of false negatives.

On the other hand, the macro specificity is around 0.91, where the highest value is obtained for RISK class, which indicates that there is a low number of false positives for this class. However, the worst specificity value is for ADL class, showing that there is a high rate of false positives for this class, which means that a high number of FALL and RISK events are misclassified as activities of daily living. According to the sensitivity results, it is likely that most false positives come from the RISK class. This possible case will be discussed and verified more easily with the confusion matrices illustrated below.

Regarding the macro precision metric, 0.83 is obtained for both architectures. In more detail, the lowest value of precision is 0.78 for FALL class in one-layer MLP model, and for RISK class in the architecture of two hidden layers. This indicates that, in the first case, the classifier predicts more falling events when they actually belong to other events; while in the second model, the same case occurs but for risk events. Nevertheless, the precision obtained in ADL class is the highest in both cases and indicates that there is a high number of ADL samples and most of them are classified correctly.

Finally, F1-score shows that the worst results are obtained for the RISK class in both models, due to the poor sensitivity obtained for this class. This lower value influences on the average results, which is slightly above 0.78.

Fig. 6 illustrates the confusion matrices for one-layer MLP model and for the MLP model with two hidden layers, respectively. In both cases, the explanation above is reinforced as it can be observed that RISK class obtains a high rate of false negatives, which directly influences negatively on the sensitivity for this class. The false negatives of RISK class are at the same time false positives for ADL and FALL classes. As in the case of non-guided systems, more than 30% of risk events are classified as activities of daily living. For ADL class, more than a 40% rate are false positives between RISK and FALL classes. This explains the lower specificity value for ADL class.

On the other hand, falling events have a 20% rate of false positives in the model of one layer and 18% for two layers. Although there is a high value of false positives, the specificity of this class remains high due to the high rate of true negatives. The same occurs for RISK fall, where the rate of true negatives is very high compared to the small rate of false positives.

In general, it can be concluded from the confusion matrices that the rate of correctly identified risk events is very low, as verified from the lower sensitivity values in the metrics values explained above.

Regarding the ROC curves of both systems (see Fig. 7), it can be observed how both models have more difficulty classifying risk events. For one hidden layer, the AUC value of RISK class is under 0.90, but for two-layer model it increases to more than 0.91. On the other hand, for the ADL class, the AUC value is over 0.94 for the two candidate models, while for FALL class it exceeds 0.96 in all cases. The same appreciation between sensitivity and specificity of ADL and FALL classes that occurs in non-guided systems is observed in this case.

Execution times per inference. Table 17 illustrates the execution times for the feature extraction process and for the total execution of the model. For both models, the execution times of extracting features are not very different using CPU and GPU. However, the total execution time is shorter on CPU than GPU.

Table 15
Guided system — Phase 3. Cross-validation study with the best candidates obtained from the previous phase: [Model1] 42:16:3, dropout 0.1, batch size 32; [Model2] 42:16:8:3, dropout 0.1, batch size 16.

	fold1		fold2		fold3		fold4		fold5		fold6		fold7		TOTAL			
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	$Mean_{train}$	$Mean_{test}$	STD_{train}	STD_{test}
Model1	90.46	91.28	91.63	84.05	91.85	90.9	91.18	91.48	92.26	89.28	91.7	92.47	89.76	90.66	91.26	90.02	0.87	2.8
Model2	91.72	90.67	92.82	87.03	91.13	91.2	92.25	91.72	91.34	89.21	92.32	92.47	92.16	90.28	91.96	90.37	0.59	1.8

Table 16
Guided system — Phase 3. Metrics for the best candidates (Hold-Out).

Model	Params			Metrics					
	Nodes	Batch size	Dropout	Accuracy	Class	Specificity	Precision	Sensitivity	f1-score
MLP 1 layer	16	32	0.1	87.70	ADL	0.8164	0.9142	0.9374	0.9256
					RISK	0.9885	0.816	0.4744	0.6
					FALL	0.9312	0.7878	0.869	0.8264
					macro	0.912	0.8393	0.7603	0.784
MLP 2 layers	16:8	16	0.1	88.1	ADL	0.8178	0.915	0.9407	0.9277
					RISK	0.9855	0.7868	0.4977	0.6097
					FALL	0.9394	0.8078	0.8671	0.8364
					macro	0.9142	0.8365	0.7685	0.7912

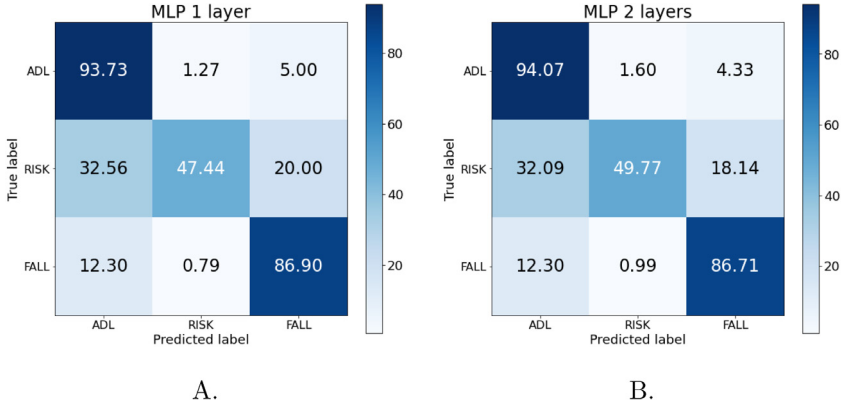


Fig. 6. Confusion matrices for best MLP models obtained from Hold-Out.

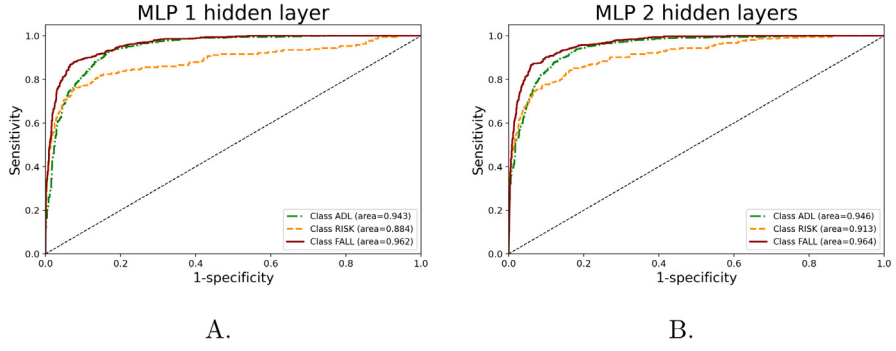


Fig. 7. ROC curve for best MLP models obtained from Hold-Out.

Table 17
Guided system — Phase 3. Execution times.

Model	CPU Intel i7-10700K CPU @ 3.80GHz				GPU (GeForce GTX 1080Ti)			
	Feature extraction		Total		Feature extraction		Total	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
MLP 1 layer	1.93E−03	6.89E−05	2.68E−03	9.07E−05	1.96E−03	3.37E−05	3.11E−03	1.02E−04
MLP 2 layers	1.90E−03	7.69E−05	2.85E−03	9.64E−05	1.98E−03	3.43E−05	3.65E−03	7.87E−05

4.3. Inner comparison

Before going in depth into the comparison between the two systems, it is worth noting that other Machine Learning models have been tested in this study, using the same dataset described earlier, so that a more comprehensive comparative can be done. For this purpose, two well-known models have been implemented: Random Forest and Support Vector Machines. A grid search was carried out for each of these

Table 18
All systems. Summary of results for models of both systems and results of other models obtained after a grid search.

Model	Accuracy	Specificity	Precision	Sensitivity/recall	F1-score
LSTM 1 layer	90.70	0.9306	0.8663	0.8114	0.8329
GRU 1 layer	90.57	0.9313	0.8604	0.8076	0.8263
LSTM 2 layers	91.35	0.9349	0.8552	0.7930	0.8105
GRU 2 layers	91.44	0.9319	0.8835	0.8043	0.8314
MLP 1 layer	87.70	0.912	0.8393	0.7603	0.784
MLP 2 layers	88.1	0.9142	0.8365	0.7685	0.7912
Random Forest	83.24	0.8875	0.8046	0.7153	0.734
Support Vector Machine	87.56	0.9003	0.8498	0.734	0.7689

Table 19
Non-guided system using different MLP models. Performance results obtained. The codes in the "model" column indicate the number of nodes that each layer that makes up the model has, where the first is the input layer.

Model	Train		Test	
	Acc	Loss	Acc	Loss
192:16:3	82.93	0.4810	77.22	1.5574
192:48:3	88.62	0.3881	79.71	5.5508
192:16:8:3	79.25	0.5325	78.30	0.7408
192:48:24:3	86.54	0.3663	82.43	0.8870

Table 20
Non-guided system using different MLP models. Execution times (in s).

Model	CPU Intel i7-10700K @ 3.80GHz		GPU (GeForce GTX 1080Ti)	
	Mean	SD	Mean	SD
192:16:3	3.38E−03	1.91E−04	7.51E−03	4.04E−04
192:48:3	3.37E−03	1.88E−04	7.59E−03	3.68E−04
192:16:8:3	4.79E−03	3.72E−04	1.10E−02	9.86E−04
192:48:24:3	4.73E−03	3.24E−04	1.34E−02	1.36E−03

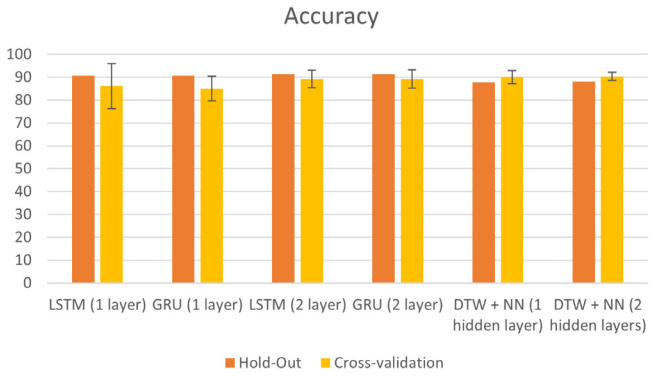


Fig. 8. Accuracy results for both systems with Hold-Out and average accuracy with standard deviation of cross-validation.

models to obtain an optimization of the hyperparameters. Specifically, more than 4000 parameter combinations have been analyzed for Random Forest, while for Support Vector Machines the search was made between 90 possible combinations of the hyperparameters.

The results of the evaluation metrics of the optimized models are shown in Table 18, as well as the summary of the macro results of the guided and non-guided systems presented and explained earlier.

The results of Random Forest and SVM models are lower than for guided and non-guided systems, with an accuracy of 83% for Random Forest and more than 87% for SVM. With Random Forest, results are the worst and this can be the indicator that the samples are complex and bigger models are needed to distinguish between all events successfully. The metric values obtained for SVM are very similar to those obtained for MLP model with one hidden layer. These two models are simpler than those analyzed in depth in this work and obtain lower values, so it may reinforce the fact that to achieve higher performance in fall and risk classification, more complex models such as deep NN or RNN should be used.

Moreover, to present a more reliable comparison, the MLP models have also been trained without following a guided process. Thus, raw information is passed as the input data of the neural network, which substantially increases the number of nodes in the input layer (192). Since the previous step of extracting features is now avoided, the MLP requires more complexity to be able to extract learning patterns, so four combinations of hidden layers have been tested: 192:16:3, 192:16:8:3, 192:48:3, and 192:48:24:3. Results of these tests using the best hyperparameters obtained from the previous classifiers can be seen in Table 19. It can be observed the same number of neurons in MLP models that receive extracted features are not enough, obtaining the worst data for the one-layer networks with 16 neurons, and two layers with 16:8 neurons (not reaching 79% in any case for the test set). Moreover, the loss value obtained is too large, which may denote an overfitting given the percentage of accuracy achieved. With respect to the new architectures with more neurons, a slight improvement can be observed, reaching 82.4% with the most complex architecture. The loss result is not good either, but no overfitting is observed in this case. Although acceptable precision values are obtained, a drop of more than 6% in comparison with MLP guided models occurs.

In addition, by evaluating execution times in the same way as with the initial classifiers, we obtained the results shown in Table 20. As happened in the MLP included in the work, using GPU the time spent is higher due to the transfer overhead. On the other hand, if we compare these results with the ones obtained by the guided MLP-based system designed using CPU computing (see Table 17), it can be observed that the neural network trained with the raw information is much more complex and requires more time to perform the classification, while the network with prior feature extraction requires less time. There is an increase of more than 30% compared with the execution times obtained for MLP guided models (reaching more than 800% in the worst case with the more accurate classifier). Overall, by using an MLP with raw information shows an increase in the complexity and therefore in the execution times. Thus, the comparison of a non-guided RNN-based system with a guided MLP-based system as it is done in this work is a correct approach, since results between them are similar and higher than more simple algorithms like SVM or Random Forest, and better than training a classical MLP with raw data.

Regarding the inner comparison of both systems, Fig. 8 illustrates the summary of the accuracy results for all models. Firstly, it can be observed that the accuracy of the Hold-Out is higher for RNN models than for MLP models. On the other hand, the average accuracy obtained in the cross-validation process shows that the guided system achieves higher values in comparison with the accuracy obtained with the Hold-Out. Moreover, the average accuracy of MLP models is higher than the average accuracy of RNN models. These results reveal a greater dependence of the non-guided models on the train dataset. If the dataset used does not contain enough characteristics and varied data that broadly represents the problem itself (in this study, the identification of falls and risk events), the tendency of this class of models to identify very particular characteristics results in overtraining. This is a common drawback of machine learning models that are delegated automated

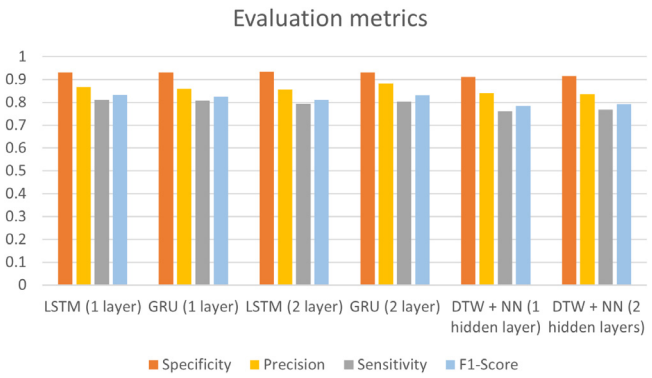


Fig. 9. Summary of evaluation metrics for all best models.

feature extraction (Mazurowski et al., 2019; Rizk et al., 2021). Another factor that influences the training of RNN models is the need to use a weighted loss function to avoid imbalance in accuracy between fall and risk events. For certain training subsets, this fact has meant that the model misses a large number of ADL events, which has had a substantial impact on the overall accuracy.

The standard deviation of the average accuracy of the folds can be observed in Fig. 8 too. As mentioned in the results section, the highest deviation occurs for LSTM model of one layer. Moreover, for all RNN models, the deviation is bigger than the one obtained for MLP models.

Summarizing, although the initial accuracy values for non-guided systems are high, with cross-validation process, the average accuracy of folds decreases, while for guided system the opposite occurs and the average accuracy is centered in similar values for all folds. This difference shows that the guided system is highly robust compared to non-guided systems and that non-guided models require a larger amount of data, more diversified, to obtain similar results.

Regarding the evaluation metrics, Fig. 9 illustrates graphically each metric for all six models. Firstly, it can be observed that the highest values are reached for the specificity metric, overcoming 0.91 in all models. This shows the high number of true negatives versus a lower rate of false positives.

Concerning the precision, more than 0.83 is achieved in all models. This value is elevated but lower than specificity, and it is due to the fact that for ADL class there is a high rate of false positives, with a rate of more than 27% of cases classified as risk or fall instead of activities of daily living. This rate is lower for FALL class but remains quite elevated, which explains the lower values for all six models' precision.

Finally, it can be observed that the results obtained for sensitivity in all cases are the worst. This is mainly due to the lower sensitivity values for risk class in all models. Fall risks occur in a short amount of time compared to falls or ADLs, and they are less characteristic than a fall event, where the acceleration peaks are very clear. Moreover, the way of dealing with a stumble or fall resistance is very varied in each risk event, and the frequency components can easily be mistaken for a fall, if there is a high acceleration peak, or for an ADL event, such as bending down.

As a result, the identification of fall risk is still an unsolved issue, so it could be interesting to go deeper in this research, since fall prevention implies high benefits.

In general, better results are obtained for the four RNN models of the non-guided system, which indicates a better performance of the models classifying between the three possible events than for the MLP models.

Regarding the execution times, Fig. 10 illustrates the summary of the execution times of all models, both using a CPU and a GPU. For MLP models, it can be observed that the execution times are very low both using a CPU and a GPU. Moreover, there is almost no difference between the use of a GPU or a CPU for these models, and the total

Fig. 10. Execution times for all best models. (A) Tests run without GPU optimizations, (B) Tests run with GPU optimizations.

execution time (including the feature extraction process) is less than 4 ms in all cases.

The main difference occurs for RNN models. Firstly, the execution time is higher when GPU and CPU than for MLP models. However, using the GPU, RNN models can take benefit from the power of the GPU and have a faster implementation, but with CPU this advantage is taken off and the execution times are much higher.

In short, non-guided systems obtain higher results and classify better the three possible events. Although, guided systems reflect higher robustness and more independence on the dataset used. Moreover, this one provides faster execution times with and without GPU optimization.

Regarding future embedded implementations, the tests performed without GPU optimization represent in a more reliable way the execution time difference between both methods, as embedded systems do not include the GPU processing powerful.

4.4. Comparative with previous works

In this section, a deep comparison with the previous works explained in Section 2 is done. It is important to mention that the majority of previous works obtained a classification between fall events and non-fall or ADL events. However, the results of this study were focused on the classification of three events (ADL, risk, and fall). Thus, to make a comprehensive comparison, the results of the binary classification between fall and ADL were obtained for all models of the non-guided system and guided system, as well as for Random Forest and SVM models (see Table 21).

Table 21
ADL-FALL prediction. Performance comparison.

Model/ref	Accuracy	Specificity	Precision	Sensitivity/recall	F1-score	AUC
LSTM 1 layer	95.41	0.9667	0.9023	0.9167	0.9096	0.9814
GRU 1 layer	95.91	0.9773	0.9306	0.9047	0.9175	0.9804
LSTM 2 layers	95.36	0.9807	0.9382	0.873	0.9044	0.9782
GRU 2 layers	96.31	0.978	0.9335	0.9187	0.926	0.9870
MLP 1 layer	93.36	0.948	0.852	0.8901	0.871	0.9762
MLP 2 layers	92.91	0.9493	0.8521	0.8691	0.8605	0.9684
Random Forest	88.02	0.8907	0.723	0.8492	0.781	0.8699
Support Vector Machine	93.01	0.96	0.876	0.8413	0.8583	0.9001

Table 22
Comparison with non-guided systems' works detailed in Section 2.

Work	Classifier	Sensor location	Classes	Dataset	Activities	Results
Yoo et al. (2018)	ANN	Wrist	2: fall, ADL	Own (5 men)	7 (1-6)	Ac: 96.9, Se: 94 Sp: 99.3
Santos et al. (2019)	CNN	Pocket	2: fall, ADL	URFD SmartWatch Notch	11 (5-6) 8 (4-4) 11 (4-7)	Ac: 85.7, 98.4, 86.7 Se: 83.3, 91.7, 22.7 Sp: 87.5, 99.5, 99.1 Pr: 83.3, 97, 83.3
Yu et al. (2020)	CNN, RNN, ConvLSTM	Waist	3: fall, no fall, pre-impact	SisFall	34 (15-19)	Ac: 90, 91.6, 93.2 Se: 89.9, 91.5, 93.1 Sp: 91.5, 94, 94.5
Meyer et al. (2020)	BiLSTM	Chest, Thigh	2: fall, ADL	Own (37 patients)	1	Ac: 86, Se: 88 Sp: 83, F1: 86 AUC: 88
Waheed et al. (2021)	BiLSTM	Waist	2: fall, ADL	SisFall UP-Fall	34 (15-19) 11 (5-6)	Ac: 97.2, Se: 96.9 Sp: 91.4
This work	RNN	Ankle	2: fall, ADL 3: fall, risk, ADL	AnkFall	12 (4-3-5)	[2] Ac: 96.3, Se: 91.8 Sp: 97.8, Pr: 93.3 F1: 92.6, AUC: 98.7 [3] Ac: 91.4, Se: 80.4 Sp: 93.2, Pr: 88.3, F1: 83.1 AUC: (97.2, 91.5, 95.8)

ANN: Artificial Neural Network *CNN:* Convolutional Neural Network *RNN: Recurrent Neural Network* *ConvLSTM: Combination of CNN + RNN (LSTM)* *BiLSTM: Bidirectional RNN (LSTM)*.

In order to compare papers using the same feature extraction mechanism, the results are presented in two different tables. Table 22 illustrates the results of previous works of non-guided systems and those of the best RNN model (GRU with two hidden layers), and Table 23 shows the results of previous works of guided systems, as well as the best NN model (MLP with two hidden layers), considering for each study the classifiers used, the number of features extracted for guided systems, the location of the accelerometer, the number and names of the events that are classified, the dataset used, the number and distribution of activities performed and finally the results obtained.

However, despite this division (which is merely organizational and visual), explanations and comments will focus on specific works, so that explanations of guided works can be mixed with non-guided works if it is considered interesting for the reader and/or if they have similar aspects that should be commented on.

Firstly, about non-guided systems studies, in Yoo et al. (2018) classification results were better than those achieved with our RNN model. However, the ANN model used in that work consists of 525 nodes in the input layer with three hidden layers of 500, 500 and 2000 neurons, respectively, which can easily lead to an overfitting problem. Moreover, it should be noted that the dataset contains information of only 5 participants, all of them male, and with one type of falling activity, which reduces significantly the reliability of the results obtained.

The work of Santos et al. (2019) obtained high results, but only higher than our guided and non-guided systems for their SmartWatch dataset. Moreover, they carried out a data augmentation but without using an optimal approach, since they duplicated more than 100 times the URFD dataset and more than 10 times the remaining datasets. Additionally, the samples were highly unbalanced, with less than 20% of fall events.

On the other hand, by comparing the non-guided model of Yu et al. (2020) with our RNN model for three classes, the results are lower in their case for CNN model, but with their convolutional LSTM model and RNN model, all results reach higher values than our system. However, the detected classes consist of fall, non-fall, and pre-impact events, where pre-impact is labeled manually and always leads to a fall event, while our system can distinguish a falling risk event, which does not always end in a fall. For that, it can be assumed that our model provides an added value of risk detection.

In Meyer et al. (2020), it has to be noted that the classification is made between fall and non-fall events, which come from one type of activity, while we obtained a dataset with a distribution of 4 types of ADL activities, 3 types of falling risks and 5 falls. This work applies a non-guided process to a bidirectional LSTM model, which is a more complex model of LSTM, so it explains the higher metrics obtained, yet still lower in comparison with our models. Moreover, the fact that only one activity is used to obtain the dataset and label the data as fall and non-fall can show low variability and can be less reliable than a classification made with different types of activities, more similar to a real environment.

Finally for non-guided systems, the work of Waheed et al. (2021) classified between fall and ADL events, obtaining higher accuracy and sensitivity values than those of our two-class RNN model, but lower specificity values. In addition, the bidirectional LSTM model used in their work is more complex than the RNN model of our study.

About guided systems, the study made by Howcroft et al. (2018) classifies between fall and non-fall events by extracting 87 features. It is worth noting that in their work, besides acceleration data, they also used a pressure-sensing insole and applied Naïve Bayesian algorithm only to the insole, achieving higher results, but to compare properly

Table 23
Comparison with guided systems' works detailed in Section 2.

Work	Classifier	Features	Sensor location	Classes	Dataset	Activities	Results
Howcroft et al. (2018)	SVM, ANN	87	Head, Pelvis, Leg	2: fall, no fall	Own (75 people)	1	Ac: 78.9, 77.8 Se: 100, 57.1 Sp: 63.6, 91.7 F1: 77.8, 66.7
Putra et al. (2018)	kNN, LR, SVM	27	Chest, Waist	2: fall, ADL	Cogent SisFall	13 (6-7) 34 (15-19)	Se: 94.5, 94.6, 92.7 Pr: 87.5, 88.4, 90.3 F1: 90.7, 91.3, 91.1
Khojasteh et al. (2018)	ANN, DT, RBS, SVM	24	Wrist	2: fall, no fall	UMAFall DaLiac	11 (3-8) 13 (0-13)	Ac: 89, 87, 90, 92 Se: 87, 87, 88, 89 Sp: 92, 86, 91, 95 Pr: 94, 90, 93, 96
Chen et al. (2019)	Ensemble 4xANN	4 × 30	Wrist	2: fall, ADL	Own (11 people)	18 (3-15)	Se: 99.1 Sp: 96.2
Rivolta et al. (2019)	LRM, ANN	21	Chest	1: ADL	Own (79 patients)	8 (0-8)	Se: 71, 86 Sp: 81, 90
Hassan et al. (2019)	ConvLSTM	58	Pocket	2: fall, no fall 3: fall, standing, lying	MobiAct	13 (4-9)	[2]Ac: 97, Se: 97 Pr: 97, F1: 97 AUC: (97, 97) [3] Ac: 96.7, Se: 97 Pr: 97, F1: 97 AUC: (96, 98, 95)
Wang et al. (2020)	CNN	54	Waist	2: fall, ADL	SisFall	34 (15-19)	Ac: 99.1
Jansi et al. (2020)	Threshold	5	Waist	2: fall, ADL	URFD	11 (5-6)	Ac: 84.2, Se: 100 Sp: 72.5, Pr: 73.1
Althobaiti et al. (2020)	LDA, DT, SVM, ANN	72	Chest	2: fall, ADL	Own (35 people)	7 (1-6)	Ac: 98.3, 97.1, 98.4, 98.1 F1: 98.1, 97, 98.4, 98
Alarifi et al. (2021)	CNN	1404	Chest, Waist, Head, Both Wrists, Both Ankles	2: fall, ADL	Own (14 people)	36 (20-16)	Se: 99.5, Pr: 99.4 F1: 99.5
Galvão et al. (2021)	CNN, RNN	165	Waist	2: fall, ADL	URFD UP-Fall	11 (5-6) 11 (5-6)	Ac: 93.2, 85.3
This work	ANN	42	Ankle	2: fall, ADL 3: fall, risk, ADL	AnkFall	12 (4-3-5)	[2] Ac: 93.4, Se: 89 Sp: 94.9, Pr: 85.2 F1: 87.1, AUC: 97.6 [3] Ac: 88.1, Se: 76.8 Sp: 91.4, Pr: 83.9, F1: 79.1 AUC: (96.4, 91.3, 94.6)

LR: Linear Regression *DT: Decision Tree RBS: Rule-Based System LRM: Linear Regression Model kNN: k-Nearest Neighbors LDA: Linear Discriminant Analysis SVM: Support Vector Machine.*

with our work, only the results related to acceleration data are shown. Comparing these results with our work, it can be observed that the number of extracted features is more than twice as much as our guided system, and they obtained data from three different locations while our acceleration data is from only one location. Moreover, the dataset used of falls and non-falls only have one type of activity, as was the case in Meyer et al. (2020). According to the evaluation metrics, it can be observed that their work achieves lower values than our guided and non-guided models, except for the case of their SVM model where the sensitivity achieves 100%.

Comparing the results of the guided model of Putra et al. (2018) with our ANN model, their results reach higher values, although they are lower than our RNN model. In addition to having better results than our guided system, the number of extracted features is lower. However, the dataset used in their work is highly unbalanced, with lower samples for fall events.

Khojasteh et al. (2018) also used a guided system with lower number of features than our model, obtaining higher precision values for all their models in comparison with our ANN model. However, the remaining metrics reach lower values than those obtained with our system. Besides, the features extracted are spatial features, while our work intends to demonstrate that the extraction of features in the time-frequency domain combined with spatial features is more informative and proves by comparing this study with our work that our approach can reach better results.

Chen et al. (2019) implements a guided system extracting 24 spatial features and 6 frequency-domain features from a dataset composed of 13 fall activities categorized as 3 types of fall events, and 16 ADLs. Comparing the two metrics obtained with our results, they reach higher values but with a smaller and more unbalanced dataset.

The study made by Rivolta et al. (2019) extracts a lower number of features and used a bigger dataset than AnkFall dataset. However, the metrics obtained show lower results for this approach in comparison with our systems, even for the classification of our models for three events results are better, which is a more reliable and accurate classification.

The results obtained in Hassan et al. (2019) are higher than the results reached in our models for two and three classes. However, taking a closer look to their study, it can be observed that their model classifies between falls, standing, and lying down events, which are very distinguishable states compared to the difficulty to differentiate fall from activities in constant motion like ADLs.

On the other hand, the results from Wang et al. (2020) reach higher accuracy values than all of our models and by using a different approach with statistical features.

The work of Jansi et al. (2020) applies a threshold algorithm to detect fall events. Although they also use data from a Kinect sensor, we only illustrated those results obtained with acceleration data. It can be observed that all metrics are lower than the metrics of our

models, except for the sensitivity value that reaches 100%. It has to be noted that the dataset used in the study is formed by 30 different falls and 40 ADLs, but in general very similar activities and with very few participants, which does not provide highly reliable results.

The results of the work of Althobaiti et al. (2020) show much higher values than those obtained with our systems. However, their metrics are presented for the entire dataset, including training and testing results, whereas all studies (including ours) illustrate only testing results. In general terms, for machine learning models, the training results are usually higher than the testing results, in addition to having a higher train set than the test set. Thus, by joining both results, the final metrics should be higher than those for the test set only.

The study of Alarifi et al. (2021) consists of a guided system with a CNN model, extracting 1404 features and obtaining much higher metric values compared to our guided model. However, our ANN model has a very low number of features extracted in comparison with this study, which should reduce significantly the complexity and the computational load.

Finally, Galvão et al. (2021) used acceleration data to detect falls but also data from other mechanisms with which they reached higher values. In this comparison, as mentioned earlier and to relate all works to acceleration data, the results from data obtained with mechanisms other than accelerometers were discarded. It can be observed that the results for the acceleration data only are lower than those obtained with our guided model for the 2-class implementation, in addition to having fewer features extracted in our model.

Overall, the majority of previous works used guided systems with feature extraction process. However, none of them applied the same approach as our study, in which frequency features from a DWT are extracted and combined with temporal features. Some studies that detected fall events with a guided system achieved better results than those obtained with our models. However, those works usually used a highly unbalanced dataset or contained few samples (Putra et al., 2018), or used a more complex model with a higher number of extracted features (Chen et al., 2019; Hassan et al., 2019; Wang et al., 2020; Althobaiti et al., 2020; Alarifi et al., 2021). Moreover, these works do not present results related to execution times, as they do not consider their integration in embedded systems to operate in real time; but in our work, this premise is fundamental and, therefore, we try to simplify the implemented models and perform a meticulous study regarding its execution times.

On the other hand, other studies applied a non-guided system to classify between fall and ADL events, and three of them reached higher results than our non-guided RNN model. Nevertheless, the first best model implemented a more complex model (Waheed et al., 2021); the second best performed a less optimal data augmentation for the dataset (Santos et al., 2019); and the third best used a small dataset with very low diversity (only male) (Yoo et al., 2018).

It is clear in all previous works presented that with both types of systems, guided or non-guided, high results can be achieved. However, it remains unclear which system is the best option from these previous results, in terms of accuracy and computational load. Our study intends to resolve or further help the understanding of the differences between both systems by making a more comprehensive and reliable comparison since the same dataset and parameters are used for guided and non-guided systems. It is important to note that for a FDS, it is necessary that the system is integrated in a wearable device in order to obtain a successful detection and, for this to be achieved, the classifier used should have low computational load. From the comparison conducted in this study, it is noted that the best option is the guided system based on the MLP classifier,since the computational load is greatly reduced, with which the response time of the alert system that can be implemented is much faster, the performance of the system is reduced and the autonomy of the system increases, while the drop in the value of the accuracy is acceptable as it is very slight.

Finally, it is worth noting that, even though 9 out 16 previous works exposed obtained higher results, our work also classifies the falling risk events, providing the benefit and novelty of preventing falls, while other studies that classify between three classes do not detect falling risk events.

5. Conclusions

The main objective of this work is to compare two types of existing systems and present a comprehensive assessment of which is more appropriate to use in classification models. For that purpose, the suitability of using a guided feature extraction pre-processing to reduce the complexity of a classifier system with respect to a more complex classifier with a non-guided feature extraction process is studied. To carry out this study, the research process is applied to fall detection systems (FDS) using a dataset that distinguishes between three types of activities: ADL, falling risk and fall.

The two systems implemented are, on the one hand, a system based on recurrent neural networks (non-guided system) and, on the other hand, a system based on a classical multilayer perceptron-based artificial neural network using temporal and frequency features (guided model). Both systems have followed an identical design, implementation, and optimization process, in which a grid search with multiple combinations of hyperparameters has been carried out, the best candidates have been selected and subjected to a third phase of cross-validation and hold-out to obtain the final models used for the comparison.

Results have shown higher evaluation metrics for non-guided systems, where the accuracy reached more than 91% accuracy for the best RNN model and more than 50% of risk samples were correctly classified, which is high according to the difficulty of identifying falling risks. On the other hand, for guided systems with lower metric values (higher than 88% accuracy), we obtained a better performance in cross-validation process, revealing that this approach has a higher robustness. The guided models have also shown to have lower execution times in non-GPU-powered systems (less than 10% of the time required for the unguided system). These results show that, if the most useful features of the signal are correctly extracted, the results are very similar to those obtained in non-guided systems (and may even surpass them); moreover, the robustness of the guided system has been shown to be superior, as well as its suitability for use in systems with low computational resources (such as embedded systems).

In addition, a literature review has been carried out on the most relevant works in the field of FDS in recent years. It is shown that the tendency is to mainly use guided systems; moreover, it is certified that the greatest difficulty in these systems is to find the most useful features, since there is a great variability in the number and type of features used in previous works.

Finally, the results obtained in this work have been compared with previous works, showing the benefits of the systems designed, providing a solution that is computationally lighter and can be integrated into a real-time embedded system, in addition to the detection of falling risk events, which are not commonly detected.

In future works, the integration of the guided model in embedded systems will be studied. In this way, the performance of a less complex classifier to detect fall events in real time can be evaluated and it can be determined whether the loss of accuracy in real-time systems versus a better execution time is worthwhile.

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.

Acknowledgments

This work was supported by the Andalusian Regional I+D+i FEDER Projects DAFNE US-1381619 and MSF-PHIA US-1263715.

References

Alarifi, A., et al., 2021. Killer heuristic optimized convolution neural network-based fall detection with wearable IoT sensor devices. Measurement 167, 108258. http://dx.doi.org/10.1016/j.measurement.2020.108258.

Alkhodari, M., Fraiwan, L., 2021. Convolutional and recurrent neural networks for the detection of valvular heart diseases in phonocardiogram recordings. Comput. Methods Programs Biomed. 200, 105940. http://dx.doi.org/10.1016/j.cmpb.2021.105940.

Althobaiti, T., et al., 2020. Triaxial accelerometer-based falls and activities of daily life detection using machine learning. Sensors 20 (13), 3777. http://dx.doi.org/10.3390/s20133777.

Arif, A., 2015. Physical activities monitoring using wearable acceleration sensors attached to the body. PLOS ONE 10, 1–16. http://dx.doi.org/10.1371/journal.pone.0130851.

Ayrulu-Erdem, B., Barshan, B., 2011. Leg motion classification with artificial neural networks using wavelet-based features of gyroscope signals. Sensors 11, 1721–1743. http://dx.doi.org/10.3390/s110201721.

Azar, J., et al., 2021. Deep recurrent neural network-based autoencoder for photoplethysmogram artifacts filtering. Comput. Electr. Eng. 92, 107065. http://dx.doi.org/10.1016/j.compeleceng.2021.107065.

Braccesi, C., et al., 2015. Random multiaxial fatigue: A comparative analysis among selected frequency and time domain fatigue evaluation methods. Int. J. Fatigue 74, 107–118. http://dx.doi.org/10.1016/j.ijfatigue.2015.01.003.

Bruce, L., et al., 2002. Dimensionality reduction of hyperspectral data using discrete wavelet transform feature extraction. IEEE Trans. Geosci. Remote Sens. 40 (10), 2331–2338. http://dx.doi.org/10.1109/TGRS.2002.804721.

Buongiorno, D., et al., 2021. Deep learning for processing electromyographic signals: A taxonomy-based survey. Neurocomputing 452, 549–565. http://dx.doi.org/10.1016/j.neucom.2020.06.139.

Casal, R., et al., 2021. Classifying sleep–wake stages through recurrent neural networks using pulse oximetry signals. Biomed. Signal Process. Control 63, 102195. http://dx.doi.org/10.48550/arXiv.2008.03382.

Chakraborty, J., Nandy, A., 2020. Discrete wavelet transform based data representation in deep neural network for gait abnormality detection. Biomed. Signal Process. Control 62, 102076. http://dx.doi.org/10.1016/j.bspc.2020.102076.

Chatterjee, A., et al., 2019. Understanding emotions in text using deep learning and big data. Comput. Hum. Behav. 93, 309–317. http://dx.doi.org/10.1016/j.chb.2018.12.029.

Chen, L., et al., 2019. Intelligent fall detection method based on accelerometer data from a wrist-worn smart watch. Measurement 140, 215–226. http://dx.doi.org/10.1016/j.measurement.2019.03.079.

Cho, K., et al., 2014. On the properties of neural machine translation: Encoder-decoder approaches. http://dx.doi.org/10.48550/arXiv.1409.1259, arXiv preprint arXiv:1409.1259.

Cuevas-Trisan, R., 2019. Balance problems and fall risks in the elderly. Clin. Geriatr. Med. 35 (2), 173–183. http://dx.doi.org/10.1016/j.pmr.2017.06.006.

Daneshjou, R., et al., 2021. How to evaluate deep learning for cancer diagnostics–factors and recommendations. Biochim. Biophys. Acta-Reviews on Cancer 1875 (2), 188515. http://dx.doi.org/10.1016/j.bbcan.2021.188515.

Do, H.M., et al., 2018. Rish: A robot-integrated smart home for elderly care. Robot. Auton. Syst. 101, 74–92. http://dx.doi.org/10.1016/j.robot.2017.12.008.

Farsi, M., 2021. Application of ensemble RNN deep neural network to the fall detection through IoT environment. Alex. Eng. J. 60 (1), 199–211. http://dx.doi.org/10.1016/j.aej.2020.06.056.

Fu, Y., et al., 2021. A review of deep learning based methods for medical image multi-organ segmentation. Phys. Medica 85, 107–122. http://dx.doi.org/10.1016/j.ejmp.2021.05.003.

Galvão, Y., et al., 2021. A multimodal approach using deep learning for fall detection. Expert Syst. Appl. 168, 114226. http://dx.doi.org/10.1016/j.eswa.2020.114226.

González-Cañete, F.J., Casilari, E., 2021. A feasibility study of the use of smartwatches in wearable fall detection systems. Sensors 21 (6), 2254. http://dx.doi.org/10.3390/s21062254.

Guyon, I., Elisseeff, A., 2006. An introduction to feature extraction. In: Guyon, I., Nikravesh, M., Gunn, S., Zadeh, L.A. (Eds.), Feature Extraction: Foundations and Applications. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–25. http://dx.doi.org/10.1007/978-3-540-35488-8_1.

Hassan, M.M., et al., 2019. A smartphone-enabled fall detection framework for elderly people in connected home healthcare. IEEE Netw. 33 (6), 58–63. http://dx.doi.org/10.1109/MNET.001.1900100.

Hertel, L., Phan, H., Mertins, A., 2016. Comparing time and frequency domain for audio event recognition using deep learning. In: 2016 International Joint Conference on Neural Networks (IJCNN). pp. 3407–3411. http://dx.doi.org/10.1109/IJCNN.2016.7727635.

Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. http://dx.doi.org/10.48550/arXiv.1207.0580, arXiv preprint arXiv:1207.0580.

Hochreiter, S., 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 6 (02), 107–116. http://dx.doi.org/10.1142/S0218488598000094.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hoo, Z.H., et al., 2017. What is an ROC curve?. http://dx.doi.org/10.1136/emermed-2017-206735.

Howcroft, J., et al., 2018. Prospective elderly fall prediction by older-adult fall-risk modeling with feature selection. Biomed. Signal Process. Control 43, 320–328. http://dx.doi.org/10.1016/j.bspc.2018.03.005.

Huang, Z., et al., 2018. Video-based fall detection for seniors with human pose estimation. In: 2018 4th International Conference on Universal Village (UV). IEEE, pp. 1–4. http://dx.doi.org/10.1016/j.bspc.2018.03.005.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. PMLR, pp. 448–456. http://dx.doi.org/10.48550/arXiv.1502.03167.

Jackson, R.E., et al., 2017. Barriers to falling risk. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting. Vol. 61, SAGE Publications Sage CA: Los Angeles, CA, pp. 1680–1684. http://dx.doi.org/10.1177/1541931213601909.

Jansi, R., et al., 2020. Detection of fall for the elderly in an indoor environment using a tri-axial accelerometer and kinect depth data. Multidimens. Syst. Signal Process. 31 (4), 1207–1225. http://dx.doi.org/10.1007/s11045-020-00705-4.

Ji, N., Ma, L., Dong, H., Zhang, X., 2019. Eeg signals feature extraction based on DWT and EMD combined with approximate entropy. Brain Sci. 9 (8), http://dx.doi.org/10.3390/brainsci9080201, URL https://www.mdpi.com/2076-3425/9/8/201.

Jia, W., et al., 2022. Feature dimensionality reduction: a review. Complex Intell. Syst. http://dx.doi.org/10.1007/s40747-021-00637-x.

Khojasteh, S.B., et al., 2018. Improving fall detection using an on-wrist wearable accelerometer. Sensors 18 (5), 1350. http://dx.doi.org/10.3390/s18051350.

Lee, Y., et al., 2018. A real-time fall detection system based on the acceleration sensor of smartphone. Int. J. Eng. Bus. Manage. 10, 1847979017750669. http://dx.doi.org/10.1177/1847979017750669.

Lotfi, A., et al., 2018. Supporting independent living for older adults; employing a visual based fall detection through analysing the motion and shape of the human body. IEEE Access 6, 70272–70282. http://dx.doi.org/10.1109/ACCESS.2018.2881237.

Luna-Perejon, F., et al., 2019. An automated fall detection system using recurrent neural networks. In: Conference on Artificial Intelligence in Medicine in Europe. Springer, pp. 36–41. http://dx.doi.org/10.1007/978-3-030-21642-9_6.

Luna-Perejón, F., et al., 2019. Wearable fall detector using recurrent neural networks. Sensors 19 (22), 4885. http://dx.doi.org/10.3390/s19224885.

Luna-Perejón, F., et al., 2021a. Ankfall—Falls, falling risks and daily-life activities dataset with an ankle-placed accelerometer and training using recurrent neural networks. Sensors 21 (5), 1889. http://dx.doi.org/10.3390/s21051889.

Luna-Perejón, F., et al., 2021b. IoT garment for remote elderly care network. Biomed. Signal Process. Control 69, 102848. http://dx.doi.org/10.1016/j.bspc.2021.102848.

Mazurowski, M.A., et al., 2019. Deep learning in radiology: An overview of the concepts and a survey of the state of the art with focus on MRI. J. Magn. Reson. Imaging 49 (4), 939–954. http://dx.doi.org/10.1002/jmri.26534.

Meyer, B., et al., 2020. Wearables and deep learning classify fall risk from gait in multiple sclerosis. IEEE J. Biomed. Health Inf. 25 (5), 1824–1831. http://dx.doi.org/10.1109/JBHI.2020.3025049.

Mitchell, E., et al., 2013. Classification of sporting activities using smartphone accelerometers. Sensors 13, 5317–5337. http://dx.doi.org/10.3390/s130405317.

Moran, K., et al., 2015. Detection of running asymmetry using a wearable sensor system. Procedia Eng. 112, 180–183. http://dx.doi.org/10.1016/j.proeng.2015.07.196, URL https://www.sciencedirect.com/science/article/pii/S1877705815014459, 'The Impact of Technology on Sport VI' 7th Asia-Pacific Congress on Sports Technology, APCST2015.

Putra, I., et al., 2018. An event-triggered machine learning approach for accelerometer-based fall detection. Sensors 18 (1), 20. http://dx.doi.org/10.3390/s18010020.

Rivolta, M.W., et al., 2019. Evaluation of the tinetti score and fall risk assessment via accelerometry-based movement analysis. Artif. Intell. Med. 95, 38–47. http://dx.doi.org/10.1016/j.artmed.2018.08.005.

Rizk, H., et al., 2021. Device-independent cellular-based indoor location tracking using deep learning. Pervasive Mob. Comput. 75, 101420. http://dx.doi.org/10.1016/j.pmcj.2021.101420.

Santos, G.L., et al., 2019. Accelerometer-based human fall detection using convolutional neural networks. Sensors 19 (7), 1644. http://dx.doi.org/10.3390/s19071644.

Sarabia-Jácome, D., et al., 2020. Highly-efficient fog-based deep learning AAL fall detection system. Int. Things 11, 100185. http://dx.doi.org/10.1016/j.iot.2020.100185.

Sekine, M., et al., 1998. Classification of acceleration waveform in a continuous walking record. In: Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. Vol. 20 Biomedical Engineering Towards the Year 2000 and beyond (Cat. No.98CH36286). Vol. 3, pp. 1523–1526. http://dx.doi.org/10.1109/IEMBS.1998.747177.

Sokolova, M., et al., 2009. A systematic analysis of performance measures for classification tasks. Inf. Process. Manag. 45 (4), 427–437. http://dx.doi.org/10.1016/j.ipm.2009.03.002.

Songra, R., Lockhart, T., Berge, N., 2011. An approach for identifying gait events using wavelet denoising technique and single wireless IMU. Proc. Hum. Factors Ergonomics Soc. Annu. Meeting 55, 1990–1994. http://dx.doi.org/10.1177/1071181311551415.

Stylianou, N., Vlahavas, I., 2021. Transformed: End-to-end transformers for evidence-based medicine and argument mining in medical literature. J. Biomed. Inform. 117, 103767. http://dx.doi.org/10.1016/j.jbi.2021.103767.

Syed, A., Sierra-Sosa, D., Kumar, A., Elmaghraby, A., 2021. A hierarchical approach to activity recognition and fall detection using wavelets and adaptive pooling. Sensors (Basel, Switzerland) 21, http://dx.doi.org/10.3390/s21196653.

Too, J., et al., 2017. Classification of EMG signal based on time domain and frequency domain features. Int. J. Hum. Technol. Interact. 1 (1), 25–30.

Toraman, A., et al., 2010. The falling risk and physical fitness in older people. Arch. Gerontol. Geriat. 51 (2), 222–226. http://dx.doi.org/10.1016/j.archger.2009.10.012.

Torti, E., et al., 2019. Embedding recurrent neural networks in wearable systems for real-time fall detection. Microprocess. Microsyst. 71, 102895. http://dx.doi.org/10.1016/j.micpro.2019.102895.

Varuna Shree, N., Kumar, T.N.R., 2018. Identification and classification of brain tumor MRI images with feature extraction using DWT and probabilistic neural network. Brain Inform. 5 (1), 23–30. http://dx.doi.org/10.1007/s40708-017-0075-5.

Waheed, M., et al., 2021. NT-FDS—A noise tolerant fall detection system using deep learning on wearable devices. Sensors 21 (6), 2006. http://dx.doi.org/10.3390/s21062006.

Wang, G., et al., 2020. CMFALL: A cascade and parallel multi-state fall detection algorithm using waist-mounted tri-axial accelerometer signals. IEEE Trans. Consum. Electron. 66 (3), 261–270. http://dx.doi.org/10.1109/TCE.2020.3000338.

Williams, R.J., Zipser, D., 1995. Gradient-based learning algorithms for recurrent. Backpropagation: Theory, Archit., Appl. 433, http://dx.doi.org/10.5555/201784.201801.

World Health Organization, 2021. Falls (facts sheet, 21 april 2021). URL https://www.who.int/news-room/fact-sheets/detail/falls, (Accessed on January 15, 2022).

Yang, J., 2009. Toward physical activity diary: Motion recognition using simple acceleration features with mobile phones. In: ACM International Workshop on Interactive Multimedia for Consumer Electronics. Vol. 39, pp. 1–10. http://dx.doi.org/10.1145/1631040.1631042.

Yoo, S., et al., 2018. An artificial neural network–based fall detection. Int. J. Eng. Bus. Manage. 10, http://dx.doi.org/10.1177/1847979018787905, 1847979018787905.

Yu, X., et al., 2020. A novel hybrid deep neural network to predict pre-impact fall for older people based on wearable inertial sensors. Front. Bioeng. Biotechnol. 63. http://dx.doi.org/10.3389/fbioe.2020.00063.

Zhao, B., Li, S., Gao, Y., Li, C., Li, W., 2020. A framework of combining short-term spatial/frequency feature extraction and long-term indrnn for activity recognition. Sensors 20 (23), http://dx.doi.org/10.3390/s20236984, URL https://www.mdpi.com/1424-8220/20/23/6984.