Received 26 May 2019; | Revised 31 January 2020; | Accepted 10 March 2020

DOI: 10.1111/coin.12318

ARTICLE TYPE

A study of Deep Neural Networks for Human Activity Recognition

Emilio Sansano[1] | Raúl Montoliu[2] | Óscar Belmonte Fernández[3]

[1]Institute of New Imaging Technologies, Universitat Jaume I, Castellón de la Plana. Castellón, Spain

[2]Institute of New Imaging Technologies, Universitat Jaume I, Castellón de la Plana. Castellón, Spain

[3]Institute of New Imaging Technologies, Universitat Jaume I, Castellón de la Plana. Castellón, Spain

Correspondence
Institute of New Imaging Technologies
Universitat Jaume I, Castellón de la Plana.
Castellón, Spain. Email: esansano@uji.es

Summary

Human activity recognition and deep learning are two fields that have attracted attention in recent years. The former due to its relevance in many application domains, such as ambient assisted living or health monitoring, and the latter for its recent and excellent performance achievements in different domains of application such as image and speech recognition. In this paper, an extensive analysis among the most suited deep learning architectures for activity recognition is conducted to compare its performance in terms of accuracy, speed, and memory requirements. In particular, convolutional neural networks (CNN), long short term memory networks (LSTM), bidirectional LSTM (biLSTM), gated recurrent unit networks (GRU) and deep belief networks (DBN) have been tested on a total of ten publicly available datasets, with different sensors, sets of activities and sampling rates. All tests have been designed under a multi-modal approach to take advantage of synchronized raw sensor' signals. Results show that CNNs are efficient at capturing local temporal dependencies of activity signals, as well as at identifying correlations among sensors. Their performance in activity classification is comparable to, and in most cases better than, the performance of recurrent models. Their faster response and lower memory footprint make them the architecture of choice for wearable and IoT devices.

KEYWORDS:
human activity recognition,deep learning,convolutional neural network,recurrent neural network

1 | INTRODUCTION

The main goals of Human Activity Recognition (HAR) systems are to observe and analyze human activities, and to interpret ongoing events successfully. There are several application domains, such as video surveillance systems, ambient assisted living (AAL) systems for smart homes or health care monitoring applications, that require a reliable activity recognition system.

HAR systems retrieve and process contextual data to classify human behavior into a set of complex activities (i.e. standing, walking, jogging). Image-based HAR analyzes human behavior using images or videos, which is generally considered a highly intrusive technique. On the contrary, sensor-based HAR systems study the motion data coming from wearable sensors such as accelerometers, gyroscopes, RFID, and so on. Besides the inclusion of such sensors, the ubiquity and unobtrusiveness of smartphones and smart-watches, and the availability of different wireless interfaces, such as Wi-Fi, 3G, and Bluetooth, make them an attractive platform for human activity recognition.

A traditional HAR pipeline employs individual smart devices to collect raw data from embedded sensors. Then, the associated activity is extracted from the data by applying machine learning and data mining techniques. Typically, this process has been divided into three components:

- **Sensing**: The system monitors an actor's behavior by gathering relevant data through a series of sensors, such as accelerometers or gyroscopes.
- **Feature processing**: The sensors' data is processed into handcrafted features that extract relevant and discriminative characteristics from the raw data.
- **Classification**: The human activity is inferred by a machine learning model designed to recognize an activity from the previously obtained features.

The aforementioned process can be simplified through the use of deep learning (DL), a subfield of machine learning that has recently achieved large success. Although the main concept of deep learning has been around for a few decades, it has not been until lately that this technique has emerged, thanks to the achievement of its excellent empirical performance in several different domains of applications such as speech recognition, image recognition, and natural language processing [1,2].

Deep learning finds features and classification boundaries through optimizing a certain loss function, employing a deep neural network architecture. The typical structure of a DL algorithm stacks multiple layers of simple neural network architecture to extract hierarchical abstractions. Each layer combines features derived from previous layers and transforms them via a non-linear function to produce a new feature set. This process builds a hierarchy where basic abstract features are detected in the first layers of the network and are combined in the deeper ones to form complex feature maps, giving the network the ability to automatically learn the best features for a specific problem domain. This feature set is learned by the model from the input data. There is no need for human intervention to manually craft explicative features. The training process of deep learning algorithms can extract key features from raw data and, at the same time, find the meaningful patterns that characterize each category of data (different activities in the case of HAR). This is the strong point of deep learning against traditional machine learning approaches.

The ubiquity of wearable devices provides an excellent platform to detect what activity is an user performing. In recent years, several works have explored the problem of identifying human actions using inertial signals from body-worn devices. Many of them investigated the use of manually engineered features extracted from the data as the input to different classical machine learning algorithms. More recently, the majority of research is centered on effectively applying DL to the detection of human activities.

The present work provides a broad comparison of the performance of five representative deep learning architectures for human activity recognition. The comparative is based on a set of premises:

- The main goal of this work is to compare different performance of various deep learning architectures in the problem of human activity recognition using raw inertial data. Thus, we avoid the use of engineered features.
- To provide a broader scope for comparison, we use a total of ten publicly available benchmark datasets. The differences among the datasets, such as the set of recorded activities, the subjects' conditions, the sensors typology (dedicated IMUs, smart-phones and smart-watches) or the on-body sensor placements, provide us with an enormous quantity of heterogeneous data that allows for reaching more meaningful and generalizable results.
- The data used for the experiments is composed of inertial signals from accelerometers and gyroscopes from a variety of devices worn on several positions, in different environments, and by various users.
- Previous works have showed that using accelerometer and gyroscope data provides better results than using either alone. Therefore, the data used as input to the algorithms is composed of the combined signals of both sensors.
- There is a significant number of different DL architectures, many of them aimed to solve very specific problems. We limit our comparison to five standard models that cover the majority of architectures used in recent works in the field of HAR and DL.

In this paper several models have been tested for each architecture to compare not only the performance but also the speed of each model and its memory footprint, to provide a broader basis and more solid grounds for conclusions. We summarize our contributions as follows:

- We conducted an extensive set of experiments using ten different publicly available datasets containing motion data from accelerometer and gyroscope sensors.

- We compared the general performance of five different deep learning architectures (DBN, CNN, LSTM, biLSTM, and GRU) on human activity recognition tasks.
- We compared the computational cost and memory footprint among the best performing models of each architecture, to assess their suitability in environments with a scarcity of resources, such as wearable and Internet of Things (IoT) devices.

As far as we know, this is the first paper comparing the performance and suitability of various DL architectures over such a large number of different datasets. The diversity and heterogeneity of the considered data, acquired by different people in several environments using diverse devices and procedures, enforces the conclusions of this work.

The remaining of the paper is organized as follows: Section 2 describes the related work on DL and human activity recognition. Section 3 provides a brief introduction to DL and the basis of the DBN, CNN and RNN architectures. Section 4 describes the datasets used in the study. The experiments description, methodology and setup are presented in Section 5. Results are presented and discussed in Section 6. Finally, Section 7 presents the main conclusions.

2 | RELATED WORK

Classical machine learning algorithms such as decision trees, bayesian methods (naïve Bayes and bayesian networks), k-nearest neighbors, neural networks (multilayer perceptron), support vector machines, fuzzy logic, regression models, Markov models (hidden Markov models, conditional random fields) and classifier ensembles (boosting and bagging)[3] have been used traditionally to address human activity recognition tasks. In the last years, following the wave of deep learning renaissance, several works have underlined the potential of a variety of deep learning architectures for activity recognition, such as deep convolutional neural networks (CNN), many types of deep feed-forward networks (FFN), such as deep belief networks (DBN) and Restricted Boltzman Machines (RBM), and different flavors of recurrent neural networks (RNN). These works show that DL algorithms are generally better than classical methods for activity recognition, with the added advantage of not requiring a preliminary stage of feature engineering.

Research on HAR using deep learning has been based on three different main approaches. In[4] researchers compared various algorithms for feature extraction using four datasets and accelerometer data. The results showed other techniques such as PCA+ECDF outperformed Restricted Boltzmann Machines[5] as feature extractors. Some later works[6,7,8,9] employed DBNs and RBMs in classification tasks. These works report better accuracy results compared with traditional approaches such as *k* Nearest Neighbors (kNN), Support Vector Machines (SVM), Hidden Markov Models (HMM), decision trees algorithms such as C4.5, logistic regression, etc. A more recent work[10] shows good results against classical algorithms but relies on a preprocessing stage to extract a set of features from sensor raw data.

Convolutional neural networks[11] have achieved state-of-the-art results in image recognition tasks, where the nearby pixels typically have strong relationships with each other. Similarly, in the human activity recognition realm, adjacent sensor readings are likely to be correlated. In multi-modal approaches, where more than one sensor are used to characterize an activity, correlations among distinct types of sensors may also have an impact on the correct interpretation of data. The CNN approach applied to HAR classification tasks has been proved to outperform previous state-of-the-art methods such as dynamic time warping (DTW), hidden Markov models (HMM) and support vector machines (SVM), among others. In one of the first works[12] to use CNN on raw sensor time series data for gesture recognition, researchers obtained better results than classical methods, outperforming another deep learning model such as bidirectional LSTM. In[13], researchers used a single convolutional layer architecture to classify accelerometer data, giving some insights on the best values for some parameters, such as the sampling layer pooling size, or the dropout and weight decay values to prevent over-fitting and improve generalization. Later works, such as[14,15,16,17,18,19,20] use more complex CNN architectures, with three or more convolutional layers, and explore the influence that the number of layers and other parameters have on the classification performance. These experiments show that increasing the number of convolutional layers increases the performance of the model. The main conclusion to be drawn from the aforementioned works is that CNNs are able to capture local dependencies, both among sensor axes and among different sensors, and to build powerful features upon them that result in a performance boost from previous non deep learning methods. These results confirm that, although this architecture has been used prominently for computer vision tasks, it is also relevant for sequence processing tasks, with the additional benefit of their relatively cheaper computational cost.

Some recent works[21,22,23,24] explore the performance of recurrent neural networks[25] on HAR classification tasks. This architecture has been especially conceived to work with sequential data, and thus, seems a good choice for HAR, since the input data

is typically composed of series of sensor readings. More advanced RNN architectures such as long short term memory (LSTM) units[26] and gated recurrent unit (GRU)[27] have eased the use of this type of neural networks, overcoming some important issues with the original RNN architecture. One of the first works[21] used bidirectional LSTM (biLSTM) to exploit the fact that activity recognition may depend on past and future context. The authors used a concatenation of accelerometer and gyroscope information synchronized in time, with no further feature engineering except the normalization of values between -1 and +1, as input to a biLSTM architecture. Authors in[22] used a combination of LSTM and CNN to perform activity recognition tasks on sensor data acquired from wearable devices. Their model outperformed previous results, including a CNN baseline, using the *opportunity* and *skoda* benchmark datasets. The use of ensembles of LSTMs[23] has been explored, with better classification results than previous deep learning models, including[22]. Finally,[24] compared the performance of various LSTM models against CNN, sequential extreme machine learning (ELM) and SVM on five benchmark datasets, obtaining good results and demonstrating the suitability of RNN architectures for HAR.

As far as we know, there is only one work that compares the performance of different deep learning architectures on HAR. In their paper, Hammerla et. al.[28] measure the effectiveness of FFNs, CNNs and two types of LSTM networks on three datasets, analyzing the influence of hyper-parameters on the performance of these algorithms, using accelerometer data as input. The conclusions of their work showed that CNN models obtained the best results for repetitive activities, while LSTM achieved better results on the *opportunity* dataset.

Table 1 shows a summary of previous works exploring deep learning effectiveness on HAR. The table shows the number of datasets used to asses the performance of the distinct algorithms, the deep learning architecture considered, and the typology of sensors used to classify the activities. The vast majority of previous studies use at least the accelerometer data, but many of them use it in conjunction with the gyroscope data, since the use of both sensors produces better accuracy[29,30]. Other elements such as magnetometers, light sensors, radio frequency identification (RFID), barometers, temperature sensors or WiFi readings, are also used in some papers.

From the aforementioned works some conclusions can be drawn:

- Deep learning models that take into account local dependencies to build descriptive features, such as CNN and RNN, seem more suitable for HAR than fully connected models, since the data used as input for activity recognition tasks consist of time series from sensor readings. On the contrary, fully connected network architectures such as DBN assume that inputs are independent of each other, so in order to model a time series it is necessary some previous feature engineering to include temporal information in the input data.
- Both RNNs and CNNs provide state-of-the-art performance. Although their architectures can be over-engineered, and even combined, to boost their classification scores, it remains unclear which type is more adequate for HAR tasks, both in terms of performance and resource consumption.

3 | DEEP LEARNING ARCHITECTURES

One of the core advantages of deep learning is its ability to automatically learn features from raw data. Previously proposed schemes for HAR remained in the conventional supervised learning paradigm that relies on the design of handcrafted features. Although these schemes could achieve high accuracy, the requirement for domain knowledge limits its scalability. Finding a good set of features from the raw data is crucial to isolate key information and highlight important patterns, but it requires expert knowledge and it is difficult and time-consuming. Deep learning eliminates the need for manual feature engineering.

There are several different architectures for deep neural networks (DNN). In general, they can be grouped into three main categories: Feed-Forward Networks, Convolutional Neural Networks and Recurrent Neural Networks. Their main characteristics are succinctly described in the next sections.

3.1 | Deep Belief Networks

A deep belief network is composed of several fully connected layers. Its structure is essentially the same as for a multi layer perceptron (MLP), where the only significative difference relies on the pretraining process. DBNs are formed by stacking restricted Boltzmann machines. RBMs are fully connected shallow neural networks composed by an input layer and a hidden layer, in which all the units are binary and stochastic. In a RBM, the visible units represent the observations, and are connected to the hidden units using weighted connections. The nodes of any single layer do not communicate with each other laterally.

TABLE 1 Summary of previous works. Architectures are: deep belief networks (DBN), restricted Boltzmann machines (RBM), convolutional neural networks (CNN), recurrent neural networks (RNN), long short term memory networks (LSTM), gated recurrent unit networks (GRU) and deep feed-forward networks (FFN). Sensors are: accelerometer (acc), gyroscope (gyr) and other types such as magnetometer, barometer, light, temperature, WiFi, etc (other).

Authors and year	Architectures	Datasets	Sensors
Plötz et al. (2011)[4]	DBN	3	acc
Lefebvre et al. (2013)[21]	LSTM	1	acc, gyr
Gjoreski et al. (2016)[31]	CNN	2	acc
Zeng et al. (2014)[13]	CNN	3	acc
Duffner et al. (2014)[12]	CNN	1	acc, gyr
Yang et al. 2015[14]	CNN	2	acc, gyr, other
Chen & Xue (2015)[15]	CNN	1	acc
Jiang & Yin (2015)[16]	CNN	3	acc, gyr
Zhang et al. (2015)[6]	DBN	3	acc
Ha et al. (2015)[32]	CNN	2	acc, gyr
Alsheikh et al. (2016)[7]	DBN	3	acc
Bhattacharya & Lane (2016)[8]	RBM	3	acc, gyr, other
Ordóñez & Rogen (2016)[22]	LSTM+CNN	2	acc, gyr, other
Hammerla et al. (2016)[28]	FFN, LSTM, CNN	3	acc
Radu et al. (2016)[9]	RBM	1	acc, gyr
Ronao & Cho (2016)[17]	CNN	1	acc, gyr
Zebin et al. (2016)[18]	CNN	1	acc, gyr
Ravi et al. (2016)[33]	CNN	4	acc, gyr
Guan & Plötz (2017)[23]	LSTM	3	acc, gyr, other
Murad & Pyun (2017)[24]	LSTM	5	acc, gyr, other
Camps et al. (2018)[19]	CNN	1	acc, gyr, other
Moya et al. (2018)[20]	CNN	3	acc, gyr

Restricted Boltzmann machines are simmetrical bipartite graphs. Their training process consists in learning to reconstruct data by themselves in an unsupervised approach, making several forward and backward passes between the visible and hidden layers. In the forward pass, the activations represent the probability of an output given a weighted input x: $p(a|x;w)$. In the backward pass, the result is an estimation of the probability of inputs x given the weighted activations a: $p(x|a;w)$. These two estimates lead to the joint probability distribution of inputs x and activations a: $p(x,a)$.

The training strategy for RBMs involves the concept of activation energy E_i. If there are n visible units and m hidden units, we can express the states of the visible and hidden layers with vectors v and h. For a given state (v,h) the energy in the RBM is:

$$E(v,h) = -\sum_{i=1}^{n} a_i v_i - \sum_{j=1}^{m} b_j h_j - \sum_{i=1}^{n}\sum_{j=1}^{m} v_i w_{ij} h_j \tag{1}$$

where a_i is the bias of the i^{th} visible unit, b_j is the bias of the j^{th} hidden unit, and w_{ij} express the weight between the visible unit i and the hidden unit j.

The marginal probability of the units, namely, the probability that the given weights will generate the visible units v is:

$$P(v) = \frac{1}{\sum_{v,h} e^{-E(v,h)}} \sum_{h} e^{-E(v,h)} \tag{2}$$

The training process works by updating the weights using the contrastive divergence (CD) method to calculate the gradients. This method approximates the gradients of the log-likelihood based on a short Markov chain started at the last example seen. Each time CD is run, it's a sample of the Markov Chain composing the restricted Boltzmann machine. The weights are updated following the rule:

$$\Delta w_{ij} = \epsilon(< v_i.h_j >_{data} - < v_i.h_j >_{recon}) \tag{3}$$

where ϵ is the learning rate, $< v_i.h_j >_{data}$ represents the expectation of the observed data and the results of the weights in the training set, and $< v_i.h_j >_{recon}$ is the reconstruction distribution of the model.

Once all the RBMs of a DBN have been trained, the generative pretraining stage is finished. Their weights are used as the initial weights of the DBN. The generative pretraining process helps the discriminative training of the model to achieve better generalization solutions[34]. This stack of RBMs might end with a Softmax layer to create a classifier, or it may simply help cluster unlabeled data in an unsupervised learning scenario.

DBNs can be used to classify human activities from raw sensor data by pretraining the network layers individually in an unsupervised way and then fine-tunning the complete network with the backpropagation algorithm. In general, these architectures outperformed classical machine learning approaches, but today they are mostly out of favor and rarely used[35], at least for HAR classification tasks using raw inertial data. There is an inherent difficulty for this type of architecture to learn discriminative patterns from time series data, since it has separate parameters for each input feature. This forces the model to learn all the rules that characterize an activity separately at each position in the input, or in other words, at each time step. As an example, since a step detection can occur at any time step in the input data, in a feed forward network architecture the detection of the step has to be learned for each position in the input layer. By comparison, recurrent neural networks share the same weights across several time steps, and the pooling layers of convolutional neural network architectures provide partial invariance to small local translations. These characteristics make them more appropriate for time series classification

3.2 | Convolutional Neural Networks

Convolutional neural network architecture is inspired by the hierarchical structure of the human visual cortex, which processes the visual information coming from the eye through a series of ordered and inter-connected visual areas that perform feature recognition, from simple edge detection in the first areas, to more complex shape structures in the higher levels[36]. CNNs have gained popularity due to their ability to learn suitable representations and capture local dependencies from images or temporal series. In the last few years, the use of deep CNN models has led to very good performance on a variety of problems, such as visual and speech recognition. Human activity recognition is also a good field for convolutional architectures, especially when considering the translation invariance and temporally correlated readings of time-series signals from activities, and their hierarchical structure as a combination of small movements. Due to this potential to identify the representative patterns of HAR's signals, CNN have recently been applied to human activity recognition in many research papers.

The operation performed by a convolutional layer consists of an element-wise product followed by a sum. The input, as a 2D matrix, is convoluted with a learnable kernel, a 2D matrix of a particular size, in a sliding-window fashion. The result of this operation forms the output feature map, which is another 2D matrix. Note that more than one kernel can be applied to the input, hence the output will be composed of as many feature maps as kernels are used. Different kernels will perform different convolution operations on the inputs, such as edge detection or sharpening. Each kernel can be considered as a specific feature detector, so the key task when training a convolutional neural network is to get it to learn the best kernels, those that extract the most meaningful features from the input. The convolution layer operation for a two-dimensional input can be expressed as:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{4}$$

where K represents the kernel and I is the input of size mxn.

After the convolution, a nonlinear activation function is applied to enable nonlinear transformation of the data. Then, a pooling layer is used to subsample the data, by sampling one input out of every region it looks into. The most commonly used subsampling strategy is max-pooling (taking the maximum value of the input), but other strategies can be considered, such as average-pooling (taking the average value of the input) or probabilistic pooling (taking a random value from the input). Besides turning the input into a smaller representation of the original data, the pooling layer makes the model invariant to small translations of the input data. Therefore, the pooling layer does not do any learning, but introduces sparseness as well as translation invariance. Since it only considers the maximum or the average value in a local neighborhood, a small distortion in the input will not change the result of pooling.

Convolutional and pooling layers are the building blocks of convolutional neural networks. Many convolutional and pooling layers can be stacked to form a deep neural network architecture. These layers act as a hierarchical feature extractor, each one building feature detectors over the outputs of the previous layer. The lower layers obtain the local influence of the signals (for

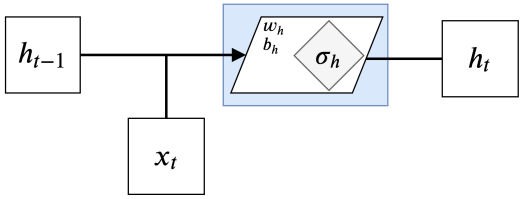

FIGURE 1 Vanilla recurrent neural network cell scheme.

example, the characteristics of each basic movement in a human activity), while the higher layers obtain a high-level representation features and patterns (for example, combinations of several basic movements). CNN can exploit the local dependency characteristics inherent in time-series sensor data and the translation invariant nature of activities.

3.3 | Recurrent Neural Networks.

Recurrent neural networks[25] are a family of neural networks specialized in processing a sequence of values. They have the ability to capture long-distance dependencies in the input stream, or, in other words, to remember information about previous inputs. Recurrent neural networks have one or more cycles, so it is possible to follow a path from a unit back to itself[37]. These cycles make it possible for RNNs to model long-distance dependencies, as they can pass information among time steps. When unfolded in time, an RNN can be seen as a feed-forward multilayer neural network with all layers sharing the same weights. The deepness of this unrolled network can be potentially infinite, as each layer represents a step in time. Unrolling the network allows to obtain a standard computation graph on which to apply forward computation and backward propagation, or backpropagation through time (BPTT)[38,39,40], to learn the parameters of the network. An RNN receives a sequence as input, which is processed in an internal loop over its elements (each sequence time step). In each step of the loop, the internal state of the RNN, a sort of 'memory' of previous time steps, is combined with the current time step input to produce an output. This operation can be expressed as follows:

$$h_t = \sigma_h(w_h[h_{t-1}, x_t] + b_h) \tag{5}$$

where w_h represents the weights of the cell, x_t is the input at current timestamp t, σ_h is the activation function, and h_t and h_{t-1} are the outputs (states) of the cell at current and previous time steps, respectively. Figure 1 shows a scheme of the inputs, outputs and operations that conform a RNN cell.

When a sequence has been processed, the internal state of the network is initialized to zero before processing the next sequence. While processing the input data, the recurrent layers generate an output for each time step in the sequence. Since each output is based on the current and all previous time steps, the last output will provide the most accurate prediction. This last output can be connected to a soft-max layer to perform multi-class logistic regression, producing a probability distribution over the activity class labels.

Vanilla RNN architecture introduces some critical issues, like the vanishing gradient and the exploding gradient problems[41], which makes optimization a great challenge. The vanishing gradient problem states that, given that for traditional activation functions the gradient is bounded, when these gradients are computed by backpropagation through the chain rule, the error signal decreases exponentially within time steps. This makes it hard for RNN to account for long-term dependencies, since the weights will not be updated beyond a few time steps. Depending on the activation function and the parameters used in the network, the problem can be the opposite, and the gradients can grow exponentially. The exploding gradient problem can be easily addressed by clipping the value of gradients to a predefined threshold[42], but the vanishing gradient is more problematic to correct. Regularization, the use of ReLU as the activation function, and proper initialization of the weights can reduce the effect of the problem.

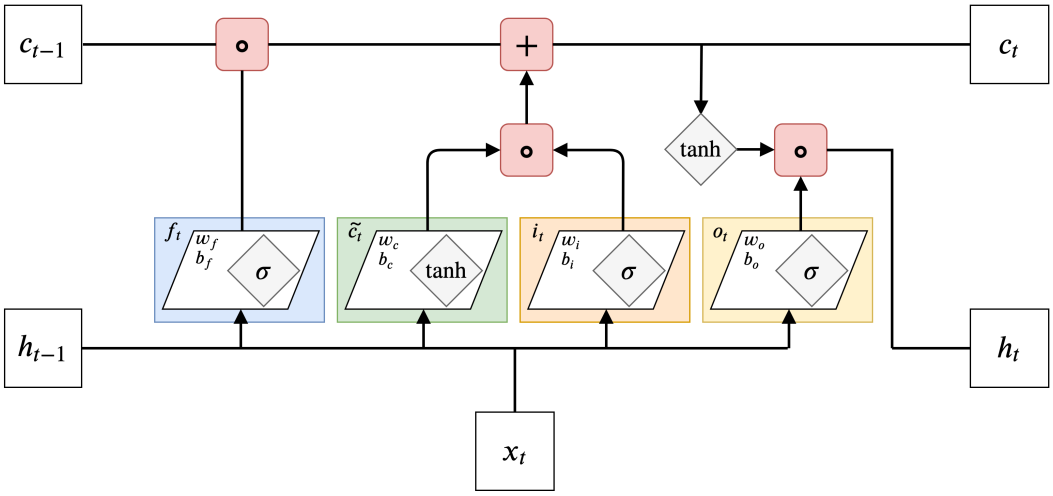

FIGURE 2 Long short-term memory cell scheme. Symbol ∘ represents the element-wise (Hadamard) product.

Some alternative RNN architectures have been developed to address such issues. Long short term memory networks[26] and gated recurrent unit networks[27] are known to be good solutions to bypass the vanishing/exploding gradient problem and efficiently learn long-range dependencies. LSTMs are the most widely RNN architectures used today to process sequential inputs like speech and language. The operation of an LSTM cell can be described as follows:

$$\begin{aligned} i_t &= \sigma(w_i[h_{t-1}, x_t] + b_i) \\ f_t &= \sigma(w_f[h_{t-1}, x_t] + b_f) \\ o_t &= \sigma(w_o[h_{t-1}, x_t] + b_o) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(w_c[h_{t-1}, x_t] + b_c) \\ h_t &= o_t \circ \tanh(c^t) \end{aligned} \tag{6}$$

where i_t, f_t, o_t are the output of the input, forget and output gates, respectively, w_i, w_f, w_o are the weights for the input, forget and output gates, respectively, w_c are the weights for the candidate layer, b_i, b_f, b_o are the biases for the input, forget and output gates, respectively, b_c is the bias for the candidate layer, σ is the sigmoid function, h_{t-1} is the output for the previous time step $t-1$, x_t is the input at current time step t, c_{t-1}, c_t are the cell states at time steps $t-1$ and t, h_t is the output of the cell at current time step t, and $\circ$ is the Hadamard product. Figure 2 shows a scheme of the inputs, outputs and operations that conform an LSTM cell.

GRUs are simplified versions of LSTMs. Compared to LSTM, GRUs have fewer parameters and are usually used to conserve memory or computation time. The main difference with LSTMs is that a single gating unit simultaneously controls the forgetting factor and the decision to update the state unit. The operation of a GRU cell can be described as follows:

$$\begin{aligned} z_t &= \sigma(w_z[h_{t-1}, x_t] + b_z) \\ r_t &= \sigma(w_r[h_{t-1}, x_t] + b_r) \\ h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \tanh(w_h[r_t \circ h_{t-1}, x_t] + b_h) \end{aligned} \tag{7}$$

where z_t, r_t are the output of the update and reset gates, respectively, w_z, w_r are the weights for the update and reset gates, respectively, b_z, b_r are the biases for the update and reset gates, respectively, σ is the sigmoid function, h_{t-1} is the output for the previous time step $t-1$, x_t is the input at current time step t, h_t is the output of the cell at current time step t, and $\circ$ is the Hadamard product. Figure 3 shows a scheme of the inputs, outputs and operations that conform a GRU cell.

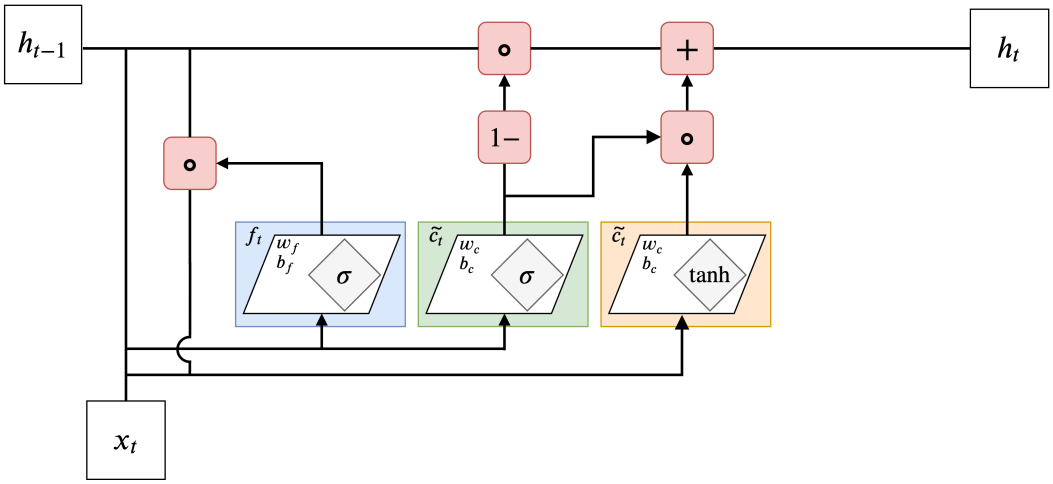

FIGURE 3 Gated recurrent unit cell scheme. Symbol ∘ represents the element-wise (Hadamard) product.

TABLE 2 Number of datapoints per dataset. Columns 1-12 represent each activity.

dataset	1	2	3	4	5	6	7	8	9	10	11	12	total
activemiles	13648	7020	1288	784	16304	3408	8404	-	-	-	-	-	50856
hhar	39226	41936	39237	45514	37464	33873	-	-	-	-	-	-	237250
swell	2348	2348	2473	1455	1712	2264	-	-	-	-	-	-	12600
fusion	8950	8950	8950	8950	8950	8055	8950	-	-	-	-	-	61755
usc-had	3744	2518	2685	2048	1904	1695	1000	2545	2290	3680	1552	1552	27207
mhealth	1200	1200	1200	1200	1200	1106	1148	1146	1200	1200	1200	384	13384
uci-har	2257	2059	1880	2359	2582	2567	-	-	-	-	-	-	13704
pamap2	22548	21634	22126	26378	11110	19208	21610	13638	12210	20560	27886	5424	224332
opportunity	28481	8116	16972	2945	-	-	-	-	-	-	-	-	56514
realworld	19375	22654	4695	31147	34777	27015	28987	27304	-	-	-	-	195954

4 | DATASETS

Focusing on a multi-modal approach by using a fusion of accelerometer and gyroscope data may be useful in accuracy-sensitive applications with a complex activity set, and allows for taking into account not only temporal dependencies, but also possible dependencies among sensors.

To conduct the experiments to assess the performance of different RNN and CNN architectures, we used a total of ten publicly available datasets that have been widely used within the community. These datasets contain continuous sensor readings from inertial measurement units (accelerometers and gyroscopes) worn by participants of the particular studies at different positions on their bodies, while performing typical tasks for human activity recognition. Some are realistic benchmark datasets, such as the *opportunity* dataset, while others might not be directly mirroring real-world situations but still are widely used in the research community, such as the *pamap2* dataset. The number of activities considered in each dataset, as well as the class balancing, show substantial variability among them, as shown in Table 2. This is usual for mobile application scenarios where there is a preeminence of static activities, such as sitting or standing, over dynamic activities such as walking or running. Examples of this scenario are health assessments, where problematic activities or behaviors are the rare exception within other more common activities.

For all the datasets we consider only recordings from on-body accelerometers and gyroscopes. We created 100 samples-wide sliding windows from the raw data, with 50% overlapping. Even though different overlap values can be used, an overlap of 50% has been shown to produce reasonable results[43,44,29]. The time span of the window is fixed for each dataset, and depends on the sampling rate of the raw data obtained from the sensors. For a rate of 50Hz, a window will contain 2 seconds of sensory data.

The main characteristics of the datasets considered in this experiment are as follows:

- **Activemiles**[45]: This dataset contains around 30 hours of labeled raw data from real world human activities performed by 10 subjects. The data was collected using five distinct smartphones with independent device configurations, sensor brands and sampling rates. No limitations were put on where the device was located (i.e., pocket, bag, or held in the hand). Annotations record the start time, end time, and a label describing the activity. For this work, both accelerometer and gyroscope raw data have been downsampled to 50Hz.

- **Fusion**[30]: The Fusion dataset collected data for seven basic motion activities in daily life (walking, running, sitting, standing, jogging, biking, walking upstairs and walking downstairs) from ten participants. Every subject performed each activity for 3–4 min, equipped with five smartphones on five body positions (left and right jeans pockets, right upper arm, right wrist, and right leg belt position). The same model of smartphone was used for all the positions, with different orientation depending on the device location. The data recorded included the readings from the accelerometer, the gyroscope and the magnetometer, all collected at a sampling rate of 50Hz.

- **hapt (UCI HAR)**[46]: The data in this dataset was recorded by 30 volunteer subjects who performed six different activities while wearing a waist-mounted smartphone. The subjects performed a protocol composed of six basic activities: three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs and walking upstairs). The raw accelerometer and gyroscope tri-axial signals where sampled at a rate of 50 Hz.

- **HHAR**[47]: The Heterogeneity dataset for Human Activity Recognition contains the readings of accelerometer and gyroscope sensors recorded while users executed activities carrying smartwatches and smartphones. The signals were sampled at the highest frequency the respective device allowed. A total of six different activities (biking, sitting, standing, walking, stairs up and stairs down) where performed by 9 subjects carrying 4 smartwatches, 2 on each arm, and 8 smartphones, all placed around the user's waist. Each participant conducted five minutes of each activity.

- **MHealth**[48]: Mobile Health dataset recorded 12 daily activities from 10 volunteers of diverse profile. This data was acquired by means of four different types of sensors: 3 tri-axial accelerometer sensors, 2 tri-axial gyroscope sensors, 2 magnetometer sensors, and 1 two-lead Electrocardiogram sensor, at a sampling rate of 50Hz. The sensors were placed on the subject's chest, right wrist and left ankle of the subjects.

- **Opportunity**[49]: The Opportunity Activity Recognition dataset consists of annotated recordings from a total of 4 participants, who wore 7 IMUs and 12 accelerometers placed on various body parts, and were instructed to carry out 18 different Activities of Daily Living (ADL), specifically focusing on kitchen routine. The sampling frequency for all IMUs was 30Hz. Every participant performed five different runs of the activities, following a loose description of the overall actions to perform. We only use the 4 activities from this dataset which are similar to the activities included in the rest of datasets; Stand, Walk, Sit and Lie, taking into account only the on-body sensors

- **Pamap2**[50,51]: The PAMAP2 Physical Activity Monitoring dataset contains data from 18 different physical activities, performed by 9 subjects wearing 3 inertial measurement units, attached to the hand, chest and ankle, and a heart rate monitor. The participants were instructed to carry out a total of 12 activities of daily living, plus 6 optional activities, covering domestic routines and various sportive exercises (Nordic walking, running, etc). The data was recorded from inertial measurement units located on the hand, chest and ankle of the participants, over a total of 10 hours.

- **Swell**[52]: This dataset contains raw accelerometer, magnetometer and gyroscope signals acquired by means of four smartphones placed on four body positions: right jeans pocket, belt, right arm and right wrist. It recorded six different phhysical activities, performed by four participants, at a sampling rate of 50Hz. The activities that the subjects conducted are walking, running, sitting, standing, walking upstairs and walking downstairs. The dataset contains roughly 3-5 minutes of data per activity and participant.

- **USC-HAD**[53]: The USC human activity dataset is intended as a benchmark for algorithm comparison, particularly for health-care scenarios, and consists of 12 basic human daily activities: walking forward, walking left, walking right, walking upstairs, walking downstairs, running forward, jumping up, sitting, standing, sleeping, in elevator up, and in elevator down. The dataset was recorded by 14 subjects wearing a high performance inertial sensor device, with tri-axial accelerometer and gyroscope, located on the front right hip, at a sampling rate of 100Hz.

- **RealWorld**[54]: Realworld is a complete and realistic dataset that covers seven activities (climbing stairs down and up, jumping, lying, standing, sitting, running/jogging, and walking) performed by fifteen subjects. Each subject performed each activity roughly 10 minutes except for jumping ($\sim$ 1.7 minutes). The dataset was recorded using seven wearable devices (smartphones and smartwatches) positioned on chest, forearm, head, shin, thigh, upper arm and waist. The devices were attached to the relevant body positions using a sport armband case, trouser pocket, shirt pocket, head belt, or the bra. There was no further fixation of the device to closely resemble their use in everyday life. The sampling frequency for all sensors was 50Hz.

5 | EXPERIMENTS SETUP

5.1 | Data preprocessing

The purpose of this stage is to obtain labeled segments from the continuous stream of data stored in each of the ten datasets used to evaluate the different deep learning architectures. These segments are sliding windows of sensor measurements, containing 100 consecutive readings from both sensors, accelerometer and gyroscope, for a particular activity, and with an overlapping factor of 50%. The time span of the window depends on the sampling rate at which the data was recorded. While for the *opportunity* dataset, whose data was acquired at 30Hz, a window represents 3.33 seconds of a particular activity, for the USC-HAD dataset, recorded at 100Hz, a window is 1 second wide. Some authors add the magnitude of measurements (the norm of the three axis) as an additional input to reduce the effect of sensor placement and orientation. This may increase the performance, but also the computational requirements and the training time. Since this experiment is intended to compare the performance of different architectures on the same data, we keep things simple and use only standardized raw data as input, with no further engineering.

Given the variety of sensors used in the benchmark datasets, with different typologies and brands, and thus distinct sensitivity and measurement ranges, it is convenient to scale the data to speed-up the convergence of DL algorithms. We performed preliminary tests to determine the best option among linear scaling in the range 0 to 1, linear scaling in the range -1 to +1, and standardizing to 0 mean and 1 standard deviation. The latter option showed the best results, with lower average convergence times for all architectures except DBN. Therefore, once the windows are extracted from the dataset, the values are standardized to have 0 mean and 1 standard deviation. In the case of DBN, to be able to work with real-valued data we use Gaussian distributed visible neurons and Bernoulli distributed hidden neurons in the first RBM. Consequently, input data has to be scaled in the range 0 to 1.

Figure 4 shows the resultant standardized distributions of accelerometer and gyroscope values for each dataset. The gyroscope distributions look similar, with the majority of values around zero, except for the *mhealth* dataset, that shows great variability. The main reason for this might be the sensor used, a Shimmer3 IMU with an *Invensense MPU9150* gyroscope, with low RMS noise and good range, sensitivity and resolution, in contrast with low-cost smartphone's sensors used in other datasets, combined with the small size of the dataset and the big number of activities (12) recorded. The accelerometer distributions present more diversity among datasets, with many peaks spread along the range of values.

5.2 | Experiment setup

Apart from standardizing the raw data and concatenating the accelerometer and gyroscope data together, sensor signals are used without employing any prior feature extraction methods, which is in line with the majority of recent deep learning based analysis methods in HAR and with the objective of this work to determine the capacity of various DL architectures to find convenient features for recognition of human activities. All the models, except for the DBN architectures, have been designed to receive as input the same data. Each data point has a dimensionality of 100x6, which corresponds to 100 time steps with 6 readings each, one for axis and sensor. In the case of DBNs, the input is reshaped to form a vector of 600 values. The composition of the different models have been designed taking into account previous works, but with the goal of using only simple architectures that are representative of each DL algorithm, to be able to compare the performance of each different structure without over-engineering the design to obtain state-of-the-art results. The guidelines when designing the models for each of the different deep learning architectures considered are as follows:

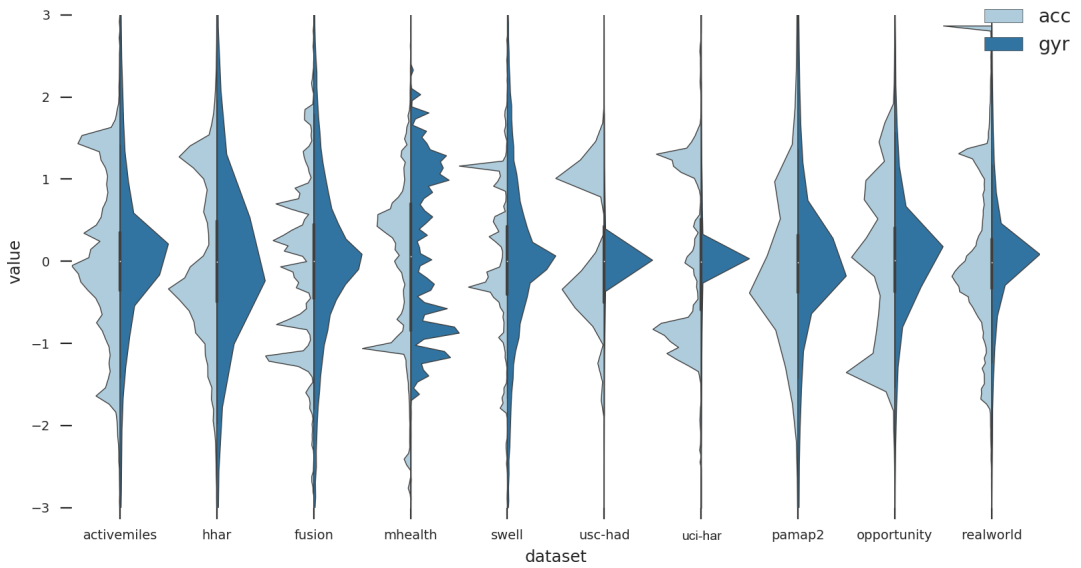

FIGURE 4 Distributions of accelerometer and gyroscope signals.

- DBN: The sizes of the models range from 1 to 6 RBM layers. In previous works[6,7], researchers used layer sizes ranging from 256 to 2000 hidden units. We fixed the number of hidden units to an intermediate value of 512, as it is showed in Table 3. Once pretrained, RBMs are stacked forming a DBN that is then fine-tuned in a supervised way using backpropagation algorithm, with the output of the last RBM connected directly to a soft-max layer for classification.
- CNN: The models are composed of three 2D convolutional layers (with convolutional and pooling stages) connected to a fully connected neural network and a soft-max classifier. This three-layer convolutional architecture has been used in previous works, such as[14,15,17], with good results for human activity recognition tasks. The filter sizes of the first layer range from 19x3 to 3x3. The models are designed to study the influence of the temporal size of the filter (the number of time steps it encompasses) on the performance of the model. On the axis dimension of the input, the filter of the first convolutional layer encloses the three axis of each sensor, with a stride factor of 3. Consequently, this layer will not take into account any dependency among sensors, but only dependencies among axis of the same sensor. It is not until the last convolutional layer that local dependencies between accelerometer and gyroscope are taken into account. Table 4 shows the configuration parameters for each CNN model. The last fully connected layer is sent directly to a soft-max layer for classification.
- LSTM, biLSTM and GRU: We use a similar architecture for all RNN models. The models are three layers deep and the number of hidden units range from 100 up to 600 for LSTM and GRU models, and from 100 to 500 for biLSTM models, as it is showed in Table 5. Previous works[21,24] used a similar architecture, with 100 hidden units per layer. Other authors use more complex structures involving the use of mixed architectures[23] such as CNNs combined with LSTMs. The last prediction of the last recurrent layer is sent directly to a soft-max layer for classification.

In order to avoid potential over-fitting, and also to reduce the influence that a particular partition of data could have on the final result, we evaluated the performance of each model using a random 5-fold stratified cross validation strategy. Prior to the training process of models, a test set, comprising 20% of the total data points, is extracted from each dataset. This set will be later used to assess the performance of the trained model. The remaining data is randomly divided into 5 equally sized and mutually exclusive folds. On each iteration, one fold is used as validation, and the remaining are used as training data. Once all the validation folds have been used, the performance is averaged among the five results obtained on the test set.

To further improve the training procedure and generalization performance, the following regularization techniques have been used:

TABLE 3 DBN models. Table shows the number of hidden layers, the number of units in each layer, and the number of trainable parameters of the model.

number of layers	hidden units per layer	number of parameters
1	512	312320
2	512	620032
3	512	927744
4	512	1235456
5	512	1543168
6	512	1850880

TABLE 4 CNN models. Table shows the size of the filter for the first convolutional layer of the models, the number of units in the fully connected layers (fcl1 and fcl2), and the number of trainable parameters.

first filter size	fully connected layer 1	fully connected layer 2	number of parameters
19x3	1536	100	1144676
17x3	1536	100	1176676
15x3	1536	100	1274212
13x3	1536	100	1306212
11x3	1536	100	748388
9x3	1536	100	845924
7x3	1536	100	943460
5x3	1792	100	673380
3x3	2816	100	742244

TABLE 5 LSTM, biLSTM and GRU models. All the models are composed of three recurrent layers and a soft-max output layer. Table shows the number of hidden units in each layer, and the number of trainable parameters of the model.

hidden units per layer	LSTM parameters	biLSTM parameters	GRU parameters
100	128400	256800	96300
200	496800	993600	372600
250	771000	1542000	578250
300	1105200	2210400	828900
350	1499400	2998800	1124550
400	1953600	3907200	1465200
450	2467800	4935600	1850850
500	3042000	6084000	2281500
550	3676200	–	2757150
600	4370400	–	3277800

– Dropout[55]: It is a regularization strategy that removes non-output units randomly from the original network, independently for each hidden unit and for each training case. Thus, applying dropout to a neural network is equal to sub-sampling a smaller and less complex neural network from it. Dropout was performed in all fully connected layers and in all recurrent layers, with a probability value of 0.5.

- Weight decay[56]: It is well known that more sparse neural networks generalize better. Weight decay penalizes large weights and limits the freedom of the model by adding an additional term in the weight update rule, forcing weights to be closer to zero. Based on previous works, such as[17], we chose a value of 0.00005 for weight decay.
- Early stopping: It is a simple technique that can be superior to other regularization methods in many cases, e.g.[57]. The validation error is used as an estimate of the generalization error, i.e., the validation set is used to anticipate the behavior on the test set, assuming that the error on both will be similar. The method consists in training the model on the training set and evaluating its performance every epoch on the validation set. Given an early stopping parameter k, the training process stops when k epochs have passed without any performance increase. The final performance metric is evaluated on the test set using the model state that performed better on the evaluation set.

In the training process we use the AdaGrad[58] updating function, with a learning rate of 0.005, and a mini-batch size of 1000 data points, except for the most complex recurrent models, on which we used a mini-batch size of 500. These parameters have been determined based on previous experiments and precedent research works on deep learning and HAR. We also used the cross entropy as loss function. As our evaluation metrics, we employ both the accuracy and the f1-score, which is defined as the harmonic mean of precision and recall:

$$f_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Since more than two classes are considered, we report the weighted average of the individual f1-scores of all classes as the evaluation metric for each model, in line with standard practices in HAR research[59], since it is more resilient than accuracy on imbalanced datasets:

$$\hat{f}_1 = \frac{\sum_{i=1}^{c} w_i f_1^i}{\sum_{i=1}^{c} w_i}$$

where c is the number of classes and w_i is the weight (the number of instances) of the i^{th} class.

6 | RESULTS AND DISCUSSION

6.1 | Results

The results obtained are summarized in Figure 5, which shows the best f1 and accuracy scores for each architecture and dataset. Using both metrics, CNN performed better approximately if half of the datasets considered, while GRU performed better on the rest. In all the experiments, DBN was the worst performing architecture.

Regarding the absolute scores obtained, one thing to note is that those datasets whose signals showed more variability (see Figure 4), such as *mhealth*, *fusion*, *hapt* or *swell*, achieved better general classification scores than the rest, rounding a maximum score of 99%. This highlights the important function that variability plays, making it easier for the models to find meaningful correlations among sensor signals, and among those and the correspondent activity, thus easing the discrimination among activities.

Figure 5 shows the performance loss in percentage over the best scores for each dataset. In those cases in which GRU is the best performing architecture, the average difference of performance over the CNN model is 0.67%. On the contrary, in those cases in which the CNN model obtains the best score, the best GRU model is 0.42% worse on average. The worst performing architecture (DBN) is 6.04% worse, on average, over the best architecture.

When comparing the speed and the memory footprint of the architectures, DBN is the fastest arquitecture, while CNNs are the most efficient in terms of memory use. Figure 6 shows the time that the best model of each architecture takes to process an input and generate a prediction, as a percentage over the time taken by the fastest model. In all cases the fastest architecture is DBN, which was an expected result since their architecture, regardless of the special training process, is the simplest.

Regarding memory footprint, Figure 6 shows the number of trainable parameters, as a percentage over the number of parameters of the least memory-consuming model, among the best for each architecture and dataset. In nine of the ten

(a) f1

Architecture	activemiles	hhar	fusion	mhealth	swell	usc-had	uci-har	pamap2	opportunity	realworld
CNN	**96.79**	**96.31**	**99.19**	99.18 -0.17%	**98.39**	92.75 -0.12%	**98.59**	92.44 -1.94%	**92.57**	91.48 -0.44%
GRU	96.33 -0.47%	96.28 -0.03%	99.08 -0.11%	**99.34**	98.15 -0.24%	**92.87**	97.98 -0.62%	**94.27**	91.63 -1.02%	**91.89**
biLSTM	96.37 -0.43%	95.82 -0.50%	98.75 -0.44%	99.15 -0.20%	97.62 -0.78%	92.39 -0.52%	97.56 -1.04%	93.30 -1.03%	90.34 -2.42%	91.18 -0.77%
LSTM	95.99 -0.82%	95.45 -0.89%	98.65 -0.54%	98.62 -0.72%	96.98 -1.44%	91.77 -1.18%	96.04 -2.59%	92.37 -2.02%	88.28 -4.63%	90.89 -1.09%
DBN	92.99 -3.93%	92.64 -3.81%	96.93 -2.28%	93.92 -5.46%	89.19 -9.35%	84.13 -9.41%	93.58 -5.09%	85.49 -9.32%	88.24 -4.68%	88.29 -3.92%

(b) accuracy

Architecture	activemiles	hhar	fusion	mhealth	swell	usc-had	uci-har	pamap2	opportunity	realworld
CNN	**96.84**	96.34 -0.05%	**99.29**	99.17 -0.23%	**98.48**	92.97 -0.06%	**98.74**	92.63 -1.87%	**92.44**	91.51 -0.48%
GRU	96.33 -0.53%	**96.39**	99.29 -0.00%	**99.39**	98.27 -0.21%	**93.03**	98.21 -0.54%	**94.40**	92.02 -0.45%	**91.96**
biLSTM	96.22 -0.64%	95.93 -0.47%	98.93 -0.37%	99.19 -0.21%	97.89 -0.60%	92.58 -0.49%	97.66 -1.10%	93.41 -1.05%	90.59 -2.00%	91.33 -0.68%
LSTM	96.02 -0.85%	95.52 -0.90%	98.77 -0.53%	99.03 -0.37%	97.18 -1.32%	92.11 -0.99%	96.41 -2.36%	92.56 -1.94%	88.79 -3.94%	90.91 -1.14%
DBN	91.43 -5.59%	92.46 -4.07%	97.13 -2.18%	93.32 -6.11%	89.36 -9.27%	82.13 -11.72%	93.49 -5.31%	85.08 -9.87%	87.13 -5.75%	88.55 -3.71%

FIGURE 5 Per dataset heatmap of the maximum f1 *(a)* and accuracy *(b)* scores for each architecture. The best score for each dataset is boxed in dashed lines. The table also shows the percentage of performance loss over the best architecture for each dataset.

datasets the best CNN model has a smaller memory footprint. The best performing LSTM model for each dataset needs an average of 202% more parameters. The best GRU models have an average of 312% more memory needs than the best CNN models.

As for the structure of the models of each architecture, Figure 7, shows the influence of the number of hidden units on the model's performance and training time of the different architectures, as a percentage over the performance of the simplest model. As might be seen in the figures, in the case of recurrent models, the positive impact of increasing the number of hidden units decreases as their number grows. Each step also increases the network's complexity, reaching a point where adding more units will not increase the model's performance. Obviously, each increase in complexity causes a slowdown of the model along with a boost in the cost of training. Bidirectional LSTM models give its peak of average performance at 450 hidden units, while GRU models average performance peaks at 500 hidden units and LSTM models at 550 hidden units.

Regarding the results obtained for the CNN models, Figure 7 shows the impact of the temporal size of the first convolutional layer filter. The best results are obtained with filters that are 11, 9 and 7 time steps wide, which for a sampling rate of 50Hz will correspond to roughly 0.2 seconds. As for the speed, the range between the worst and the best model (around 70%) is narrower than the range in recurrent models (around 500%), showing that changes in the filter size have a small impact on the speed of the model.

The results obtained for DBNs show that the best results are obtained with 3 hidden layers. As for the speed, as it can be expected, decreases as the complexity of the model grows.

The k-fold stratified cross validation strategy followed in these experiments implies that the training, validation and test sets have been drawn from the same distribution. This is, variables that can influence the parameters of the distribution for each set have been equally distributed, and patterns found in the training set will probably be found in the test data as well, thus facilitating the capacity of the trained model to achieve good results. One simple example of this kind of variable can be the user who acquired the data. Distinct users may show different peculiarities when performing activities, and using data from all users when training allow the model to learn these differences.

FIGURE 6 Per dataset heatmap of the time *(a)* taken to process an input and memory footprint *(b)* for the best model for each architecture, in percentage over the fastest. The best score for each dataset is boxed in dashed lines.

To assess the ability of the best-performing deep architectures to learn significative patterns in these type of scenarios, we selected three datasets to perform some additional tests. For each dataset, we trained the best performing model from the best architectures with a leave-one-out strategy, this is, we reserved the data from one user to evaluate the performance of the model and used the rest of the data to train the model. This process was repeated for each user. The reported result for each dataset and model is the average f1 score for all users. The number of users varies from 10 in the *fusion* and *mhealth*, to 30 users for the *uci-har* dataset. The results are presented in Figure 8. Results show a big difference in the general performance of deep models compared with previous results, where the data on which they where tested was very similar to the data on which they were trained. While these might be state-of-the-art models, with really high performance, it would be only on the very specific distribution represented by the training data. When used in a new task which might still be similar to the one they were trained on, as is the case of a new data from a new user, they end up suffering a significant loss in performance. In these circumstances, neural networks fail to form a general understanding of the meaningful patterns that characterize each class. Combating this issue [60] is a difficult problem which generally requires more data to train and/or specific training procedures to prevent overfitting. Ironically, the use of engineered features can help, to some extent, to prevent these issues. Using the same leave-one-out strategy on the set of manually extracted features provided in the *uci-har* dataset with a classic machine learning algorithm such as SVM provides an average accuracy of 0.75. Although this simple test might not be significative, it seems reasonable to consider that a good set of well selected features might be representative for the distinct characteristics of human activities, and more resilient to changes in the environment such as the user that is performing such activities.

Finally, in order to compare the performance of the trained deep learning models with various classical machine learning algorithms we performed some tests using the set of features provided in the *uci-har* dataset. Using three different machine learning classifiers (support vector machines, 3-nearest neighbors and random forest) the results obtained were noticeably inferior to those achieved by convolutional and recurrent networks. Table 6 shows the results obtained. The support vector machines classifier obtained the best results.

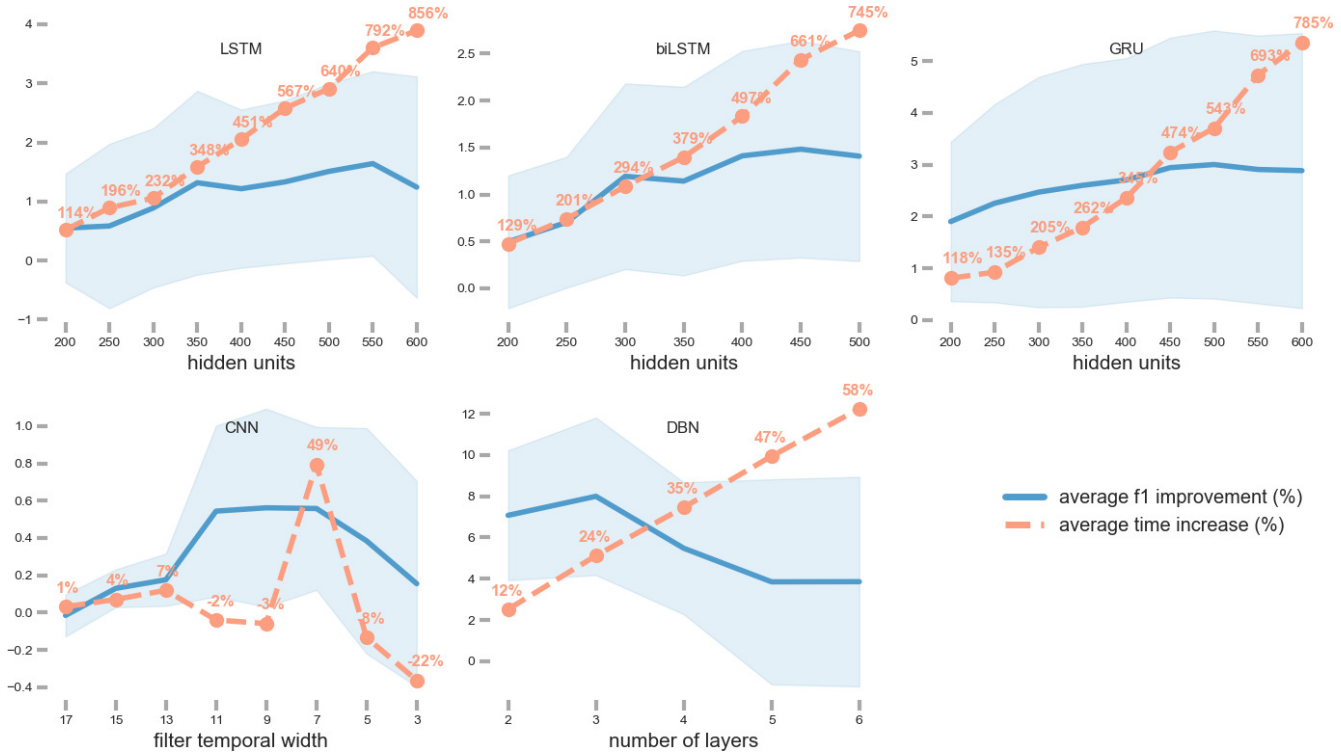

FIGURE 7 F1 score and time of prediction for each model, as a percentage over the simplest model. The grayed area represents the confidence level interval.

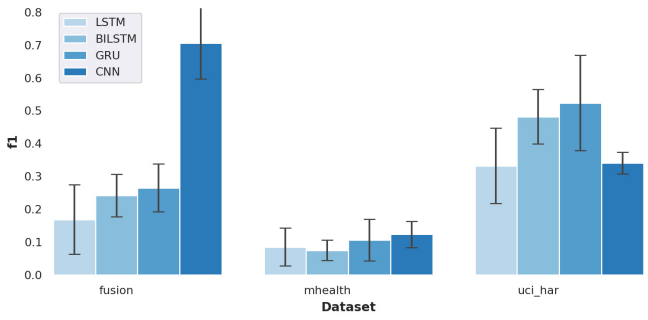

FIGURE 8 F1 score for each dataset and architecture when training with a leave-one-out strategy. Error bars show the standard deviation of the results.

6.2 | Discussion

One of the key points of the CNN architecture is its shift invariance property, that enables a pattern in the input to be recognized in any position. This type of DL architecture is able to successfully capture the temporal dependencies in raw sensor data structured as time series through the application of relevant filters. The pooling layers contribute providing limited scale invariance properties, allowing for slight deformations in the input signals. Given that different people may execute an action with distinct paces (e.g., an elder person may walk with a slower pace than a young person), this scale deformation tolerance enables CNNs to effectively identify the pattern that characterizes the activity regardless of its pace of execution. The temporal size of the filter for the best models, of around 10 time steps, indicates that there is no need to capture long-distance dependencies in the input stream to obtain good results characterizing human activities.

TABLE 6 Accuracy and f1 for three classical machine learning algorithms on the uci-har dataset using the set of features provided by the authors.

algorithm	accuracy	f1
support vector machines	0.9504	0.9504
3 nearest neighbors	0.8907	0.8898
random forest	0.9253	0.9251

Architectures designed to do so, such as RNN, do not perform consistently better than CNN, and have some disadvantages, such as their computational complexity and higher memory needs.

The unfolded computational graph of the RNN architecture results in the sharing of parameters across the deep network structure. This facilitates the use of the same parameters for different time steps, enabling the model to efficiently learn discriminative patterns regardless of their position in the input sequence. The results obtained with GRU models, which are comparable to the results achieved by CNN models, validate the use of GRU for human activity recognition tasks. This type of recurrent neural networks achieves better results than LSTM and bidirectional LSTM, at least using simple vanilla architectures, and come with the advantage of their relative low cost in terms of memory and computation resources, at least compared with other recurrent architectures.

Finally, we see that the fully connected architecture of DBN limits their suitability for time series classification tasks, at least without a previous manual feature extraction. The pretraining process using unlabeled data enables initializing layer weights in a way that should make feasible extracting significant features from the raw data. But the patterns that define the different activities can be present at any time stamp, and the fact that the weights are not shared implies that those pattern detectors have to be learned for each possible position in the input. However, since the availability of unlabeled data is significantly higher, exploring ways of leveraging this information to get better models is a research line on which these kind of generative models can have their application. Either DBN or other architectures such as autoencoders or variational autoencoders could be employed as feature extractors trained in an unsupervised process, and then connected to a CNN or RNN model for fine-tuning with labeled data.

In summary, the results are compatible with the hypothesis that CNN are efficient at capturing local temporal dependencies of activity signals, and are also capable of identifying multi-modal correlations among sensors. Their performance in activity classification is comparable to, and in some cases better than, RNNs. Their faster response and lower memory footprint make them the architecture of choice for wearable and IoT devices, where restrictions both in terms of power consumption, computational capabilities and memory availability might play a decisive role.

6.3 | Results reproducibility

To allow for the verification and validation of results, the software framework implemented to run the experiments has been made publicly available. The public repository [1] includes the log files of all the individual experiments and results for each model and architecture, as well as the scripts used to generate the plots and figures included in this paper. The software is easily extensible, and has been implemented with the aim to allow for the reproducibility of the experiments included in this paper, and to ease the creation of new ones to effortlessly extend this research. The repository provides details about the requirements, data sources and how to run experiments and create new ones. The datasets are not included in the repository, although instructions on how to obtain them is provided.

The experiments were conducted on two identical machines with Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz, NVIDIA GeForce GTX 1080 Ti GPU and running Ubuntu v17.10 as the operating system, using Pytorch version 0.3 and CUDA version 9.0. It took more than 4000 hours to complete the total set of experiments.

[1] https://github.com/esansano/dl-for-har-comparison

7 | CONCLUSSIONS AND FUTURE WORK

In this paper, we used a testbed of ten publicly available benchmark datasets to compare the performance, speed and complexity of five different architectures of DNNs on HAR classification tasks. The results show that CNNs achieve very competitive performance scores with respect to GRU networks, and better results that biLSTM, LSTM and DBN models. Regarding memory requirements, CNN is clearly the least memory demanding architecture.

The continuous progression of artificial intelligence into the mainstream due to the advent of enabling technologies such as machine learning, smart devices, cloud storage or high-speed networks, will require systems to be able to provide real-time insights in key fields such as ambient assisted living or health monitoring, often through the use of low-cost embedded devices with strict restrictions on memory availability, power consumption, connectivity, and computational capabilities. Overall, both in terms of speed and memory requirements, the study allows to confirm that CNNs are especially appropriated for activity recognition tasks in such scarce resources scenarios.

Future work. Given the extensive quantity of HAR data available, we plan to assess the possibility of using pre-trained models to speed up training and improve the performance on datasets where few data is available, decreasing thus the need for large volumes of new data and the time needed to adjust a new model. Transfer learning[61] makes use of the knowledge acquired by a model during the training process on a big dataset to solve a related classification problem where few labelled data is available. For example, a CNN can be trained with a big dataset such as *pamap2* or *realworld*, or a combination of both, and, once the training process has ended, be tuned to classify classes from a dataset with a limited amount of data and a different set of activities, such as *uci-har* or *swell*.

ACKNOWLEDGMENTS

This work has been partially funded by the Spanish Ministry of Science, Innovation and Universities through the "Proyectos I + D Retos investigación" programme (RTI2018-095168-B-C53) and by Jaume I University "Research promotion plan 2017" programme (UJI-B2017-45).

Financial disclosure

None reported.

Conflict of interest

The authors declare no potential conflict of interests.

References

1. Hinton Geoffrey, Deng Li, Yu Dong, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine.* 2012;29(6):82–97.

2. Mikolov Tomáš, Deoras Anoop, Povey Daniel, Burget Lukáš, Černock\'y Jan. Strategies for training large scale neural network language models. In: :196–201Institute of Electrical and Electronics Engineers (IEEE); 2011.

3. Lara Oscar D, Labrador Miguel A, Others . A survey on human activity recognition using wearable sensors.. *IEEE Communications Surveys and Tutorials.* 2013;15(3):1192–1209.

4. Plötz Thomas, Hammerla Nils Y, Olivier Patrick. Feature learning for activity recognition in ubiquitous computing. In: :1729; 2011.

5. Hinton Geoffrey E, Osindero Simon, Teh Yee-Whye. A fast learning algorithm for deep belief nets. *Neural computation.* 2006;18(7):1527–1554.

6. Zhang Licheng, Wu Xihong, Luo Dingsheng. Recognizing human activities from raw accelerometer data using deep neural networks. In: :865–870Institute of Electrical and Electronics Engineers (IEEE); 2015.

7. Alsheikh Mohammad Abu, Selim Ahmed, Niyato Dusit, Doyle Linda, Lin Shaowei, Tan Hwee-Pink. Deep Activity Recognition Models with Triaxial Accelerometers.. In: ; 2016.

8. Bhattacharya Sourav, Lane Nicholas D. From smart to deep: Robust activity recognition on smartwatches using deep learning. In: :1–6; 2016.

9. Radu Valentin, Lane Nicholas D, Bhattacharya Sourav, Mascolo Cecilia, Marina Mahesh K, Kawsar Fahim. Towards multimodal deep learning for activity recognition on mobile devices. In: :185–188ACM; 2016.

10. Hassan Mohammed Mehedi, Uddin Md Zia, Mohamed Amr, Almogren Ahmad. A robust human activity recognition system using smartphone sensors and deep learning. *Future Generation Computer Systems.* 2018;81:307–313.

11. LeCun Yann, Boser Bernhard, Denker John S, et al. Backpropagation applied to handwritten zip code recognition. *Neural computation.* 1989;1(4):541–551.

12. Duffner Stefan, Berlemont Samuel, Lefebvre Grégoire, Garcia Christophe. 3D gesture classification with convolutional neural networks. In: :5432–5436Institute of Electrical and Electronics Engineers (IEEE); 2014.

13. Zeng Ming, Nguyen Le T, Yu Bo, et al. Convolutional neural networks for human activity recognition using mobile sensors. In: :197–205Institute of Electrical and Electronics Engineers (IEEE); 2014.

14. Yang Jianbo, Nguyen Minh Nhut, San Phyo Phyo, Li Xiaoli, Krishnaswamy Shonali. Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition.. In: :3995–4001; 2015.

15. Chen Yuqing, Xue Yang. A deep learning approach to human activity recognition based on single accelerometer. In: :1488–1492Institute of Electrical and Electronics Engineers (IEEE); 2015.

16. Jiang Wenchao, Yin Zhaozheng. Human activity recognition using wearable sensors by deep convolutional neural networks. In: :1307–1310ACM; 2015.

17. Ronao Charissa Ann, Cho Sung-Bae. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications.* 2016;59:235–244.

18. Zebin Tahmina, Scully Patricia J, Ozanyan Krikor B. Human activity recognition with inertial sensors using a deep learning approach. In: :1–3Institute of Electrical and Electronics Engineers (IEEE); 2016.

19. Camps Julià, Samà Albert, Mart\'\in Mario, et al. Deep learning for freezing of gait detection in Parkinson's disease patients in their homes using a waist-worn inertial measurement unit. *Knowledge-Based Systems.* 2018;139:119–131.

20. Moya Rueda Fernando, Grzeszick René, Fink Gernot A, Feldhorst Sascha, Ten Hompel Michael. Convolutional neural networks for human activity recognition using body-worn sensors. In: :26Multidisciplinary Digital Publishing Institute; 2018.

21. Lefebvre Grégoire, Berlemont Samuel, Mamalet Franck, Garcia Christophe. BLSTM-RNN based 3D gesture classification. In: :381–388Springer; 2013.

22. Ordóñez Francisco Javier, Roggen Daniel. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors.* 2016;16(1):115.

23. Guan Yu, Plötz Thomas. Ensembles of deep lstm learners for activity recognition using wearables. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT'17).* 2017;1(2):11.

24. Murad Abdulmajid, Pyun Jae-Young. Deep recurrent neural networks for human activity recognition. *Sensors.* 2017;17(11):2556.

25. Rumelhart David E, Hinton Geoffrey E, Williams Ronald J. Learning representations by back-propagating errors. *nature.* 1986;323(6088):533.

26. Hochreiter Sepp, Schmidhuber Jürgen. Long short-term memory. *Neural computation.* 1997;9(8):1735–1780.

27. Chung Junyoung, Gulcehre Caglar, Cho KyungHyun, Bengio Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555.* 2014;.

28. Hammerla Nils Y, Halloran Shane, Ploetz Thomas. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880.* 2016;.

29. Wu Wanmin, Dasgupta Sanjoy, Ramirez Ernesto E, Peterson Carlyn, Norman Gregory J. Classification accuracies of physical activities using smartphone motion sensors. *Journal of medical Internet research.* 2012;14(5).

30. Shoaib Muhammad, Bosch Stephan, Incel Ozlem Durmaz, Scholten Hans, Havinga Paul J M. Fusion of smartphone motion sensors for physical activity recognition. *Sensors.* 2014;14(6):10146–10176.

31. Gjoreski Hristijan, Bizjak Jani, Gjoreski Martin, Gams Matjaž. Comparing deep and classical machine learning methods for human activity recognition using wrist accelerometer. In: ; 2016.

32. Ha Sojeong, Yun Jeong-Min, Choi Seungjin. Multi-modal convolutional neural networks for activity recognition. In: :3017–3022Institute of Electrical and Electronics Engineers (IEEE); 2015.

33. Ravi Daniele, Wong Charence, Lo Benny, Yang G. A deep learning approach to on-node sensor data analytics for mobile or wearable devices. *IEEE Journal of Biomedical and Health Informatics.* 2016;.

34. Erhan Dumitru, Bengio Yoshua, Courville Aaron, Manzagol Pierre-Antoine, Vincent Pascal, Bengio Samy. Why does unsupervised pre-training help deep learning?. *Journal of Machine Learning Research.* 2010;11(Feb):625–660.

35. Goodfellow Ian, Bengio Yoshua, Courville Aaron. *Deep Learning*. MIT Press; 2016.

36. LeCun Yann, Bengio Yoshua, Hinton Geoffrey. Deep learning. *nature.* 2015;521(7553):436.

37. Jordan Michael I. Serial order: A parallel distributed processing approach. In: Elsevier 1997 (pp. 471–495).

38. Robinson A J, Fallside Frank. *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering; 1987.

39. Werbos Paul J. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks.* 1988;1(4):339–356.

40. Mozer Michael C. A focused backpropagation algorithm for temporal. *Backpropagation: Theory, architectures, and applications.* 1995;:137.

41. Bengio Yoshua, Simard Patrice, Frasconi Paolo. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks.* 1994;5(2):157–166.

42. Pascanu Razvan, Mikolov Tomas, Bengio Yoshua. On the difficulty of training recurrent neural networks. In: :1310–1318; 2013.

43. Bao Ling, Intille Stephen S. Activity recognition from user-annotated acceleration data. In: :1–17Springer; 2004.

44. Preece Stephen J, Goulermas John Yannis, Kenney Laurence P J, Howard David. A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data. *IEEE Transactions on Biomedical Engineering.* 2009;56(3):871–879.

45. Ravi Daniele, Wong Charence, Lo Benny, Yang Guang-Zhong. Deep learning for human activity recognition: A resource efficient implementation on low-power devices. In: :71–76Institute of Electrical and Electronics Engineers (IEEE); 2016.

46. Anguita Davide, Ghio Alessandro, Oneto Luca, Parra Xavier, Reyes-Ortiz Jorge Luis. A Public Domain Dataset for Human Activity Recognition using Smartphones.. In: ; 2013.

47. Stisen Allan, Blunck Henrik, Bhattacharya Sourav, et al. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In: :127–140ACM; 2015.

48. Banos Oresti, Garcia Rafael, Holgado-Terriza Juan A, et al. mHealthDroid: a novel framework for agile development of mobile health applications. In: :91–98Springer; 2014.

49. Roggen Daniel, Calatroni Alberto, Rossi Mirco, et al. Collecting complex activity datasets in highly rich networked sensor environments. In: :233–240Institute of Electrical and Electronics Engineers (IEEE); 2010.

50. Reiss Attila, Stricker Didier. Introducing a new benchmarked dataset for activity monitoring. In: :108–109Institute of Electrical and Electronics Engineers (IEEE); 2012.

51. Reiss Attila, Stricker Didier. Creating and benchmarking a new dataset for physical activity monitoring. In: :40ACM; 2012.

52. Shoaib Muhammad, Scholten Hans, Havinga Paul J M. Towards physical activity recognition using smartphone sensors. In: :80–87Institute of Electrical and Electronics Engineers (IEEE); 2013.

53. Zhang Mi, Sawchuk Alexander A. USC-HAD: a daily activity dataset for ubiquitous activity recognition using wearable sensors. In: :1036–1043ACM; 2012.

54. Sztyler Timo, Stuckenschmidt Heiner. On-body localization of wearable devices: An investigation of position-aware activity recognition. In: :1–9Institute of Electrical and Electronics Engineers (IEEE); 2016.

55. Srivastava Nitish, Hinton Geoffrey, Krizhevsky Alex, Sutskever Ilya, Salakhutdinov Ruslan. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research.* 2014;15:1929–1958.

56. Plaut David C, Others . *Experiments on Learning by Back Propagation..* : Computer Science Department, Carnegie-Mellon University; 1986.

57. Finnoff William, Hergert Ferdinand, Zimmermann Hans Georg. Improving model selection by nonconvergent methods. *Neural Networks.* 1993;6(6):771–783.

58. Duchi John, Hazan Elad, Singer Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research.* 2011;12(Jul):2121–2159.

59. Bhattacharya Sourav, Nurmi Petteri, Hammerla Nils, Plötz Thomas. Using unlabeled data in a sparse-coding framework for human activity recognition. *Pervasive and Mobile Computing.* 2014;15:242–262.

60. Neyshabur Behnam, Bhojanapalli Srinadh, McAllester David, Srebro Nati. Exploring generalization in deep learning. In: :5947–5956; 2017.

61. Pan Sinno Jialin, Yang Qiang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering.* 2010;22(10):1345–1359.

How to cite this article: Sansano E., R. Montoliu, and O. Belmonte (2019), A study of Deep Neural Networks for Human Activity Recognition, *Computational Intelligence*, *2019;00:1–6*.