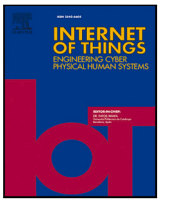
Internet of Things 24 (2023) 100925

Contents lists available at ScienceDirect

Internet of Things

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



Research article

Deep learning models for real-life human activity recognition from smartphone sensor data

Daniel Garcia-Gonzalez *, Daniel Rivero, Enrique Fernandez-Blanco, Miguel R. Luaces

Department of Computer Science and Information Technologies, University of A Coruna, CITIC, 15071 A Coruna, Spain

ARTICLE INFO

Dataset link: http://lbd.udc.es/research/real-life-HAR-dataset, https://data.mendeley.com/datasets/3xm88g6m6d/2, http://gitlab.lbd.org.es/dgarcia/deep-learning-models-har

Keywords:
HAR
CNN
LSTM
Real life
Smartphones
Sensors

ABSTRACT

Nowadays, the field of human activity recognition (HAR) is a remarkably hot topic within the scientific community. Given the low cost, ease of use and high accuracy of the sensors from different wearable devices and smartphones, more and more researchers are opting to do their bit in this area. However, until very recently, all the work carried out in this field was done in laboratory conditions, with very few similarities with our daily lives. This paper will focus on this new trend of integrating all the knowledge acquired so far into a real-life environment. Thus, a dataset already published following this philosophy was used. In this way, this work aims to be able to identify the different actions studied there. In order to perform this classification, this paper explores new designs and architectures for models inspired by the ones which have yielded the best results in the literature. More specifically, different configurations of Convolutional Neural Networks (CNN) and Long-Short Term Memory (LSTM) have been tested, but on real-life conditions instead of laboratory ones. It is worth mentioning that the hybrid models formed from these techniques yielded the best results, with a peak accuracy of 94.80% on the dataset used.

1. Introduction

Research in the field of human activity recognition (HAR) has shown stable progress in recent times. With the rise of wearable devices (mainly bracelets) and, above all, smartphones, it is feasible to think about the possibility of transferring the work carried out in this area to a large part of the world's population. To this end, sensor data from these devices are analysed in search of the classification of actions performed by a particular individual [1–3]. In this way, the applications of the work carried out in this field are multiple, from healthcare [4–6] to fitness [7,8], as well as more specific cases such as home automation [9]. For all these reasons, and thanks to the high portability and accuracy of the sensors of these devices, researchers find in HAR an incredibly tempting research opportunity [10–12].

However, there are some problems that need to be tackled. Firstly, there is the need to handle the temporality of the data, which is especially difficult when dealing with the large amount of information these devices produce. While it is true that previous works have made significant advances [13–15], there are still many activities whose relationship with prior data is still unclear. In addition, most of those works are carried out in a laboratory environment, with a series of pretty specific conditions that are not entirely feasible to transfer to real life. Although these works are helpful to get an approximate idea of the information collected and the actions performed, their outstanding results for the cases studied are very relative. One of the main issues is that the orientation and positioning of the device during the experimentation time can notably affect the final result [16]. Most researchers work with

* Corresponding author.
E-mail addresses: d.garcia2@udc.es (D. Garcia-Gonzalez), daniel.rivero@udc.es (D. Rivero), enrique.fernandez@udc.es (E. Fernandez-Blanco), miguel.luaces@udc.es (M.R. Luaces).

https://doi.org/10.1016/j.iot.2023.100925
Received 20 June 2023; Received in revised form 28 July 2023; Accepted 30 August 2023
Available online 9 September 2023

a smartphone around the waist [17] or using a wristband or bracelet [18]. Those developments could lead to remarkably reduced performances if they are applied to other datasets, especially real-life ones. In fact, specifically for the case of smartphones in everyday life, each person carries and uses them in a different way. That would highly affect the data provided by their sensors, as the previously mentioned orientation and positioning would vary considerably. Even when using different models of smartphones, differences in final measurements may occur [19]. Moreover, in addition to the latter, each individual has a series of physical peculiarities that could also influence the final result, even when using the device in the same way and performing the same action [20]. In fact, this problem has been studied for years, in order to personalise artificial intelligence models for large numbers of people [21,22].

For all those reasons, this work sought to help close that gap between all the acquired knowledge in HAR and its application in real life. To this end, a dataset already formed for this purpose was used [23]. The data was taken from the personal smartphone sensors of 19 individuals, ensuring that the actions performed were as similar as possible to those in their daily lives. To date, some work has been done using such a dataset, but using techniques related to traditional machine learning [24,25]. In this way, a study was carried out on the most suitable configurations for the deep learning algorithms that are yielding the best results in HAR: Convolutional Neural Networks (CNN) and Long-Short Term Memory models (LSTM) [26,27]. To this end, based on these techniques, new architectures have been designed to exploit this real-life data. In such a manner, it is hoped to obtain results that, if not ideal, are close to the optimum that is being sought.

Therefore, the key findings of this paper can be condensed into the following statements:

- An in-depth exploration of the most appropriate configurations for CNN and LSTM networks with real-world data in the HAR domain, using smartphone sensors.
- A new architecture to exploit HAR data taken from different smartphone sensors in a daily life environment, applying deep learning algorithms.
- The use of much more straightforward models than those used in previous real-life HAR domain work, without the need to manually compute its features.
- The improvement of the current results and approaches applying deep learning to a real-life HAR dataset.

The rest of the paper is organised as follows: Section 2 focuses on the evolution of HAR and the most relevant and recent work in this area, Section 3 describes the deep learning algorithms selected to perform all the related experiments, Section 4 depicts the preparation of the data and highlights the evaluation and validation techniques used, together with the proposed architectures and models, Section 5 discusses the main results of the work, and finally, Section 6 contains a series of conclusions and possible lines of future work.

2. Related work

This section is divided into two distinct parts. Firstly, in Section 2.1, a detailed comparison is made between the main datasets used by the scientific community and the one used in this paper. Then, in Section 2.2, a series of notable recent works that made use of those datasets are presented.

2.1. Smartphone datasets

Over the past decade, there have been numerous contributions to human activity recognition (HAR), leading to continuous advancements in the field. These developments have been supported by various datasets used as benchmarks to validate experiments and expand knowledge in the domain [28]. The data within these datasets originate from distinct wearable devices, such as activity wristbands, heart rate monitors, and more recently, smartphones. Among the latter, the UCI HAR dataset was the most widely used one by the scientific community [29]. It focused on activities like walking, sitting, and going upstairs, using data from the accelerometer and gyroscope of a specific smartphone. In addition, 30 participants were involved in the study, placing the smartphone on the left side of their waist. Each activity was performed for a few seconds to collect relevant features. Finally, the output data were sampled at a frequency of 50 Hz, and all the data collection took place in a laboratory setting.

The WISDM dataset [30] is another widely used dataset for human activity recognition, alongside the UCI HAR dataset. The activities included in this dataset are highly similar to those found in the UCI HAR one. Additionally, both datasets involve studying activities performed for several seconds. However, the main difference lies in the placement of the smartphone. In the case of the WISDM dataset, the smartphone was positioned in one of the front trouser pockets of each of the 29 participants who took part in the study. Unlike the UCI HAR dataset, the WISDM one only uses accelerometer data, sampling them at a fixed frequency of 20 Hz. As with the previous dataset, the data collection process for the WISDM dataset was also carried out under controlled laboratory conditions.

Similarly, the HHAR dataset [19] gathered data from eight smartphones and four smartwatches. The smartphones included four different models, while the smartwatches consisted of two distinct types. To collect the data, each participant had the smartphones securely placed in a pouch attached to their waist, and two smartwatches were worn on each wrist. The study involved only nine individuals as participants. As for the activities performed in this dataset, these were basic examples like walking, cycling, or running, but they were recorded over a more extended period of five minutes. Unlike the previous datasets mentioned, the data collection for HHAR did not take place in a laboratory setting. Instead, participants were instructed to follow specific routes within designated

Table 1
Comparison of the main HAR datasets based on smartphone sensor data, along with the real-life one used in this paper.

Dataset	Sensor(s) used	Activities recording time	Number of subjects	Sampling frequency	Device(s) used	Device placement	Environment
UCI HAR	Acc. and gyro.	Few seconds	30	50 Hz	1 smartphone	Left belt	Controlled
WISDM	Acc.	Few seconds	29	20 Hz	1 smartphone	Front pants leg pocket	Controlled
HHAR	Acc. and gyro.	5 min	9	Variable	8 smartphones and 4 smartwatches	Waist and wrist	Semi-controlled
UniMiB SHAR	Acc.	Fixed flow duration	30	50 Hz	1 smartphone	Trouser front pockets	Controlled
Real-life dataset	Acc., gyro., magn. and GPS	Free	19	Variable	19 personal smartphones	Free	Free

timeframes. Regarding the sampling rate, efforts were made to use the maximum value supported by Android. However, there was finally some variability in the sampling rates recorded during the study.

Another noteworthy dataset is the UniMiB SHAR one [31]. For data collection, a specific smartphone was positioned in the front trouser pocket of each of the 30 participants. Unlike some previous datasets, only accelerometer data were used, sampled at a fixed frequency of 50 Hz. Regarding the activities studied, these encompassed walking, standing up, running, jumping, and various others. The entire data collection process was conducted under controlled laboratory conditions, with researchers guiding the participants through specific activities.

As can be seen, the mentioned datasets exhibit significant differences among them. However, they all share a limitation in their data-gathering conditions. Specifically, the measuring devices were fixed to specific body parts, and the activities were performed in predetermined ways for set durations. To address this limitation, the current paper employed a real-life dataset in which the participants carried out the specified activities in a more natural and unrestricted way. In addition, in this dataset, data were collected from participants' personal smartphones, allowing them to carry out and measure the actions as they do regularly, with the smartphone positioned in their preferred habitual manner. In this way, Table 1 presents a summary of key information from each discussed dataset, comparing them to the one used in this work. Note that the abbreviations used in the table correspond to accelerometer (acc.), gyroscope (gyro.), and magnetometer (magn.). There, several distinctions are evident with the real-life dataset. Firstly, including the GPS sensor in data collection is a significant difference. This sensor's ability to detect speed and orientation could be beneficial for classifying the activities under study. Moreover, another noteworthy contrast is the variability in the sensors' sampling frequencies, which deviates from most datasets found in the literature. Unlike other measurement devices that can be set to a specific frequency value, smartphones lack this consistency throughout the data collection process, even if the highest value supported by the smartphone's operating system is set. This variability may not pose a problem for short and controlled data collection, as corrupted data will be minimal. However, for longer durations, such as in the chosen dataset, that needs to be considered, and appropriate data processing will be required. Furthermore, as for the number of participants and the use of different device models, higher variability would be preferred to ensure a more reliable representation of real-world contexts. In this way, the main difference lies in how the data were collected in a free environment with no specific conditions, making the proposals using other datasets less applicable to real-life scenarios.

2.2. Latest approaches

The introduction of wearable devices and widespread smartphone usage has significantly boosted the development of HAR. Since then, there has been a continuous rise in the diversity, improvement, and optimisation of artificial intelligence models that use this type of data. Following a chronological order, in the first years of the last decade, many works focused mainly on the exploitation of Support Vector Machines (SVM), as they seemed to be the models that yielded the best results for this subject [32,33]. Later, other possibilities began to be explored, as in the case of [34]. There, a comparison was carried out between other machine learning algorithms that also get good results in more fields, such as K-Nearest Neighbours (KNN), Multi-layer Perceptron (MLP) or the ones based on Bayes' Theorem. However, SVM still presented the best results to that date. In fact, later, another paper was also published in which an analysis of the principal machine learning algorithms used globally was also carried out, together with SVM [35]. Once again, SVM proved to be the most suitable for HAR. However, they also carried out a study on the influence of smartphone orientations on the returned data. The results showed that variations in this respect could significantly affect the final results. In the same way, work was also carried out on selecting the most convenient features to train these models, such as [36,37]. The results shown by these works proved that frequency-based parameters seemed to be the most suitable for HAR, having the highest percentage of correctness in the trained classifiers.

More recently, other works have been carried out in which deep learning approaches have been applied. Some of the most relevant ones are those of [38,39], for proving the high-grade results obtained by applying deep learning techniques, specifically Convolutional Neural Networks (CNN), on data from the HAR domain. In this way, a detailed comparison between different machine learning algorithms, combined with some custom features, is subsequently presented in [14]. The algorithms used were: CNN, Random Forest (RF) and KNN. Out of all the methods tested, CNN yielded significantly superior results. As a result, they also conducted an extensive analysis to determine the optimal architectures and configurations for this particular case. Since then,

although more traditional algorithms have continued to be used, deep learning became the most chosen option by researchers to solve this problem. In addition to the excellent results, the fact that no manual feature selection is required for some deep learning algorithms, like CNN, makes them even more attractive. Along the same lines, in addition to CNNs, models based on the Long Short-Term Memory (LSTM) technique began to be used to a large extent. That is due to the usual treatment of the temporality of the data in HAR. LSTMs are known for being models capable of including information from the past in their training, which is very positive for optimally classifying the data. Nonetheless, a drawback of these methods is their requirement for a substantial amount of data and time to attain appropriate training, which makes them quite different from CNNs. A few instances that demonstrate the application of this technique are represented in [26,40–42], where excellent results were obtained. In fact, in [41,42], a variant of this technique is presented, capable of achieving even better results than in its original form. This variant is called Bidirectional LSTM (Bi-LSTM) and can store data both from the past and the future (assuming that LSTMs usually store it unidirectionally from the past), adding it to the learning during the training of the models that implement it. Compared to its original form, the disadvantage is its higher complexity, as it has to study two time directions instead of one, leading to even longer training times. Given that, work in HAR is currently focused on the use of CNN and LSTM and their variants, in search of the most suitable model, as both techniques yield outstanding results in this field [15,43–45]. In any case, while both techniques are suitable for brief activities such as sitting or hand raising, it seems that, in general, there is a slight bias towards CNN over LSTM, given its speed and ease of implementation [46].

On the other hand, not all the research carried out was based primarily on the accelerometer and gyroscope sensors. Examples such as [47,48] prove the potential of other sensors, such as the magnetometer or GPS, with excellent results when added to their studies. More specifically, these sensors seem to work well with diverse types of long-themed activities, such as walking or running, as shown in these studies.

However, although all those works served to expand the knowledge in HAR, their advances would not be sufficient to be transferred to an application in everyday life. They have obtained their data in very controlled environments, with pretty specific instructions, so it is not feasible to expect the same good results if we transfer the proposed models to real life. While there are some works such as [49,50] that have tackled this problem, there is still a long way to go. In these cases, they achieved good results by transforming the smartphone's coordinate system into the Earth's one. Anyhow, their performance drops when changing the device's orientation device. In addition, they neither take into account the possible different placements of the smartphone, resulting in the same problems as in the other works.

Fortunately, very recent works have been published that seek to fill that gap between the laboratory models and their real-life applications [23,25]. Specifically, in [23], a dataset was published expressly focused on solving this problem, which will be used in this paper. In [25], the same dataset was also used, with an in-depth study of different machine learning algorithms and configurations, with a particularly significant improvement in the initial results. Other recent work using the same dataset, such as [24], is also worth mentioning, with a graph-theory approach based on Random Forest. However, those works still fall in manual feature engineering, oppositely to deep learning which pursues the automation of this part. Given the latest trends and advances made in HAR, algorithms such as CNN or LSTM should result in an optimisation of the final performance. In fact, there is already a paper using such algorithms on that dataset [51], but they do not mention how the data were preprocessed to feed the proposed models. Likewise, they also do not match the percentage of data per class they present in comparison with the original dataset. The class that should have the highest number of samples is presented as one of the classes with the fewest. Finally, they only use the accelerometer from the four original sensors. For all these reasons, it is impossible to reproduce their experiments as the specific conditions under which they were carried out are unknown. Hence, it is not possible to compare it with the present paper. Anyhow, it is considered that their approaches can be vastly improved, following a much more suitable methodology for such a dataset. Therefore, this paper aims to get the best model for the given dataset, in a quest to move towards that highly pursued real-life ideal.

3. Deep learning

Within the field of human activity recognition, the artificial intelligence algorithms used are very diverse. However, inside the deep learning area, two models stand out above the rest: Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM). Therefore, these two were used to compound the proposed models of this work. All their implementation was carried out entirely in Python, using the Tensorflow and Keras libraries [52,53]. Additionally, for the cases that implement LSTM, the cuDNN library [54] was used to take advantage of the speed of the GPUs available to carry out the experiments of this work.

3.1. Convolutional Neural Network

Convolutional Neural Networks (CNN) [55,56] are one of the most widely used models nowadays. Since the gradient modification carried out in [57], they have become a state-of-the-art model to extract information in almost any area of knowledge. These networks consist of a series of layers formed by a set of neurons or filters that receive different pieces of information as input. In this way, each filter is fed with different data from a sliding window or kernel over the initial signal or image. Unlike traditional neural networks, the weights of each of these filters are the same [58]. Therefore, the output ($X^{(l)}$) is the convolution of the input features ($X^{(l-1)}$) with a set of learnable filters ($W^{(l)}$), to which biases ($b^{(l)}$) are added. Finally, an activation function ($g^{(l)}$) is applied. The most commonly used one in HAR research (and the one selected for this paper) is the Rectified Linear Unit (ReLU), which returns

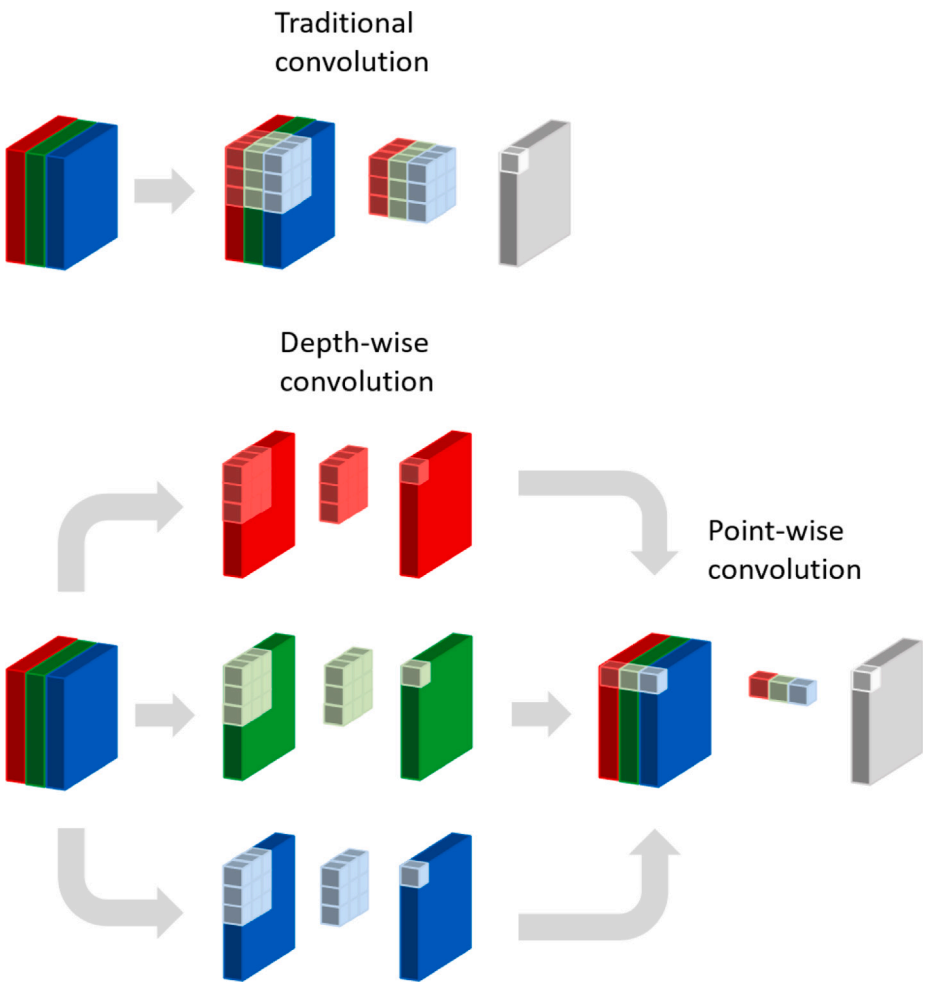


Fig. 1. Comparison of a traditional convolution and its equivalent Depth-wise Separable convolution.

0 when it receives a negative value or the value itself if it is positive. In such a way, the whole process results in the equation below (note that here the symbol "×" reflects a convolution):

$$X^{(l)} = g^{l}(X^{(l-1)} \times W^{(l)} + b^{(l)}) \tag{1}$$

That scheme can be repeated several times, where each layer will extract more features from the information already acquired in previous layers.

Moreover, for this paper, a MaxPooling layer was added after each convolutional layer. Such layers are used to down-sample the spatial dimensions of the input data, retaining the most relevant features. To do that, they divide the input data into a set of non-overlapping rectangular regions, outputting the maximum value of each one. In this case, these regions were implemented with a size of 2, as seen in many other works in HAR [59,60]. That makes the resulting models more robust to possible changes or distortions in the data and reduces the computational time required to train them [61].

Once the features have been extracted from the input matrix and transferred through each layer, they are fed into a fully connected perceptron (Dense layer). As for the final prediction and the probability vector $p_t = [p_{t_1}, p_{t_2}, \ldots, p_{t_k}] \in \mathbb{R}^k$, the softmax function was used, which converts the input values into a probability distribution, with values between 0 and 1. These input values would be the output values of the previously mentioned perceptron (z), giving rise to the following operation:

$$p_{t_i} = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{2}$$

Then, the results obtained would be returned directly, selecting the label with the highest probability after the softmax.

However, for this paper, it was opted to use the Depth-wise Separable Convolutional Neural Networks (DS-CNN) variant [62]. The choice of this variant is mainly due to its higher speed and efficiency compared to its original form. That is particularly appealing considering the large number of patterns to be used in this work. Moreover, it is starting to be applied in the most recent HAR studies,

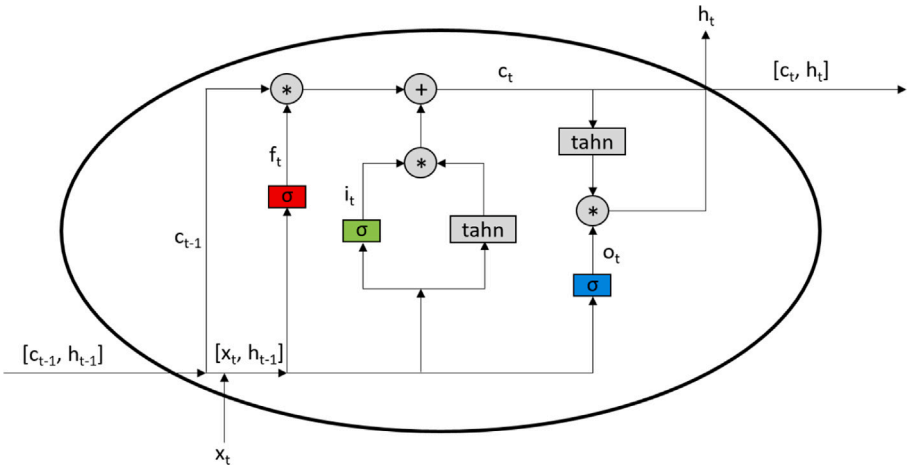


Fig. 2. Example of a LSTM unit, as shown in [40] (weight matrices and bias not displayed).

with excellent results [63]. Nevertheless, this modification is known for drastically reducing the requirements by significantly cutting down the number of necessary parameters [64]. To do that, the kernel is applied separately on each of the available channels of the input signal, rather than on all of them at once. This convolution would work the same way as the traditional one but using fewer features in each case. Then, the information obtained for each channel is combined through another convolution, projecting the resulting data onto a new feature map. The difference here is that the latter is carried out as a point-wise convolution (i.e. 1×1 convolution). As shown in Fig. 1, this ensues in fewer operations by integrating the data from the different channels. In this way, the computations are done with much less data and an equivalent outcome to traditional CNNs.

3.2. Long Short-Term Memory

Unlike their precursor, the Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) networks [65] are a type of system capable of selectively remembering or forgetting data. To this end, they perform a series of slight modifications to the data they use, based on so-called cell states. For ease of understanding, an example of an LSTM unit is shown in Fig. 2. As can be seen, the typical LSTM network consists of a series of memory blocks called cells, between which two different states are transferred: the cell state (c) and the hidden state (h). In order for these blocks to be able to remember data, they implement a structure consisting of three different gates, as detailed below:

1. Forget Gate (the red one in Fig. 2). It removes all information that is no longer relevant for learning. To do that, the input data of the current time (x_t) and the hidden state of the previous cell (h_{t-1}) are multiplied by their correspondent weight matrix (W). Also, a bias (b) is added to the operation to get a better fit of the data. That constructs a regulatory filter, which is represented by the resulting sigmoidal function σ that follows:

$$f_t = \sigma\left(W_{xf} \times x_t + W_{hf} \times h_{t-1} + b_f\right) \tag{3}$$

   That would result in a value between 0 and 1. When multiplied by the cell state, it decides whether that information should be continued or not.
2. Input Gate (the green one in Fig. 2). It is responsible for adding relevant information to the model and filtering out any that may be redundant. To this end, another sigmoidal function is constructed, multiplied by a hyperbolic tangent one ($tanh$) that outputs the data between -1 and 1. In this way, the tanh function decides which data can be added later to the model, using a sum operation with the information of the forget gate. These functions are represented as follows:

$$i_t = \sigma\left(W_{xi} \times x_t + W_{hi} \times h_{t-1} + b_i\right) \tag{4}$$

$$c_t' = tanh\left(W_{hc} \times h_{t-1} + W_{xc} \times x_t + b_c\right) \tag{5}$$

3. Output Gate (the blue one in Fig. 2). This gate decides which outcome to keep, regarding that not all information flowing through the cell state may be adequate. In much the same way as before, sigmoidal and hyperbolic tangent functions are multiplied to filter these data. These functions are shown below:

$$o_t = \sigma\left(W_{xo} \times x_t + W_{ho} \times h_{t-1} + b_o\right) \tag{6}$$

$$c_t'' = tanh(c_t) \tag{7}$$

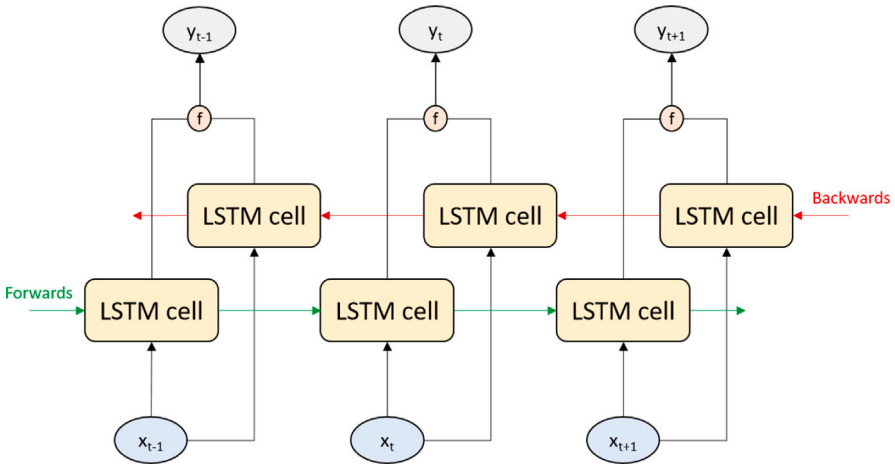


Fig. 3. Example of a Bi-LSTM network.

In this way, new cell and hidden states are obtained. Then, they are transferred to the next unit, repeating the process discussed above. These states are calculated as follows:

$$c_t = f_t \times c_{t-1} + i_t \times c'_t \tag{8}$$

$$h_t = o_t \times c''_t \tag{9}$$

As for the prediction and the probability vector $p_t = [p_{t_1}, p_{t_2}, \ldots, p_{t_k}] \in \mathbb{R}^k$, these are calculated from the resulting hidden state (h_t). This forms a softmax function (s), already commented in 3.1, which results in the following equation:

$$p_t = s(W_{hk} \times h_t + b_k) \tag{10}$$

Finally, the class label k_t is assigned to the one with the highest value in the vector of probabilities.

In the present work, in addition to traditional LSTMs, their bidirectional variant (Bi-LSTMs) was also used. This modification was formerly presented for the predecessor RNNs [66], but it can be used in the same way in a multitude of networks. The difference that characterises this variant is that it makes networks capable of storing data in both directions, usually by adding the future case (assuming that LSTMs usually store data unidirectionally from the past). This peculiarity, coupled with the fact that they are recently being used in the field with high-quality results, makes them a pretty attractive option for this work. In order to carry out this modification, two different LSTM models are trained, one that explores the input data (x) backwards and one that does the same but forwards, as shown in the example Bi-LSTM network in Fig. 3. During each model training, in each time step, a merging stage (f) is performed to mix the outputs obtained. That step can be carried out in different ways, but the most common and the one that was implemented in this work will be that of concatenation. In such a way, the output (y) of the first model is concatenated with the second model's. That ensures the latter can allow for both signal directions in the following time steps.

3.2.1. Hybrid models

A hybrid model refers to a model that combines different types of machine learning or deep learning algorithms. One of the most prominent examples in the HAR field is the combination of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) models. That is due to the peculiarities of each of them. On the one hand, CNNs try to reflect the spatial features of the data introduced into them. On the other hand, LSTM models look for these elements in the temporal section of the data that feeds them. Therefore, if the aim is to classify data with different signal distributions and time intervals, these algorithms combined could significantly improve performance.

Thus, in this work, those algorithms were combined using the variants discussed in the previous sections: Depth-wise Separable Convolutional Neural Networks (DS-CNN) and Bidirectional Long Short-Term Memory (Bi-LSTM) models. That led to the following hybrid models: (DS-CNN)-LSTM and (DS-CNN)-(Bi-LSTM). Hence, the spatial features extracted by the CNNs can be further exploited by the LSTMs, merging them with the temporal characteristics that can be derived by the latter. That should result in a substantial improvement in the final performance, although the corresponding execution times also increase with higher model complexity.

The way these models are assembled differs slightly from their individual cases. For this paper, the DS-CNN layers were always applied first, before the LSTM-based ones, to properly exploit the features as discussed in the previous paragraph. Thus, in hybrid models with more than one layer, a (DS-CNN)-(DS-CNN)-LSTM-LSTM style structure will be followed, without interleaving independent models. With this architecture, the outputs of the last MaxPool performed in the DS-CNNs will be the inputs of the LSTM-based models. Similarly, the outputs of the uttermost LSTM will be the outputs of the full hybrid model.

4. Methodology

This section explains in detail all the techniques and resources employed in this work. Firstly, Section 4.1 discusses how the data was processed and prepared for input into the subsequent artificial intelligence models. Then, Section 4.2 presents the different evaluation metrics used in this work. After that, Section 4.3 outlines various techniques to validate and improve the generalisation of the resulting models. Finally, Section 4.4 introduces the proposed models' architecture and configurations.

4.1. Data preparation

As previously mentioned, the dataset presented in [23] was used to carry out this work. Here, it is also worth noting that the data collection aimed to include individuals with diverse characteristics, encompassing physical diversity, smartphone usage patterns, and device models. Consequently, the study involved 19 participants, aged approximately 25 to 50 years, to ensure a wide range of behavioural patterns contributing to the development of future models. However, gender diversity is limited, with only two women among the participants. Nevertheless, participants' physical characteristics, habits, and preferences regarding smartphone use and positioning display considerable variation. Hence, while there is potential for improvement in variability, significant diversity remains present. As for the sensors used, there were four: accelerometer, gyroscope, magnetometer and GPS. Nonetheless, what makes the dataset most remarkable is that the individuals who took part in the data gathering were given almost total freedom, only having to use a custom Android app to start or stop the concerned activity. These activities were the following, as discussed in that work:

- Inactive: not carrying the smartphone on you at all.
- Active: any activity with movement, but without moving to a specific point in time. That would include activities such as: giving a lecture, cleaning the house or being at a concert.
- Walking: any trip made on foot, whether it is a regular walk or a jog.
- Driving: all journeys made via motorised transportation, without requiring the traveller to be the driver.

Concerning data preprocessing, almost the same dynamics as in the original work were followed, carrying out the following operations:

- Every outlier found in the GPS data was removed. That is the measurements that surpassed 0.2 decimal degrees on latitude and longitude increments between observations or 500 m in the case of altitude. Given its sampling rate, these measures seem unreal to accomplish for any living being.
- The first and last five seconds of each session were eliminated to avoid confusion during the training of the deep learning models. These time intervals correspond to the stages in which individuals picked up or put away the smartphone at the start or end of the action. Therefore, they were not relevant to the activity in question. Note that each session corresponds to an independent data gathering, from the moment when an individual begins an action until they finish it.
- The GPS data are largely sparse in each session, mainly because of the long waiting time between observations (>10 s). In the original paper, if there is more than one second between samples, the first one is replicated second by second, with a different timestamp, until this time difference does not prevail, in both directions. However, in the present work, since the sliding windows will move 10 s at a time, such replication was done every ten seconds instead of only one. Anyhow, every session without any GPS observations was discarded.
- Any session with substantial time gaps without observations (>5 s) was considered corrupt and, therefore, was ignored. Note that this does not include GPS data.

To prove the importance of this preprocessing, an example of how the different sensors behave in each of the specified activities can be seen in Fig. 4. Those examples correspond to the first 15 s of different sessions taken by one specific individual in the study. The selection of this time interval is due to the fact that it allows each activity and sensor behaviour to be illustrated easily on a single figure. Note that each subfigure displayed there corresponds to data taken while performing one of the four studied activities: inactive (*a*), active (*b*), walking (*c*) and driving (*d*). Also, to represent all the values on the same scale, the values corresponding to the GPS were divided by 10. Similarly, the values for the magnetometer and accelerometer were also divided by a value of 5 and 2, respectively. In this way, it is possible to easily observe the changes occurring in each sensor, for each specified action. For the accelerometer, gyroscope and magnetometer, those data are displayed for each of its three axes: *Acc_x*, *Acc_y* and *Acc_z*, *Gyro_x*, *Gyro_y* and *Gyro_z* and *Magn_x*, *Magn_y* and *Magn_z*, respectively. As can be seen in each subfigure, there are evident irregularities for the first seconds of each session, which only add noise to their interpretation. Concerning GPS, the increments in latitude, longitude and altitude from the last observation are displayed (*GPS_lat*, *GPS_long* and *GPS_alt*), together with speed, bearing and accuracy of the current measurement (*GPS_sp*, *GPS_bear* and *GPS_acc*). For this particular case, it is possible to see the small number of observations recorded, compared to the other sensors ones represented, highlighting the need to replicate them. Anyhow, distinct patterns can be discerned for each sensor, contingent on the executed activity. However, it is worth noting that, although considering those evident differences, this may not be the case with other types of actions. After all, although four activities are being studied, all of them encompass a variety of diverse actions, except for the "inactive" activity. For instance, washing dishes or teaching a class could exhibit significantly different signals despite both falling under the "active" activity category. Similarly, the same could apply to "walking" and "driving" activities. In the former case, differences might arise within the same activity if the session involves going

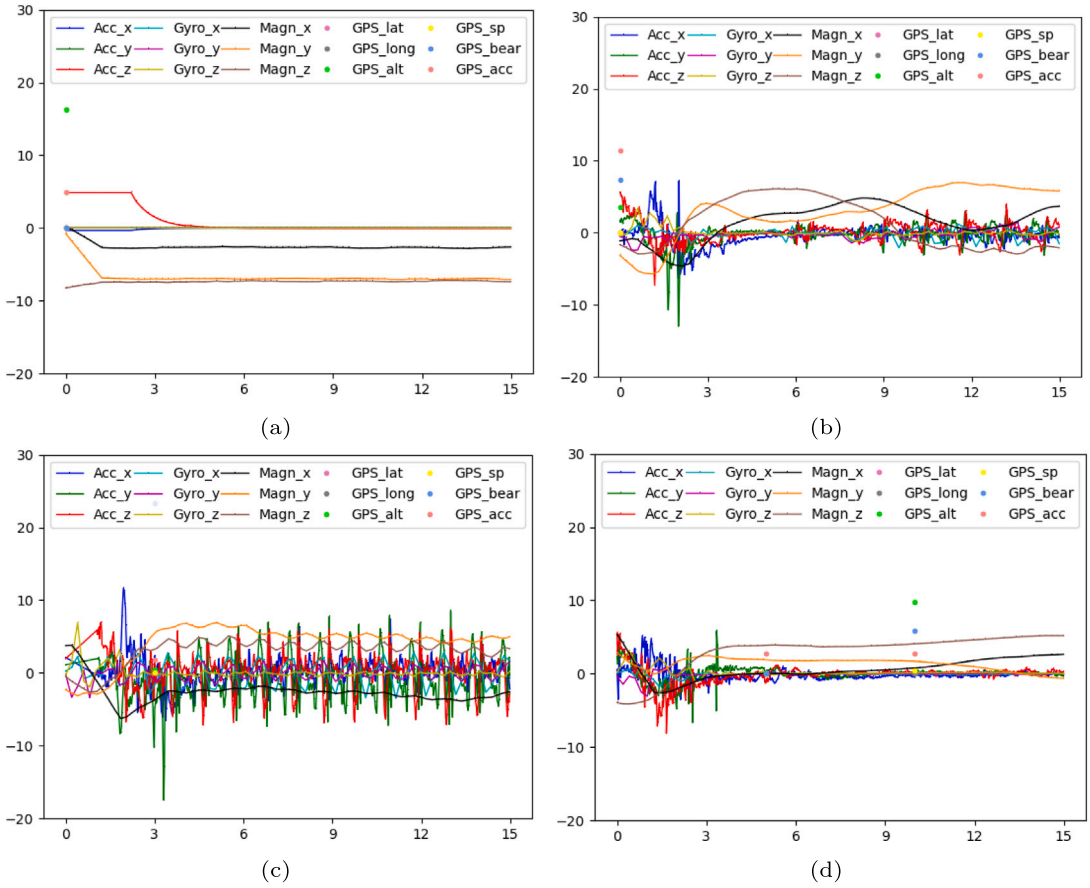


Fig. 4. Raw (scaled) data captured by a single individual's smartphone sensors, within the first 15 s of various example sessions, for each designated activity, being: (a) Inactive activity. (b) Active activity. (c) Walking activity. (d) Driving activity.

Table 2
Sensor's average sampling frequency and their respective standard deviation values for each activity measured (in Hz), after data preprocessing.

	Activity			
	Inactive	Active	Walking	Driving
Accelerometer	9.51	32.30	28.29	37.04
	±13.92	±23.49	±24.29	±20.94
Gyroscope	4.67	4.45	6.34	4.70
	±0.72	±1.45	±12.13	±2.62
Magnetometer	7.66	8.23	6.41	7.00
	±11.28	±12.38	±8.66	±9.94
GPS	0.01	0.03	0.07	0.13
	±0.09	±0.16	±0.26	±0.34

for a walk or jogging. As for the latter, substantial differences could appear depending on whether the individual is driving their own car or taking public transportation. In this way, although it is believed that these trends could also be present in other data sessions, they should be approached with a measure of caution.

Following prior data preprocessing, it was decided to apply 30, 60 and 90-s sliding windows, with an overlap of 20, 50 and 80 s, respectively (moving the window 10 s at a time). The selection of these time intervals and no others is due to the performances observed in other works using the same dataset, such as [25]. It is considered that, with this selection, it is possible to see the general behaviour of the proposed models to see if there is any trend in the results towards larger or smaller window sizes. Furthermore, it can be considered a reasonable amount of time given the long-themed nature of the activities included in this dataset, without being too broad or limited. In fact, if it were, it would not be possible to separate and identify the actions correctly, and there could be periods of inactivity in an activity that is supposed to be entirely associated with "walking", for example, by going for a random walk and stopping at a traffic light.

Table 3
Number of available patterns and their distribution among the above-mentioned activities, for an overlap of 10 s less than each full window size.

Window size	Activity				
	Inactive	Active	Walking	Driving	Overall
30	21,152	13,778	7823	6109	48,862
	43%	28%	16%	13%	
60	20,836	13,487	7204	5716	47,243
	44%	29%	15%	12%	
90	20,486	13,223	6732	5439	45,880
	44%	29%	15%	12%	

However, in order to feed the prepared data into the deep learning models, it was necessary to perform another series of operations. With the considerable differences observed in the frequency of each sensor for each activity studied, it is unattainable to transfer the data directly to the model. Table 2 shows the values of this frequency for each sensor and activity performed after preparing the data as detailed above. As can be seen, there are very abrupt cases, especially with the accelerometer, since it drastically changes its sampling frequency when any movement or vibration is detected. For that reason, for example, the frequency is much higher for the case of "walking" compared to "inactive". Likewise, there is a substantial change in the gyroscope's frequency for the "walking" activity, compared to the rest. Anyhow, the differences are slighter than with the accelerometer, as it focuses on changes in the smartphone's orientation and not on any movement or vibration that may exist. As for the magnetometer and GPS, their variations are more arbitrary, although there is a tendency to get more GPS measurements as the travelling speed increases. In addition, each smartphone may have slight differences for the same observation [19], which may also affect this sampling rate. In fact, some of these differences were thought to be mainly due to the behaviour of some sensors in moments of high or low movement [23].

Nonetheless, upon further exploration, those changes seem to be also somewhat arbitrary. Indeed, a peculiar behaviour was observed for the accelerometer, gyroscope and magnetometer. The data provided by these sensors are generally given either every 20 ms or every 200 ms. In the few cases in which this is not the case, it is by a slight difference, with a frequency closer to 10 ms, or approximately 180 ms, depending on the case. Moreover, these differences do not seem to correspond to any specific individual or activity, as they may be noted even during the same data collection session from a specific one. Therefore, the hypothesis is, apart from the movements and vibrations commented on before, that there could be some settings on the individuals' smartphones affecting these sampling rates. For example, the trigger of automatic battery saving when reaching a certain threshold, even if all permissions were activated for the data collection Android app used for such work.

For all those reasons, it was necessary to transform the data so that each sensor had the same sampling rate across all associated observations. For that purpose, one possibility could be to apply linear interpolations, as they were the most commonly used operation in the field when the context required it [15,67,68]. Only in [19] was some exploration with other more complex interpolations like the quadratic or cubic ones, but without an in-depth investigation. Regarding the values of this sampling rate, no clear consensus has been found in the scientific community for smartphone sensors. Some researchers say that around 2–3 Hz is the most appropriate [69]. Others prefer to set it between 0–15 Hz [11], or even up to 50 Hz in some situations [70]. Anyhow, they all were studies carried out in controlled laboratory environments and without handling sensors with frequencies as different as GPS's. Therefore, given the distinct and scarce approaches in the HAR literature, an experimental solution was chosen. The present work deals with a singular case in which sampling rates stabilise around a value every 20 ms or 200 ms (50 Hz and 5 Hz, respectively). Given that, it was considered that the most appropriate approach would be to fix this frequency at 5 Hz, always selecting the closest real value to each time instant, every 200 ms. In this way, the observations to be introduced later in the proposed models would be completely real, without the modification they could suffer when going through a traditional interpolation. In addition, the temporal error that could be accumulated for cases that do not strictly follow these dynamics would be small, given the little and unusual changes that occur at these frequencies. Thus, when data are given every 20 ms, only those observations that correspond to the time instants that occur every 200 ms would be selected, ignoring the rest. When the data are given every 200 ms, it would be only necessary to pick those observations that correspond to each 200 ms advance in time. Although it is true that with this approach a considerable amount of existing patterns from the original dataset are eliminated, it is considered the most appropriate choice for the problem to be solved, given the circumstances. Concerning GPS, although it is not particularly affected by that problem, it does have a very high and irregular frequency, compared to the rest of the sensors used. Hence, following the same idea as before, it was decided to set the frequency at 0.1 Hz (one value every 10 s). That is possible thanks to the replication carried out before, detailed at the beginning of this section.

After carrying out all the steps discussed above, the total number of patterns is shown in Table 3 for each proposed window size. As can be seen, there is a clear imbalance towards the "inactive" activity, probably due to the ease of collecting this type of data compared to the rest. Even so, it is considered that the overall number of patterns in each class is sufficient to perform a satisfactory classification, as seen already in other works using the same dataset [23,25].

4.2. Evaluation metrics

In order to easily view and evaluate each model classification, the most widely used option by the scientific community is the confusion matrix. From this matrix, many metrics can be extracted. Since this paper deals with a multi-class case, a *one-versus-all*

strategy was followed to reduce them to a binary type. Thus, each class is analysed separately comparing it with the rest together. From there, some of the most elementary metrics that can be extracted are the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), for a given class. In addition, from these values, other representative metrics can also be calculated, such as precision, recall, accuracy, and F_1-score [71]. Among those metrics, the most commonly used one to measure the performance of a test model is accuracy. For its calculus, the percentage of correctly identified cases out of the total is measured using the following formula:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{11}$$

However, there are cases where the latter metric can lead to some bias, especially on an imbalanced dataset. For this very reason, when there is a considerable imbalance in the data, the F_1-score [72] metric is usually also shown. To measure it, precision and recall are combined in a harmonic mean, with precision being the ratio of the TP to all cases labelled as positive by the model (TP + FP), while recall refers to the division of the same TP by the total number of positives in the ground truth (TP + FN). Given that, its formula would be as follows:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{12}$$

For this paper, the results will always be evaluated based on the accuracy metric to be able to compare it properly with the rest of the works carried out on the same dataset. Anyhow, given the clear imbalance present in the data, the F_1-score will also be shown for the most representative cases to get a closer look at the actual performance of the models. Since this paper presents a multi-class problem, an averaging process is necessary to get the overall value of this metric. Given that, the *macro* strategy, which returns the mean value obtained by computing the metric for each label individually, was followed.

4.3. Validation techniques

Cross-validation [73] is regarded as one of the most reliable methods for validation. Before data is fed directly into the model, it undergoes a process of division into training and testing. In such a way, the model will be able to use one subset of the data only for training and another for testing, the latter a priori unknown to the model. The most prevalent way to carry out this division is through *k-fold cross-validation*. This technique consists of partitioning the original dataset into a number *k* of subsets of equal size. One of these partitions will form the test set of the model, while the rest will be used for training. Then, the procedure will be reiterated *k* times, ensuring that each subset has been designated as a test set once. Finally, after feeding the models with each partition, the outcomes are averaged, and the pertinent metrics are computed. In such a manner, the random component of splitting the original set in training and testing only once, which could lead to unreliable results, is largely avoided. For this paper, a modified version of this approach, known as *stratified k-fold cross-validation*, was employed. This alternative aims to ensure the same percentages of class representativeness in all the partitions carried out. Hence, it can mitigate the influences of the present imbalance in the initial dataset. All things considered, in this work, a stratified 10-fold was applied, splitting the data into training and test, with a distribution of 90%, and 10% for each subset formed, respectively.

Nonetheless, with that approach, one of the most common issues in any work related to machine learning may arise. That is the overfitting problem [74,75]. A model is said to be overfitting a dataset when instead of extracting information from patterns, it mainly memorises them. This problem is even bigger in deep learning because of the increase in the number of weight drives, which considerably expands the memorising capacity of the network. To alleviate this issue as much as possible, for each training set, 11.11% of the data included therein has been assigned to a validation set. Thus, the general distribution for each fold would be 80% training, 10% validation and 10% test. In this way, during training phase, the model's performance is tested against the validation set. That yields a loss value that evaluates the classification at that point, based on the sum of the errors obtained for each sample. The lower the value, the better the classification, a priori. However, this value may reach a point where the improvements are almost imperceptible, leading the model to a clear case of overfitting. To avoid that, an early-stopping function was applied to each model [76]. This kind of function seeks to interrupt training when that point is reached, returning the best weights obtained by the model so far. For this work, training will be interrupted when the model has not improved the last best loss value for 20 iterations (out of a fixed total of 100 iterations). In this way, although it is impossible to guarantee that training does not stop at a local minimum, the generalisation capacity of the model may be improved.

In addition to the above, a Dropout layer was also set up for each proposed model. This layer dumps part of the outputs, forcing the model to rely on other connections. In this way, the generalisability of the model increases considerably. This layer was applied for 50% of the input units and placed just before the final output. Both the quantity and the placement selected for these layers are those commonly used in the literature [77].

4.4. Proposed approach

As discussed before, the algorithms selected for this work were Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) models, given their outstanding results in the field. Specifically, for the CNN case, Depth-wise Separable Convolutional Neural Networks (DS-CNN) were used, to speed up the experiments without affecting performance. On the other hand, in the case of LSTM, its bidirectional variant (Bi-LSTM) was also used, as it was considered that it could provide good results for the problem to be solved, given its recent applications.

Table 4
Training hyperparameters.

Hyperparameter	Value
Batch size	32
Layers	[1, 2]
Neurons (or CNN filters)	[16, 32, 64]
Kernel (CNN only)	[3, 5, 7]
Padding (CNN only)	Same
Activation	ReLU
Optimiser	ADAM
Loss	Cross-entropy
Iterations	≤100
Early stopping	20

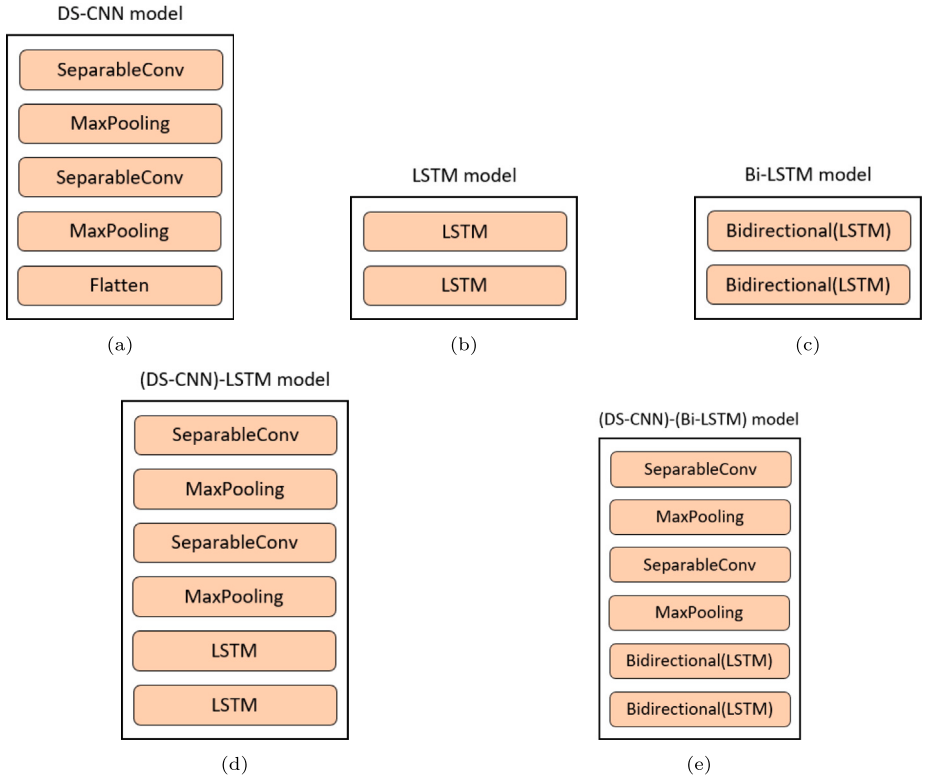


Fig. 5. Implementation of each individual algorithm used, for the case of having a number of layers equal to two, being: (a) DS-CNN model. (b) LSTM model. (c) Bi-LSTM model. (d) (DS-CNN)-LSTM model. (e) (DS-CNN)-(Bi-LSTM) model.

Nonetheless, given the prominent use of those algorithms simultaneously in the literature, it was also decided to do the same for this paper. In addition to using those algorithms individually, they were also combined, resulting in (DS-CNN)-LSTM and (DS-CNN)-(Bi-LSTM) hybrid models, as discussed in Section 3.2.1. In this way, it is possible to make a comparison between all proposed models, observing in detail the advantages and disadvantages of each one.

Concerning the hyperparameters used for each of those models, they are shown in Table 4. The batch size was set to 32. After a few preliminary explorations with higher values (64, 128, 256, 512 and even 1024), this was the best trade-off between efficiency and accuracy. Consequently, considering that the changes in classification accuracy were negligible between 32 and the rest, it was decided to discard them. Regarding the rest of the hyperparameters, note that in the hybrid models, a layer number of one would correspond to a total of two layers (one per individual network). Likewise, two layers would result in a total of four layers. For example, when we join a DS-CNN with an LSTM with a layer number of 2, it would look like this: (DS-CNN)-(DS-CNN)-LSTM-LSTM. That is two layers for DS-CNN and two for LSTM, with the DS-CNN layers always going before the LSTM ones, as said in Section 3.2.1. Therefore, it was not considered to explore with more layers, as this would remarkably increase the complexity of the models. As for the neurons, a similar combination as in [59] was used, but without going that further. Concerning kernel size, once again, the variety used in [59], with fewer options, was the one applied. In addition, the padding of the DS-CNNs was set to "same" to be able to perform convolutions of the desired size. This parameter allows the algorithm to fill with zeros evenly around the signal,

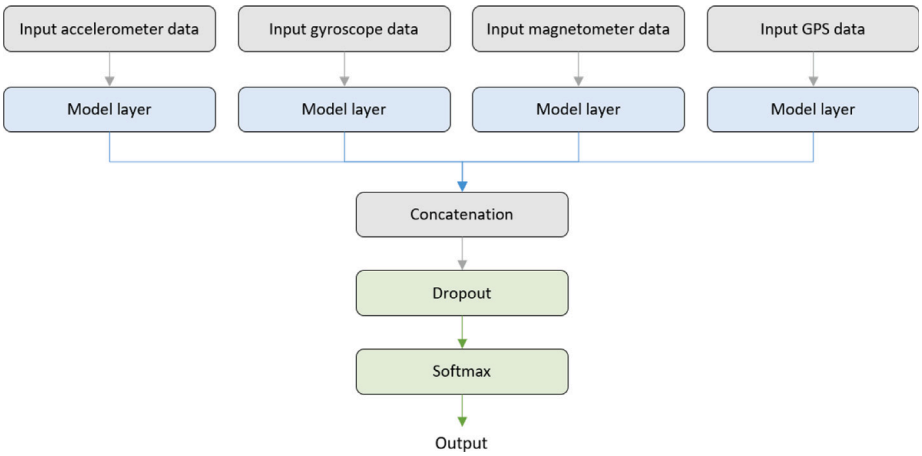


Fig. 6. General architecture of the whole model used to carry out the experiments.

allowing the input dimensions to match the output dimensions. As for the activation, optimiser and loss functions, the most widely used in the literature were the ones applied: Rectified Linear Unit (ReLU), Adaptive Moment Estimation (ADAM) and cross-entropy, respectively. Finally, a number of 100 iterations was set, with an early stop of 20 if the validation loss did not improve. Thus, the model gets enough time to recognise the patterns while avoiding overfitting. Any other parameters that might be present were kept by default.

With respect to the architecture followed to implement those models, you may see the implementation followed for each algorithm individually in Fig. 5. Note that, in the case represented there, all models are shown with two layers to visualise the most complex versions of each kind. There, each subfigure shows a different algorithm, from top (input layer) to bottom (output layer). In the case of DS-CNN (*a*), each model is formed by a convolutional layer (SeparableConv), followed by a MaxPooling layer, as seen in the rest of CNN works in HAR. Also, in the end, regardless of the previous number of layers, a Flatten layer is added. That is to format the resulting feature map to allow for being consumed by subsequent layers. To do that, the input data is converted into a one-dimensional array. Concerning LSTMs (*b*), they are formed only by the layer that implements this algorithm, in this case following cuDNN's implementation to accelerate its training time. For its bidirectional variant (*c*), this layer is enclosed by the wrapper that adds this functionality (Bidirectional()). Finally, for the hybrid models, (DS-CNN)-LSTM (*d*) and (DS-CNN)-(Bi-LSTM) (*e*), the previous cases are combined. In this way, the layers corresponding to DS-CNN are added first, which will subsequently feed those based on LSTM. As the resulting DS-CNN feature maps can already be consumed directly by LSTM, it is not necessary to add a Flatten layer for these hybrid cases.

In such wise, the entire final model consists of four different inputs, one for each sensor used in the dataset, as shown in Fig. 6. In this manner, it is possible to use each sensor's data, while avoiding dealing with the peculiarities of each one by focusing on one particular sensor at a time. Thus, the data measured by each sensor is transferred through the CNN and LSTM networks, as appropriate. As for the model layers represented there, the same one is always applied for each of the four branches. For example, if a DS-CNN model is used for the accelerometer data, the same is implemented for the rest of the sensors. Note that each of those networks would correspond to the models shown in Fig. 5. That results in different outputs for each sensor, depending on the particularities encountered in each case. Then, in order to be able to combine everything in the same model, the outputs of each of these branches are concatenated in a single layer. After that, to avoid overfitting and increase the generalisation capacity of the implemented models, a Dropout layer was added, affecting 50% of the input units, as commented in Section 4.3. Finally, a Softmax layer was set to obtain the desired output with the four activities to be studied.

5. Results and discussion

This section shows all the results obtained from the proposed experiments. On the one hand, Section 5.1 indicates the performance of each experiment, with its corresponding outcomes. Then, Section 5.2 introduces a series of comments and observations on the obtained results.

5.1. Results

With the models discussed in Section 4.4, the results shown in Tables 5, 6 and 7 were obtained. There, the values corresponding to the average accuracy of every possible combination of hyperparameters are represented, with their standard deviation below. Those hyperparameter combinations correspond to the number of layers (L), the number of neurons in each layer (N) and the kernel size (K). Moreover, each table corresponds to a specific sliding window size: 30, 60 and 90 s, respectively, as pointed out previously.

Table 5
Accuracy results obtained detailed for a window size of 30 s.

		N = 16		N = 32		N = 64	
		L = 1	L = 2	L = 1	L = 2	L = 1	L = 2
DS-CNN	K = 3	89.21% ±8.17%	88.02% ±10.37%	87.95% ±9.26%	88.30% ±9.66%	89.10% ±7.06%	89.15% ±8.95%
	K = 5	88.64% ±7.80%	88.39% ±9.22%	87.59% ±8.66%	88.68% ±8.95%	88.73% ±7.85%	88.08% ±9.54%
	K = 7	88.33% ±8.01%	87.95% ±8.70%	89.23% ±7.36%	88.57% ±9.52%	88.00% ±8.68%	87.50% ±11.53%
LSTM		90.99% ±6.99%	91.07% ±5.88%	**93.15% ± 4.52%**	91.49% ±5.58%	91.91% ±6.63%	90.06% ±7.36%
Bi-LSTM		91.91% ±4.76%	90.33% ±8.21%	89.46% ±8.72%	90.99% ±6.73%	91.22% ±5.86%	90.09% ±8.85%
DS-CNN-LSTM	K = 3	92.15% ±7.09%	90.47% ±7.71%	91.15% ±7.54%	90.38% ±8.91%	91.85% ±6.34%	91.20% ±7.88%
	K = 5	91.58% ±7.22%	90.04% ±8.2%	91.07% ±7.79%	88.96% ±10.05%	91.64% ±6.73%	92.46% ±5.61%
	K = 7	91.75% ±5.62%	90.84% ±7.04%	91.76% ±6.21%	91.23% ±6.89%	90.36% ±8.39%	89.05% ±9.43%
DS-CNN-Bi-LSTM	K = 3	91.88% ±7.24%	90.40% ±7.29%	91.64% ±7.10%	90.47% ±8.26%	91.49% ±7.26%	90.64% ±8.70%
	K = 5	92.56% ±5.84%	90.31% ±9.21%	90.92% ±7.20%	89.87% ±8.20%	91.77% ±6.94%	91.24% ±8.48%
	K = 7	90.63% ±8.16%	91.63% ±6.48%	90.99% ±7.47%	90.45% ±7.35%	91.32% ±7.42%	90.51% ±8.02%

Table 6
Accuracy results obtained detailed for a window size of 60 s.

		N = 16		N = 32		N = 64	
		L = 1	L = 2	L = 1	L = 2	L = 1	L = 2
DS-CNN	K = 3	89.64% ±7.67%	89.38% ±9.00%	88.35% ±8.74%	89.86% ±9.51%	89.13% ±8.00%	88.74% ±9.39%
	K = 5	89.18% ±7.99%	89.62% ±8.72%	89.38% ±8.00%	88.94% ±9.57%	88.31% ±8.88%	89.98% ±8.74%
	K = 7	89.64% ±8.25%	89.10% ±10.22%	88.31% ±8.77%	89.80% ±8.56%	88.29% ±8.38%	89.38% ±9.08%
LSTM		91.14% ±8.36%	92.89% ±5.07%	92.15% ±7.26%	93.11% ±6.50%	92.39% ±7.21%	92.99% ±5.04%
Bi-LSTM		91.70% ±8.02%	92.02% ±7.22%	92.75% ±6.58%	92.17% ±7.07%	92.61% ±6.04%	91.89% ±7.30%
DS-CNN-LSTM	K = 3	92.64% ±6.42%	92.71% ±6.87%	93.14% ±6.71%	92.83% ±6.04%	93.09% ±6.14%	92.89% ±5.97%
	K = 5	93.04% ±5.65%	91.55% ±8.37%	93.01% ±7.30%	91.57% ±9.27%	92.55% ±6.47%	91.72% ±7.66%
	K = 7	**93.49% ± 5.34%**	92.09% ±7.71%	93.24% ±6.39%	92.33% ±7.67%	92.42% ±9.11%	92.54% ±7.08%
DS-CNN-Bi-LSTM	K = 3	92.63% ±7.25%	91.68% ±7.50%	92.81% ±6.95%	91.66% ±8.16%	91.64% ±9.05%	90.80% ±8.98%
	K = 5	92.37% ±7.17%	92.32% ±7.08%	92.69% ±5.72%	91.91% ±6.57%	92.38% ±7.67%	91.37% ±8.09%
	K = 7	92.93% ±7.11%	92.48% ±6.57%	92.69% ±7.59%	90.95% ±8.73%	91.89% ±7.18%	90.39% ±8.99%

As can be seen, the accuracies obtained, in general, are higher than those obtained in other works on the same dataset [23,25]. Given the results, the use of deep learning algorithms can be considered one of the best options to exploit such data, especially those based on CNN and LSTM, as proved in recent works in the literature. As for the performance of the models depending on the particular algorithm selected and a specific set of hyperparameters, there do not seem to be very noticeable differences. Their results are objectively constant for each algorithm in the three tables. However, some contrasts are worth noting. Firstly, there are some differences if we look at the values obtained by each algorithm independently. In general, the best results are obtained by the hybrid algorithms mentioned above, closely followed by those based on LSTM, but worsening slightly when only DS-CNN is used. Considering these results, for the dataset used, it appears that the LSTM-based algorithms perform better than the CNN-based algorithms. That seems to indicate that the time component of the signals is more important than the features themselves. To validate these differences, a Tukey test was performed between each group of results, for each algorithm and window size indicated. The

Table 7
Accuracy results obtained detailed for a window size of 90 s.

		N = 16		N = 32		N = 64	
		L = 1	L = 2	L = 1	L = 2	L = 1	L = 2
DS-CNN	K = 3	90.27% ±7.74%	90.32% ±8.54%	89.73% ±7.99%	90.31% ±8.48%	89.76% ±8.09%	89.33% ±9.88%
	K = 5	89.54% ±7.99%	89.51% ±10.5%	89.65% ±8.54%	89.27% ±9.47%	89.28% ±10.09%	89.78% ±9.56%
	K = 7	90.70% ±7.29%	89.25% ±9.40%	88.43% ±10.38%	89.50% ±9.38%	89.89% ±8.17%	89.17% ±8.86%
LSTM		91.88% ±7.14%	91.92% ±6.81%	91.69% ±8.82%	93.52% ±5.59%	92.15% ±7.07%	91.28% ±8.55%
Bi-LSTM		93.09% ±5.10%	92.35% ±6.62%	91.60% ±8.25%	92.12% ±8.23%	91.82% ±8.07%	92.47% ±8.67%
DS-CNN-LSTM	K = 3	93.20% ±5.31%	92.65% ±8.44%	93.60% ±5.96%	92.68% ±7.84%	93.34% ±5.67%	92.77% ±8.30%
	K = 5	93.57% ±5.14%	92.86% ±8.10%	94.78% ±4.64%	92.62% ±9.16%	92.93% ±6.91%	91.48% ±9.28%
	K = 7	93.62% ±7.10%	91.56% ±10.29%	94.19% ±5.72%	92.72% ±7.80%	**94.80%** **±4.09%**	90.88% ±9.79%
DS-CNN- Bi-LSTM	K = 3	93.68% ±6.63%	93.37% ±7.39%	92.72% ±7.36%	91.26% ±8.81%	93.98% ±5.31%	90.73% ±9.33%
	K = 5	92.93% ±7.07%	93.10% ±6.59%	92.05% ±8.90%	93.32% ±7.50%	92.80% ±7.55%	93.04% ±6.71%
	K = 7	92.80% ±8.09%	94.16% ±5.06%	93.51% ±6.28%	93.37% ±7.68%	93.26% ±6.67%	92.58% ±7.99%

Table 8
Overall accuracy results obtained for each window size.

	Window size		
	30	60	90
DS-CNN	88.41% ±8.93%	89.17% ±8.79%	89.65% ±8.97%
LSTM	91.44% ±6.31%	92.44% ±6.72%	92.07% ±7.44%
Bi-LSTM	90.67% ±7.39%	92.19% ±7.07%	92.24% ±7.61%
(DS-CNN)-LSTM	91.00% ±7.63%	92.60% ±7.12%	**93.01%** **± 7.49%**
(DS-CNN)-(Bi-LSTM)	91.04% ±7.66%	91.98% ±7.66%	92.93% ±7.40%

outcomes of these tests are shown in Fig. 7. Note that the widths of the confidence intervals are plotted at 95%, calculated from Tukey's Q value, by default. As previously mentioned, every table showed significant differences in the performance of the DS-CNN algorithm and any of the other four models. However, between the hybrid models and those based solely on LSTM, there appears to be statistical equivalence. Similarly, for the groups of results concerning each individual hyperparameter, after applying another Tukey test, no statistical differences were observed between them.

Moreover, it is also possible to observe how those values change notably depending on the selected window size. Table 8 shows the mean values of the accuracy obtained for each selected algorithm and window size, in a general way, showing in small, below each value, its standard deviation. Likewise, Table 9 shows the mean F_1-score values. As can be seen, these values are higher when the window size is larger (60 and 90 s) compared to those corresponding to a window size of 30 s. In the same way as before, another Tukey test was performed for the selected window sizes (30 and 90, 30 and 60 and 60 and 90), with their corresponding detailed performances from the tables above. The results of this test can be seen in Fig. 8. Only in the case of 60 and 90 s was the p-value greater than 0.1, so no statistically significant difference was found between both sets. However, for the 30-s case, there is a statistical difference with either of the other two values. That reaffirms the hypothesis already shown in previous works such as [25], where larger window sizes obtained superior results. In fact, if we go at the very nature of the activities studied in the dataset used, they have a long-themed character, so it is logical to think that longer time intervals positively affect the classification of their corresponding data.

All things considered, a peak performance of 94.80% accuracy and 94.27% F_1-score is achieved, corresponding to the (DS-CNN)-(LSTM) model, with a window size of 90 s, a single layer, 64 neurons and a kernel size of 7. The average confusion matrix corresponding to this case can be seen in Table 10, along with its particular metrics (recall, precision and accuracy). As can be seen, the model is able to classify any of the four activities with great accuracy, although there are slight problems with the correct identification of the "active" class. This activity is quite fuzzy, as it can accommodate actions where there may be periods of

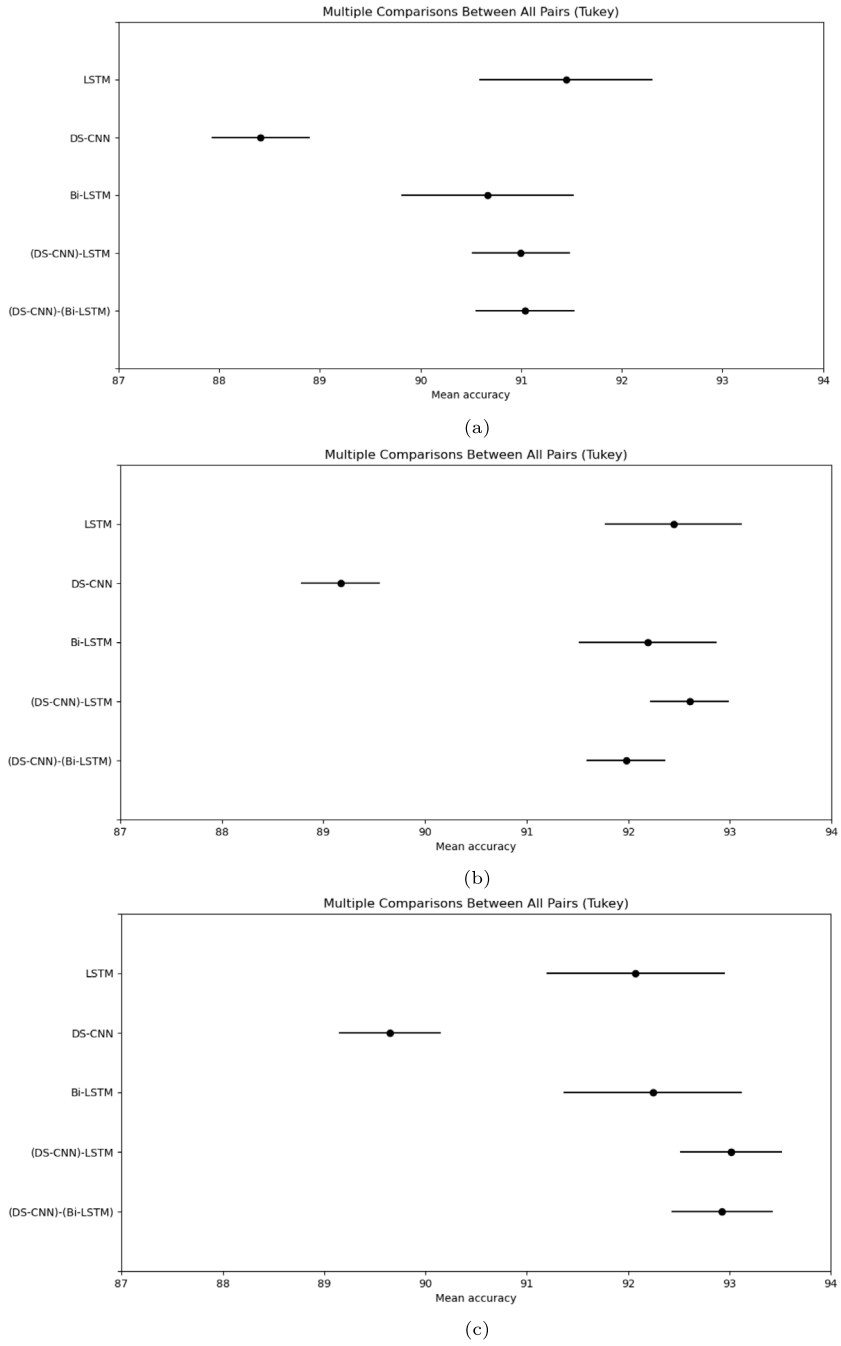


Fig. 7. Tukey test results for each group of accuracy values referring to each implemented algorithm, for each selected window size: (a) 30 s. (b) 60 s. (c) 90 s.

inactivity or where the individual is walking, which could lead to misclassification into these classes. An example of an action that could fall into this class and could be easily confused would be giving a lecture. This action alternates between times when the person may be walking (moving around the classroom) or sitting at the computer. In the first case, that walking moment could result in classifying samples of "active" as "walking". In the second case, sitting without moving at all is very similar to not having a cell phone on you, which could lead to misclassifying this action as "inactive". Even with the "driving" activity there could be confusion, since sitting in the car waiting at a red light, without moving, could also be difficult to classify correctly, even taking into account the vibrations of motor vehicles. Therefore, it is considered that, despite the discrepancies observed, the model is capable

Table 9
Overall F_1-score results obtained for each window size.

	Window size		
	30	60	90
DS-CNN	88.05% ±7.68%	88.25% ±8.95%	88.84% ±8.93%
LSTM	91.17% ±5.12%	92.52% ±5.70%	92.38% ±5.83%
Bi-LSTM	90.61% ±5.78%	92.28% ±5.77%	92.21% ±6.67%
(DS-CNN)-LSTM	90.60% ±6.64%	92.43% ±6.03%	**92.95% ± 6.22%**
(DS-CNN)-(Bi-LSTM)	90.67% ±6.71%	91.82% ±6.92%	92.87% ±6.25%

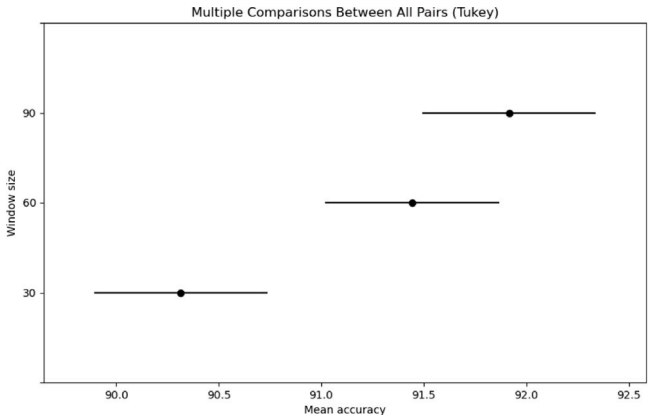


Fig. 8. Tukey test results for each group of accuracy values referring to each selected window size.

Table 10
Average confusion matrix for the best combination found.

	Ground truth				Precision
	Inactive	Active	Walking	Driving	
Inactive	1993.4	42.2	3.8	3.4	97.58%
Active	40.6	1226.8	51.2	20.3	91.63%
Walking	2.9	45.8	613.7	4.9	91.97%
Driving	11.7	7.5	4.5	515.3	95.60%
Recall	97.31%	92.78%	91.16%	94.74%	**94.80%**

of classifying the data exceptionally well, improving the results obtained with the most traditional machine learning techniques, going from 92.97% accuracy to 94.80%.

In any case, given the statistical equivalences observed previously, any model, except the DS-CNN, with a window size of 60 or 90 s, could be chosen as the preferred solution to the required classification. Thus, if the least complex option were sought, among all the statistically equivalent ones, an LSTM model with a single layer and 16 neurons could be sufficient for the problem to be solved. Likewise, a window size of 60 s could be chosen, since it would enable a more fitting classification of the activities under examination by permitting their segregation into 60-s intervals. Table 11 depicts the average confusion matrix for this particular option. As can be seen, the classification is similar to that of the best case obtained, but the confusion with the "active" class is more accentuated. In light of that, it is possible to select this choice as preferred.

Furthermore, in addition to all the experiments conducted with the proposed approach, it was decided to perform an ablation study. In this case, this investigation involved isolating each of the initial branches of the general model. Thus, the outputs of each one go directly to the subsequent layers of Dropout and Softmax, bypassing the concatenation layer. The aim is to observe the approximate influence of each sensor on the final classification based on their individual results. In this way, experimentation was done only with the best-case scenario found previously, corresponding to the confusion matrix in Table 10. As a result, more specific results are obtained while avoiding overloading the paper with extensive tables. With this configuration, the results shown in Table 12 were obtained. As can be seen, the accelerometer and gyroscope were by far the most accurate sensors. It is worth noting

Table 11
Average confusion matrix for the least complex case and statistically equivalent to the best one found.

	Ground truth				Precision
	Inactive	Active	Walking	Driving	
Inactive	1927.6	41.6	63.7	2.2	94.72%
Active	139.6	1231.6	38.2	26.6	85.77%
Walking	10.3	55.3	612.3	8.7	89.18%
Driving	6.1	20.2	6.2	534.1	94.26%
Recall	92.51%	91.32%	84.99%	93.44%	**91.14%**

Table 12
Results from the ablation study, for each individual sensor and with the best configuration found in prior experiments.

	Accuracy	F_1-score
Accelerometer	91.70% ±6.77	90.55% ±6.34
Gyroscope	90.55% ±6.93	87.56% ±9.42
Magnetometer	80.69% ±14.30	80.69% ±14.30
GPS	81.96% ±14.09	83.70% ±11.77

that even on their own, they can surpass accuracies of 90%. However, both the magnetometer and GPS yielded considerably lower results. Nevertheless, they manage to secure 80% accuracy. This is quite acceptable considering the nature of these sensors, which, while beneficial for HAR, do not adapt as well to this field as the accelerometer or gyroscope. All in all, although the individual results exhibit notable disparities, it is crucial to acknowledge that the measurements from each sensor could hold considerable value contingent upon the specific context. Significantly, some sensors might prove more suitable than others, depending on the movement type to be analysed. As a result, given the inherent variability present in the used dataset, the combination of all the sensors yields the best outcomes achieved to date for this particular dataset.

5.2. Discussion

The outcomes of this paper proved that deep learning algorithms are one of the best options in HAR, even in real-life environments such as the one discussed here. The accuracy obtained with the best combination of hyperparameters improves on that obtained with the most traditional machine learning algorithms, from 92.97% to 94.80%. Table 13 shows the comparison of the best results obtained with the methods used in the present paper, with respect to those of other papers that also used the same dataset. As can be observed, the resulting hybrid model of combining DS-CNN and LSTM yielded the most exceptional outcomes, using a window size of 90 s. The superiority of the proposed method over previous approaches may be attributed to several factors. Firstly, it could be due to the choice of feature set. In earlier machine learning endeavours, this process was manually conducted, and the selection of features might not have been the most suitable for the problem at hand. In contrast, the deep learning algorithms presented here automatically perform feature selection, which could ultimately lead to improved results. In addition, combining the chosen sensors and enabling them to analyse data individually, before concatenating their evaluations, proved advantageous for this dataset. It should be noted that, in previous works, this evaluation was performed jointly, assessing all sensors simultaneously. Finally, the intrinsic nature of LSTM, capable of retaining information from the past, also appears highly suitable for HAR, given the obtained results. In this way, considering the window size as well, it could be concluded that this combination of peculiarities presents, to date, the optimal model for the used dataset.

In the previous section, it was possible to observe how the models based solely on DS-CNN had a lower performance than the rest of the models used. Nonetheless, it should be noted that the running times of this case are much lower than those of the other algorithms implemented. To highlight that, Fig. 9 shows a comparison of the average execution time (in seconds) of each algorithm over all the experiments carried out. All these operations were performed on NVIDIA A100 40 GB GPUs. Hence, although the results are objectively worse, it could be a good option when the available time is much more limited. However, it is curious to observe how these times are longer for the individual cases based on LSTM, compared to the hybrid models that have higher complexity. Probably, the feature extraction carried out by DS-CNNs, in addition to improving the final classification in these models, may also be helping to reach a convergence point more quickly.

As already observed in other works on the same dataset like [25], there seems to be a certain tendency to improve classification with larger window sizes. That is confirmed by the results obtained in this paper, where window sizes of 60 and 90 s performed objectively better than the 30-s case. However, there did not appear to be any fundamental difference in the other hyperparameters (number of layers, number of neurons and kernel size). Therefore, as discussed in the previous section, the less complex 60-s case could be used preferably instead of the best combination one.

Moreover, it is meaningful to highlight the findings from the ablation study. As expected, the accelerometer and gyroscope outperformed the magnetometer and GPS. Nevertheless, it is significant to see these results validated in a real-life dataset, as this

Table 13
Comparison of the best results obtained with the methods used in the present paper, with respect to those of other papers that worked with the same dataset.

Work	Algorithm	Window size (s)	Eval. method	Accuracy
[23]	Support Vector Machine	20	10-fold	69.28% ±15.10%
[25]	Support Vector Machine	80	10-fold	86.56% ±11.30%
	Decision Tree	20	10-fold	89.99% ±6.13%
	Multilayer Perceptron	40	10-fold	86.85% ±6.12%
	Naïve Bayes	80	10-fold	83.27% ±7.78%
	K-Nearest Neighbour	80	10-fold	89.02% ±8.00%
	Random Forest	80	10-fold	92.97% ±6.23%
	Extreme Gradient Boosting	70	10-fold	92.23% ±7.30%
This work	DS-CNN	90	10-fold	90.70% ±7.29%
	LSTM	90	10-fold	93.52% ±5.59%
	Bi-LSTM	90	10-fold	93.09% ±5.10%
	(DS-CNN)-LSTM	**90**	**10-fold**	**94.80% ± 4.09%**
	(DS-CNN)-(Bi-LSTM)	90	10-fold	94.16% ±5.06%

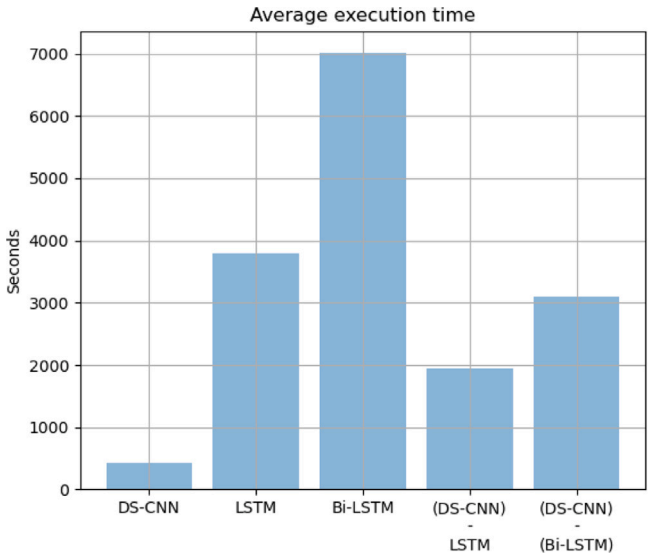


Fig. 9. Average execution time (in seconds) required to complete each of the experiments carried out by each of the implemented models.

confirmation had not been previously conducted. Additionally, the experiments with the comprehensive model demonstrated how combining these sensors, each with its unique characteristics and measurements, positively influenced the overall results.

Furthermore, it is also worth noting that there is still confusion with the "active" class. As previously mentioned, this class encompasses a multitude of actions that could be pretty fuzzy for the classification carried out by the model. Within an "active" session, there may be periods of inactivity or when the individual is walking, which may be detected by the model and marked as an incorrect activity. In any case, these confusions are minor, and they may be simply limitations in the dataset itself. Anyhow, it is feasible to think that these results could be improved, perhaps with other ways of preprocessing the data or with algorithms that may arise in the following years.

6. Conclusions and future work

This paper presents a brand-new set of experiments in human activity recognition (HAR) from smartphone sensor data from activities performed in a real-life environment. To carry them out, the deep learning algorithms that are yielding the best results in this area were applied: Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) models. By comparing their variants, combining them and making an exhaustive study of the different hyperparameters to be used, it was possible to improve the results previously achieved with the same dataset and more traditional machine learning techniques.

The results show that the most suitable models to exploit such data are those based on LSTMs, especially in conjunction with CNNs. However, at the hyperparameter level, no notable differences were observed concerning performance with different numbers of layers, neurons or kernel size. Nonetheless, improvements were detected when the window sizes were wider. When these window sizes were presented in time intervals of 60 or 90 s, the results improved substantially, compared to those obtained with window

sizes of 30 s. The activities studied in the used dataset have a long-themed nature, so it is plausible to think that longer time intervals may ease the classification of the samples fed to the implemented models.

Furthermore, it is also worth acknowledging the results of the ablation study. The accelerometer and gyroscope have indeed shown more robust performance in HAR, yielding high-grade results. That might lead to anticipate that their accuracies would surpass those of the magnetometer and GPS. However, this had not been confirmed until now on a real-life dataset. Additionally, the combination of all four sensors, each with its own distinctive characteristics that could influence the outcomes more or less positively depending on the study context, resulted in the best performance achieved to date for this dataset.

Moreover, it is also worth noting that the "active" class remains the most difficult to classify. Anyhow, in this case, the confusions are significantly lower than in other works. Given the fuzzy nature with which this activity was defined, it is possible that the results cannot be improved much further and that this is a restriction of the dataset used. Perhaps it is time to sharpen the focus and tackle much more specific activities, allowing the transfer of the acquired knowledge to everyday environments with better precision.

In any case, different data treatments could lead to better results. After all, in order to balance the sampling rates of the sensors used to collect the data, an experimental solution had to be implemented, discussed in detail in Section 4.1. A much more thorough exploration of how to address this issue, perhaps with the application of different types of interpolations or specific treatments for each kind of signal, could further refine the proposed models for the dataset used.

On the contrary, while following the same approach, the stratified 10-fold cross-validation applied to the data could have been conducted differently. As a result, data from the same individual could potentially appear in both the training and test sets using this methodology. However, it is worth noting that the data exhibit considerable variability in the different actions to be performed, as mentioned in Section 4.1. As a consequence, the impact of this separation may not be substantial, and the results could be reasonably consistent with those achieved in this paper. Nevertheless, the outcome of implementing a system that guarantees such differentiation remains uncertain. Therefore, even though the performance of the proposed models is considered outstanding, further studies could be conducted to explore and identify the best model and treatment for real-life datasets.

CRediT authorship contribution statement

Daniel Garcia-Gonzalez: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Daniel Rivero:** Conceptualization, Formal analysis, Methodology, Supervision, Writing – review & editing. **Enrique Fernandez-Blanco:** Conceptualization, Formal analysis, Methodology, Supervision, Writing – review & editing. **Miguel R. Luaces:** Funding acquisition, Project administration, Resources, Supervision.

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.

Data availability

The original complete dataset, as well as some of the scripts used to preprocess its data, can be found online at http://lbd.udc.es/research/real-life-HAR-dataset. Likewise, these have also been uploaded to Mendeley Data (https://data.mendeley.com/datasets/3xm88g6m6d/2). In addition, the code used for the entire data preparation and experimentation in this work is available online at http://gitlab.lbd.org.es/dgarcia/deep-learning-models-har.

Acknowledgments

We express our gratitude to CITIC and CESGA for their assistance in executing the code associated with this paper.

Funding

This research was partially funded by MCIN/AEI/10.13039/501100011033, NextGenerationEU/PRTR, FLATCITY-POC, Spain [grant number PDC2021-121239-C31]; MCIN/AEI/10.13039/501100011033 MAGIST, Spain [grant number PID2019-105221RB-C41]; Xunta de Galicia/FEDER-UE, Spain [grant numbers ED431G 2019/01, ED481A 2020/003, ED431C 2022/46, ED431C 2018/49 and ED431C 2021/53]. Funding for open access charge: Universidade da Coruña/CISUG.

References

[1] J.K. Aggarwal, L. Xia, Human activity recognition from 3d data: A review, Pattern Recognit. Lett. 48 (2014) 70–80.
[2] Y. Wang, S. Cang, H. Yu, A survey on wearable sensor modality centred human activity recognition in health care, Expert Syst. Appl. 137 (2019) 167–190.
[3] E. Soleimani, E. Nazerfard, Cross-subject transfer learning in human activity recognition systems using generative adversarial networks, Neurocomputing 426 (2021) 26–34.
[4] A. Subasi, M. Radhwan, R. Kurdi, K. Khateeb, IoT based mobile healthcare system for human activity recognition, in: 2018 15th Learning and Technology Conference (L&T), IEEE, 2018, pp. 29–34.
[5] F. Demrozi, G. Pravadelli, A. Bihorac, P. Rashidi, Human activity recognition using inertial, physiological and environmental sensors: a comprehensive survey, IEEE Access (2020).
[6] R. Liu, A.A. Ramli, H. Zhang, E. Datta, E. Henricson, X. Liu, An overview of human activity recognition using wearable sensors: Healthcare and artificial intelligence, 2021, arXiv preprint arXiv:2103.15990.
[7] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, L. Oukhellou, Y. Amirat, Physical human activity recognition using wearable sensors, Sensors 15 (12) (2015) 31314–31338.
[8] M.S. Zainudin, M.N. Sulaiman, N. Mustapha, T. Perumal, Monitoring daily fitness activity using accelerometer sensor fusion, in: 2017 IEEE International Symposium on Consumer Electronics (ISCE), IEEE, 2017, pp. 35–36.
[9] M. Raeiszadeh, H. Tahayori, A novel method for detecting and predicting resident's behavior in smart home, in: 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), IEEE, 2018, pp. 71–74.
[10] O.D. Lara, M.A. Labrador, A survey on human activity recognition using wearable sensors, IEEE Commun. Surv. Tutor. 15 (3) (2012) 1192–1209.
[11] M.M. Hassan, M.Z. Uddin, A. Mohamed, A. Almogren, A robust human activity recognition system using smartphone sensors and deep learning, Future Gener. Comput. Syst. 81 (2018) 307–313.
[12] Y. Tang, L. Zhang, F. Min, J. He, Multi-scale deep feature learning for human activity recognition using wearable sensors, IEEE Trans. Ind. Electron. (2022).
[13] M. Shoaib, S. Bosch, O. Incel, H. Scholten, P. Havinga, Complex human activity recognition using smartphone and wrist-worn motion sensors, Sensors 16 (4) (2016) 426.
[14] A. Ignatov, Real-time human activity recognition from accelerometer data using convolutional neural networks, Appl. Soft Comput. 62 (2018) 915–922.
[15] K. Xia, J. Huang, H. Wang, LSTM-CNN architecture for human activity recognition, IEEE Access 8 (2020) 56855–56866.
[16] P. Lago, S. Takeda, T. Okita, S. Inoue, Measured: Evaluating sensor-based activity recognition scenarios by simulating accelerometer measures from motion capture, in: Human Activity Sensing, Springer, 2019, pp. 135–149.
[17] I.A. Lawal, S. Bano, Deep human activity recognition using wearable sensors, in: Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments, 2019, pp. 45–48.
[18] S. Jeong, D. Oh, Development of a hybrid deep-learning model for the human activity recognition based on the wristband accelerometer signals, J. Internet Comput. Serv. 22 (3) (2021) 9–16.
[19] A. Stisen, H. Blunck, S. Bhattacharya, T.S. Prentow, M.B. Kjærgaard, A. Dey, T. Sonne, M.M. Jensen, Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition, in: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, 2015, pp. 127–140.
[20] E. Sansano, R. Montoliu, O. Belmonte Fernandez, A study of deep neural networks for human activity recognition, Comput. Intell. 36 (3) (2020) 1113–1139.
[21] N. Lane, Y. Xu, H. lu, S. Hu, T. Choudhury, A. Campbell, F. Zhao, Enabling large-scale human activity inference on smartphones using community similarity networks (CSN), in: UbiComp'11 - Proceedings of the 2011 ACM Conference on Ubiquitous Computing, 2011, pp. 355–364.
[22] A. Ferrari, D. Micucci, M. Mobilio, P. Napoletano, On the personalization of classification models for human activity recognition, IEEE Access 8 (2020) 32066–32079.
[23] D. Garcia-Gonzalez, D. Rivero, E. Fernandez-Blanco, M.R. Luaces, A public domain dataset for real-life human activity recognition using smartphone sensors, Sensors 20 (8) (2020) 2200.
[24] L. Hu, K. Zhao, B.W.-K. Ling, Y. Lin, Activity recognition via correlation coefficients based graph with nodes updated by multi-aggregator approach, Biomed. Signal Process. Control 79 (2023) 104255.
[25] D. Garcia-Gonzalez, D. Rivero, E. Fernandez-Blanco, M.R. Luaces, New machine learning approaches for real-life human activity recognition using smartphone sensor-based data, Knowl.-Based Syst. (2023) 110260.
[26] S. Mekruksavanich, A. Jitpattanakul, Biometric user identification based on human activity recognition using wearable sensors: An experiment using deep learning models, Electronics 10 (3) (2021) 308.
[27] I.U. Khan, S. Afzal, J.W. Lee, Human activity recognition via hybrid deep learning based model, Sensors 22 (1) (2022) 323.
[28] E. Ramanujam, T. Perumal, S. Padmavathi, Human activity recognition with smartphone and wearable sensors using deep learning techniques: A review, IEEE Sens. J. 21 (12) (2021) 13029–13040.
[29] D. Anguita, A. Ghio, L. Oneto, X. Parra, J.L. Reyes-Ortiz, A public domain dataset for human activity recognition using smartphones, in: Esann, 2013.
[30] J.R. Kwapisz, G.M. Weiss, S.A. Moore, Activity recognition using cell phone accelerometers, ACM SigKDD Explor. Newsl. 12 (2) (2011) 74–82.
[31] D. Micucci, M. Mobilio, P. Napoletano, Unimib shar: A dataset for human activity recognition using acceleration data from smartphones, Appl. Sci. 7 (10) (2017) 1101.
[32] D. Anguita, A. Ghio, L. Oneto, X. Parra, J.L. Reyes-Ortiz, Training computationally efficient smartphone–based human activity recognition models, in: International Conference on Artificial Neural Networks, Springer, 2013, pp. 426–433.
[33] J.-L. Reyes-Ortiz, L. Oneto, A. Ghio, A. Samá, D. Anguita, X. Parra, Human activity recognition on smartphones with awareness of basic activities and postural transitions, in: International Conference on Artificial Neural Networks, Springer, 2014, pp. 177–184.
[34] Z. Wu, A. Zhang, C. Zhang, Human activity recognition using wearable devices sensor data, 2015.
[35] Z. Chen, Q. Zhu, Y.C. Soh, L. Zhang, Robust human activity recognition using smartphone sensors via CT-PCA and online SVM, IEEE Trans. Ind. Inform. 13 (6) (2017) 3070–3080.
[36] S. Seto, W. Zhang, Y. Zhou, Multivariate time series classification using dynamic time warping template selection for human activity recognition, in: 2015 IEEE Symposium Series on Computational Intelligence, IEEE, 2015, pp. 1399–1406.
[37] W. Sousa, E. Souto, J. Rodrigres, P. Sadarc, R. Jalali, K. El-Khatib, A comparative analysis of the impact of features on human activity recognition with smartphone sensors, in: Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web, ACM, 2017, pp. 397–404.
[38] J. Yang, M.N. Nguyen, P.P. San, X.L. Li, S. Krishnaswamy, Deep convolutional neural networks on multichannel time series for human activity recognition, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
[39] C.A. Ronao, S.-B. Cho, Human activity recognition with smartphone sensors using deep learning neural networks, Expert Syst. Appl. 59 (2016) 235–244.
[40] Y. Guan, T. Plötz, Ensembles of deep lstm learners for activity recognition using wearables, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 1 (2) (2017) 1–28.
[41] F. Hernández, L.F. Suárez, J. Villamizar, M. Altuve, Human activity recognition on smartphones using a bidirectional LSTM network, in: 2019 XXII Symposium on Image, Signal Processing and Artificial Vision (STSIVA), IEEE, 2019, pp. 1–5.

[42] Y. Li, L. Wang, Human activity recognition based on residual network and BiLSTM, Sensors 22 (2) (2022) 635.
[43] W. Qi, H. Su, C. Yang, G. Ferrigno, E. De Momi, A. Aliverti, A fast and robust deep convolutional neural networks for complex human activity recognition using smartphone, Sensors 19 (17) (2019) 3731.
[44] S. Wan, L. Qi, X. Xu, C. Tong, Z. Gu, Deep learning models for real-time human activity recognition with smartphones, Mob. Netw. Appl. 25 (2) (2020) 743–755.
[45] E. Shalaby, N. ElShennawy, A. Sarhan, Utilizing deep learning models in CSI-based human activity recognition, Neural Comput. Appl. 34 (8) (2022) 5993–6010.
[46] Q. Teng, K. Wang, L. Zhang, J. He, The layer-wise training convolutional neural networks using local loss for sensor-based human activity recognition, IEEE Sens. J. 20 (13) (2020) 7265–7274.
[47] J. Figueiredo, G. Gordalina, P. Correia, G. Pires, L. Oliveira, R. Martinho, R. Rijo, P. Assuncao, A. Seco, R. Fonseca-Pinto, Recognition of human activity based on sparse data collected from smartphone sensors, in: 2019 IEEE 6th Portuguese Meeting on Bioengineering (ENBENG), IEEE, 2019, pp. 1–4.
[48] R.-A. Voicu, C. Dobre, L. Bajenaru, R.-I. Ciobanu, Human physical activity recognition using smartphone sensors, Sensors 19 (3) (2019) 458.
[49] Y.E. Ustev, O. Durmaz Incel, C. Ersoy, User, device and orientation independent human activity recognition on mobile phones: Challenges and a proposal, in: Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication, ACM, 2013, pp. 1427–1436.
[50] V. Janko, N. Rešçiç, M. Mlakar, V. Drobnič, M. Gams, G. Slapničar, M. Gjoreski, J. Bizjak, M. Marinko, M. Luštrek, A new frontier for activity recognition: The sussex-huawei locomotion challenge, in: Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers, 2018, pp. 1511–1520.
[51] N. Hnoohom, A. Jitpattanakul, S. Mekruksavanich, Real-life human activity recognition with tri-axial accelerometer data from smartphone using hybrid long short-term memory networks, in: 2020 15th International Joint Symposium on Artificial Intelligence and Natural Language Processing (ISAI-NLP), IEEE, 2020, pp. 1–6.
[52] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning, in: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), 2016, pp. 265–283.
[53] F. Chollet, et al., Keras, 2015, https://keras.io.
[54] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, E. Shelhamer, Cudnn: Efficient primitives for deep learning, 2014, arXiv preprint arXiv:1410.0759.
[55] K. Fukushima, S. Miyake, Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition, in: Competition and Cooperation in Neural Nets, Springer, 1982, pp. 267–285.
[56] Y. LeCun, P. Haffner, L. Bottou, Y. Bengio, Object recognition with gradient-based learning, in: Shape, Contour and Grouping in Computer Vision, Springer, 1999, pp. 319–345.
[57] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90.
[58] E. Fernandez-Blanco, D. Rivero, A. Pazos, Convolutional neural networks for sleep stage scoring on a two-channel EEG signal, Soft Comput. 24 (2020) 4067–4079.
[59] J. Zhu, H. Chen, W. Ye, A hybrid CNN–LSTM network for the classification of human activities based on micro-Doppler radar, IEEE Access 8 (2020) 24713–24720.
[60] S.K. Challa, A. Kumar, V.B. Semwal, A multibranch CNN-BiLSTM model for human activity recognition using wearable sensor data, Vis. Comput. (2021) 1–15.
[61] J. Nagi, F. Ducatelle, G.A. Di Caro, D. Cireşan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, L.M. Gambardella, Max-pooling convolutional neural networks for vision-based hand gesture recognition, in: 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), IEEE, 2011, pp. 342–347.
[62] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1251–1258.
[63] D. Alghazzawi, O. Rabie, O. Bamasaq, A. Albeshri, M.Z. Asghar, Sensor-based human activity recognition in smart homes using depthwise separable convolutions, Hum. Cent. Comput. Inf. Sci. 12 (2022) 50.
[64] E. Fernandez-Blanco, D. Rivero, A. Pazos, EEG signal processing with separable convolutional neural network for automatic scoring of sleeping stage, Neurocomputing 410 (2020) 220–228.
[65] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[66] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (11) (1997) 2673–2681.
[67] Y. Zhao, R. Yang, G. Chevalier, X. Xu, Z. Zhang, Deep residual bidir-LSTM for human activity recognition using wearable sensors, Math. Probl. Eng. 2018 (2018).
[68] J. Huang, S. Lin, N. Wang, G. Dai, Y. Xie, J. Zhou, TSE-cnn: A two-stage end-to-end CNN for human activity recognition, IEEE J. Biomed. Health Inform. 24 (1) (2019) 292–299.
[69] Y. Chen, Y. Xue, A deep learning approach to human activity recognition based on single accelerometer, in: 2015 IEEE International Conference on Systems, Man, and Cybernetics, IEEE, 2015, pp. 1488–1492.
[70] J.-L. Reyes-Ortiz, L. Oneto, A. Sama, X. Parra, D. Anguita, Transition-aware human activity recognition using smartphones, Neurocomputing 171 (2016) 754–767.
[71] M. Hossin, M. Sulaiman, A review on evaluation metrics for data classification evaluations, Int. J. Data Min. Knowl. Manag. Process 5 (2) (2015) 1.
[72] M. Bekkar, H.K. Djemaa, T.A. Alitouche, Evaluation measures for models assessment over imbalanced data sets, J. Inf. Eng. Appl. 3 (10) (2013).
[73] R. Kohavi, et al., A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Ijcai, Vol. 14, Montreal, Canada, 1995, pp. 1137–1145.
[74] L. Vanneschi, M. Castelli, S. Silva, Measuring bloat, overfitting and functional complexity in genetic programming, in: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, 2010, pp. 877–884.
[75] M. Cogswell, F. Ahmed, R. Girshick, L. Zitnick, D. Batra, Reducing overfitting in deep networks by decorrelating representations, 2015, arXiv preprint arXiv:1511.06068.
[76] L. Prechelt, Early stopping-but when? in: Neural Networks: Tricks of the Trade, Springer, 1998, pp. 55–69.
[77] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, 2012, arXiv preprint arXiv:1207.0580.