IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 9, 2021, accepted September 28, 2021, date of publication October 4, 2021, date of current version October 18, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3117336

Human Activity Recognition Based on Acceleration Data From Smartphones Using HMMs

SYLVAIN ILOGA[1,2,3], ALEXANDRE BORDAT[2,4], JULIEN LE KERNEC[2,5], (Senior Member, IEEE), AND OLIVIER ROMAIN[2], (Member, IEEE)

[1]Department of Computer Science, Higher Teachers' Training College, University of Maroua, Maroua, Cameroon
[2]ETIS UMR 8051, CNRS, ENSEA, CY Cergy Paris University, 95000 Cergy, France
[3]IRD, UMMISCO, University of Sorbonne, 93143 Bondy, France
[4]Bluelinea, 78990 Élancourt, France
[5]James Watt School of Engineering, University of Glasgow, Glasgow G12 8QQ, U.K.

Corresponding author: Sylvain Iloga (sylvain.iloga@gmail.com)

This work was supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/R041679/1 (INSHEP).

ABSTRACT Smartphones are among the most popular wearable devices to monitor human activities. Several existing methods for Human Activity Recognition (HAR) using data from smartphones are based on conventional pattern recognition techniques, but they generate handcrafted feature vectors. This drawback is overcome by deep learning techniques which unfortunately require lots of computing resources, while generating less interpretable feature vectors. The current paper addresses these limitations through the proposal of a Hidden Markov Model (HMM)-based technique for HAR. More formally, the sequential variations of spatial locations within the raw data vectors are initially captured in Markov chains, which are later used for the initialization and the training of HMMs. Meta-data extracted from these models are then saved as the components of the feature vectors. The meta-data are related to the overall time spent by the model observing every symbol for a long time span, irrespective of the state from which this symbol is observed. Classification experiments involving four classification tasks have been carried out on the recently constructed *UniMiB SHAR* database which contains 17 classes, including 9 types of activities of daily living and 8 types of falls. As a result, the proposed approach has shown best accuracies between 92% and 98.85% for all the classification tasks. This performance is more than 10% better than prior work for 2 out of 4 classification tasks.

INDEX TERMS Human activity recognition, activities of daily living, fall detection, hidden Markov models, smartphone sensors.

I. INTRODUCTION

Human Activity Recognition (HAR) has gained in importance for many decades for its capability to learn meaningful and high-level knowledge about various types of human activities including (but not limited to):

1) *Ambulation*: walking, running, climbing stairs, etc.
2) *Activities of daily living* (ADL): eating, drinking, reading, etc.
3) *Falls*: fall forward, fall backward, syncope, etc.

More detailed descriptions of the existing types of human activities are available in [1][1] and [2].[2] A review of state-of-the-art techniques for abnormal HAR is also proposed in [3]. There are two main categories of HAR. These are *video-based* HAR and *sensor-based* HAR. Video-based HAR performs high-level analysis of videos or images containing human motions from cameras. No further details related to this category of HAR are provided in the current paper which rather focuses on the second category.

The associate editor coordinating the review of this manuscript and approving it for publication was Yue Zhang.

[1]See Table 1 of [1]
[2]See Table 3 of [2]

Nevertheless, relevant surveys on video-based HAR are available in [4]–[6]. Sensor-based HAR is more popular and widely used because it better preserves privacy than video-based HAR. It relies on motion data from several types of smart sensors including:

1) *Body-worn sensors*: These sensors are worn by the user to describe the body movements. They are generally embedded in smartphones, watchs, standalone devices which include sensors such as accelerometers, gyroscopes, etc.
2) *Object sensors*: They are attached to objects to capture object movements. Radio frequency identifiers (RFID) deployed in smart home environment or accelerometer fixed on objects (e.g: glass, cup) are generally used for this purpose.
3) *Ambient sensors*: They are used to capture the interaction between humans and the environment in a smart environment. There are many kinds of ambient sensors such as radars, microphones, pressure sensors, temperature sensors, WiFi, Bluetooth, etc.
4) *Hybrid sensors*: Here, the three former types of sensors are combined. Further details are available in [7].

Besides HAR, the aforementioned sensors are also adapted for several other topics including indoor positioning methods [8] and pedestrian dead reckoning [9], [10]. Detailed presentations of these types of sensors and related papers are provided in [11].[3] When the experiments are performed in smart homes with a variety of sensors, the HAR data processing needs to be distributed over a group of heterogeneous, autonomous and interacting entities in order to be more efficient. An efficient multi-agent approach for HAR in this context has recently been proposed in [12].

HAR can be treated as a typical pattern recognition problem and existing papers in HAR can be organized into two main categories:

1) *Conventional pattern recognition techniques* [13]–[46] where the feature extraction and the model building steps are separated.
2) *Deep learning techniques* [43], [44], [47]–[88] where the features extraction and model building processes are performed simultaneously in the deep learning models.

Conventional pattern recognition techniques embed the following drawbacks analyzed in [11][4]:

1) The features are extracted via a handcrafted process that heavily relies on human experience and domain knowledge.
2) Only shallow features can be learned according to human expertise. Such shallow features can only be used for recognizing low-level activities (*walking*, *running*, etc.), but they can hardly enable to accurately infer complex activities like *having a coffee* for example.
3) These techniques often require a large amount of well-labeled data to train the model. However, most of the activity data are remaining unlabeled in real applications.

The aforementioned drawbacks are overcome by deep learning sensor-based solutions. However deep learning techniques embed the following limitations analyzed in [2][5]:

1) They require lots of computing resources.
2) The parameters of the resulting models are difficult to adjust.
3) The components of the resulting feature vectors are less interpretable. More details related to this limitation are available in [89] where an overview of explainable artificial intelligence for deep neural networks is proposed.

The current paper addresses these limitations through the proposal of a Hidden Markov Model (HMM)-based technique for HAR which derives interpretable feature vectors from the model's meta-data. The parameters of the resulting HMMs are understandable and can therefore be easily adjusted. Furthermore, the models' training time is reasonable compared to deep models. Raw data from tri-axial smartphones sensors are preferred here because studies demonstrated that samples from smartphones sensors (e.g., accelerometer and gyroscope) are accurate enough to be used in the clinical domain, such as ADLs recognition [23].

More precisely, given a signal window w, we first represent w as a sequence $w = w_1 \dots w_T$ of 3-dimensional vectors. The sequential variations of spatial locations within these 3-dimensional vectors are then captured to transform w into the Markov chain δ_w whose content later serves to fix the parameters of an initial HMM associated with w. These parameters are then iteratively adjusted at each iteration of the *Baum-Welch* algorithm to obtain the final HMM λ_w. Thereafter, meta-data derived from λ_w are saved as the components of the descriptor vector $\overrightarrow{w}$ associated with w. The performances of the proposed approach are evaluated through flat classification experiments on *UniMiB SHAR* [23] which is a database containing acceleration patterns captured by smartphones and constructed in 2017 for the objective evaluation of ADLs recognition and fall detection techniques.

The rest of this paper is organized as follows: The state of the art is presented in Section II, followed by a summarized presentation of HMMs in Section III. A detailed description of the approach proposed in this paper is given in Section IV. Experimental results are presented in Section V and the last section is devoted to the conclusion.

II. STATE OF THE ART

A. RELATED WORK

1) CONVENTIONAL PATTERN RECOGNITION TECHNIQUES

Conventional pattern recognition solutions for HAR generally rely on the following process depicted in Figure 1:

[3] See Section 3 of [11]
[4] See Section 2.2 of [11]
[5] See Table 5 of [2]

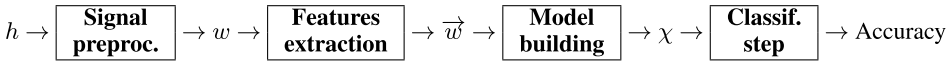

FIGURE 1. HAR using conventional pattern recognition techniques.

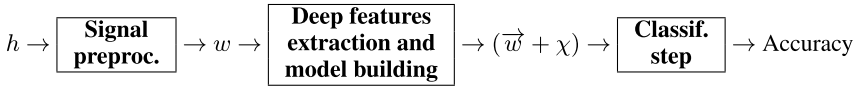

FIGURE 2. HAR using deep learning techniques.

1) **Signal preprocessing**: Here, sensing devices capture human motions and save them in raw data vectors. Small-time windows (a couple of seconds) of the input signal h are sequentially captured and sampled at a specific sampling frequency depending on the targeted application.
2) **Features extraction**: Handcrafted features are extracted from each raw data vector w associated with a given time window through the computation of informative statistics. More precisely, *time-domain features* (variance, mean, root mean square, zero-crossing rate, etc.), *frequency domain features* (Fast Fourier Transform, Discrete Cosine Transform, etc.) and *other features* (Principal Component Analysis, Linear Discriminant Analysis, etc.) are manually computed and saved as the components of the features vector $\overrightarrow{w}$ of each data vector w. More detailed lists of typical handcrafted features used in HAR are available in [1][6] and [2].[7]
3) **Model building**: Here, a conventional machine learning model (classifier) χ is trained to learn the characteristics of the human activities associated with the training feature vectors. The following models are most often used for this purpose:
   - *k Nearest Neighbor* (KNN) [13]–[25].
   - *Support Vector Machines* (SVMs) [16]–[20], [22], [23], [25]–[28].
   - *Decision trees* (DT) [13], [17], [19]–[22], [25], [29]–[34].
   - *Random Forest* (RF) [19], [22], [23].
   - *Multilayer perceptron* (MPL) [17], [21], [33], [35].
   - *Artificial Neural Networks* (ANN) [22], [23].
   - *Naive Bayes* and *Bayesian Networks* [13], [14], [17], [19], [22], [25], [31], [36].
   - *Fuzzy Inference System* (FIS) [14], [37]–[39].
   - *Boosting* and *Bagging* [40], [41].
   - *Hidden Markov models* (HMMs) [36], [42]–[46].
4) **Classification step**: The previously trained model (classifier) is now used for inferring the corresponding human activity. The *accuracy* is the most used metric for evaluating the performances of these classifiers. Other metrics like the *F1-measure*, the *Precision*, the *Recall* are also rarely used.

[6]See Table 2 of [1]
[7]See Table 4 of [2]

2) DEEP LEARNING TECHNIQUES

With deep learning techniques for HAR, the features extraction and model building processes are performed simultaneously in the deep learning models as shown in Figure 2. Here, the feature vectors are automatically learned through the network χ instead of being manually designed. A detailed study of deep neural networks for HAR is available in [90]. An evaluation framework allowing a rigorous comparison between handcrafted features and features generated by several deep models is proposed in [91]. The following deep models are most often used in HAR:

- *Deep Neural Networks* (DNN) [47]–[50].
- *Convolutional Neural Networks* (CNN) [49], [51]–[76].
- *Recurrent Neural Networks* (RNN) [49], [72], [76]–[80].
- *Deep belief network* (DBN) and *Restricted Boltzmann Machine* (RBM) [43], [44], [60], [81]–[87].
- *Stacked autoencoder* (SAE) [67], [88].
- *Hybrid models* [7], [49], [60], [67], [72].

Although handcrafted features are considered a drawback in HAR, these features can nevertheless enhance the performances of CNN in some face related problems including age/gender estimation, face detection and emotion recognition [92]. Handcrafted features have also been combined with CNN generated features for HAR [73].

3) PERFORMANCES OF EXISTING WORK

The implementation of a typical HAR system requires the design of a suitable dataset containing the raw data from which the signal windows will be derived. The datasets utilized in existing papers contain data related to human activities performed by various participants whose characteristics (gender, age, weight, height, etc.) vary from one dataset to the other. Some authors prefer private custom datasets [13]–[16], [26], [27], [29]–[32], [34], [37], [39]–[42], [45]–[48], [50], [53], [58], [60], [65], [66], [68]–[71], [77], [82], [84]–[86], but it is difficult to perform comparisons with these work in such conditions. Other authors adopt

TABLE 1. Performances of relevant existing work where publicly available datasets are experimented. The values of the *accuracy* and the *F1-measure* are in (%).

Category	N°	Authors[Ref]	Year	Classifier or Deep model	Dataset	# HA	Acc. or F1
Conventional pattern recognition techniques	1	Kwapisz et al. [33]	2011	DT, MLP	WISDM	6	91.7
	2	Anguita et al. [28]	2013	SVM	UCI-HAD	6	96
	3	Shoaib et al. [17]	2013	NB, SVM, MLP, KNN, DT	Shoaib PA	6	99
	4	Medrano et al. [18]	2014	KNN, SVM	tFall	8	95
	5	Shoaib et al. [19]	2014	NB, SVM, DT, RF, KNN	Shoaib SA	7	99.5
	6	Kabir et al. [36]	2016	HMM, NB	Kasteren-A	10	F1:70
					Kasteren-B	13	F1:69
					Kasteren-C	16	F1:69
	7	Reyes et al. [20]	2016	DT, SVM, KNN	SBHAR	7	95.8
					PAMAP2	12	94.33
					REALDISP	33	99.52
	8	Sztyler & Heiner [22]	2016	NB, SVM, DT RF, KNN, ANN	RealWorld(HAR)	8	89
	9	Vavoulas et al. [21]	2016	KNN, DT, MLP	MobiAct	13	99.88
	10	Walse et al. [35]	2016	MLP	UCI-HAD	6	98.11
	11	Micucci et al. [23]	2017	SVM, KNN RF, ANN	UniMiB SHAR	17	98.71
	12	Vavoulas et al. [24]	2017	KNN	MobiFall	13	94.65
	13	Santoyo et al. [25]	2018	NB, SVM, KNN, DT	UMA-Fall	14	85.83
Deep learning techniques	14	Plötz et al. [81]	2011	RBM	OPPORTUNITY	11	75.9
	15	Zeng et al. [51]	2014	CNN	OPPORTUNITY	11	76.83
					Actitracker	6	96.88
	16	Zheng et al. [52]	2014	CNN	PAMAP2	4	93.36
	17	Alsheikh et al. [43]	2015	RBM & HMM	WISDM	6	98.23
	18	Ha et al. [54]	2015	CNN	MHEALTH	12	98.29
	19	Hayashi et al. [83]	2015	RBM	HASC	13	91.7
	20	Jiang & Yin [55]	2015	CNN	SHO	7	99.93
					USC-HAD	11	97.01
					UCI-HAD	6	95.18
	21	Yang et al. [56]	2015	CNN	ActRecTul	12	96.0
					OPPORTUNITY	18	87.7
	22	Bhattacharya & Lane [86]	2016	RBM	OPPORTUNITY	4	72.1
	23	Chen et al. [57]	2016	CNN	WISDM	6	92.1
	24	Edel & Köppe [77]	2016	RNN	OPPORTUNITY	18	78
					PAMAP2	18	93
	25	Gjoreski et al. [58]	2016	CNN	OPPORTUNITY	4	67
	26	Ha & Choi [59]	2016	CNN	MHEALTH	12	99.66
	27	Hammerla et al. [49]	2016	DNN, CNN, RNN	OPPORTUNITY	18	F1: 92.7
					PAMAP2	12	F1: 93.7
	28	Morales & Roggen [61]	2016	CNN	OPPORTUNITY	17	F1: 60
	29	Radu et al. [87]	2016	RBM	HHAR	6	80
	30	Ravi et al. [62]	2016	CNN	ActiveMiles	7	95.1
					WISDM	6	98.2
	31	Ronao et al. [64]	2016	CNN	UCI-HAD	6	95.75
	32	Zheng et al. [67]	2016	CNN, SAE	PAMAP2	4	93.43
	33	Almaslukh et al. [88]	2017	SAE	UCI-HAD	6	97.5
	34	Guan & Plötz [79]	2017	RNN	OPPORTUNITY	18	F1: 72.6
					PAMAP2	12	F1: 85.4
	35	Murad & Pyun [78]	2017	RNN	OPPORTUNITY	18	F1: 92
					UCI-HAD	6	96.7
					USC-HAD	11	97.8
	36	Yao et al. [72]	2017	CNN, RNN	HHAR	6	94.2
	37	Dong et al. [73]	2018	CNN	UCI-HAD	6	96.9
	38	Moya et al. [74]	2018	CNN	OPPORTUNITY	18	92.44
					PAMAP2	12	92.55
	39	Xu et al. [80]	2019	RNN	OPPORTUNITY	18	94.6
					PAMAP2	18	93.5
					UCI-HAD	6	94.5
	40	Wan et al. [75]	2020	CNN	UCI-HAD	6	92.71
					PAMAP2	12	91.00
	41	Xia et al. [76]	2020	CNN, RNN	OPPORTUNITY	18	92.63
					UCI-HAD	6	95.78
					WISDM	6	95.85

TABLE 2. Performances of relevant existing work where custom datasets are experimented. Accuracies are in (%).

Category	N°	Authors[Ref]	Year	Classifier or Deep model	# HA	Acc.
Conventional pattern recognition techniques	1	Bao & Intille [13]	2004	NB, DT, KNN	20	84.26
	2	Maurer et al. [29]	2006	DT	6	92.8
	3	Parkka et al. [30]	2006	DT	7	86
	4	Minnen et al. [40]	2007	Boosting	14	90.3
	5	Tapia et al. [31]	2007	NB, DT	30	97.6
	6	Chen et al. [37]	2008	FIS	8	92.86
	7	Ermes & Parkka [32]	2008	DT	6	94
	8	He & Jin [26]	2008	SVM	4	92.25
	9	Jatoba et al. [14]	2008	NB, KNN, FIS	6	86.6
	10	Brezmes et al. [15]	2009	KNN	6	80
	11	He et al. [27]	2009	SVM	4	97.51
	12	Zhu & Sheng [42]	2009	HMM	4	89.74
	13	Altun & Barshan [16]	2010	SVM, KNN	19	98.8
	14	Berchtold et al. [39]	2010	FIS	10	97
	15	Lara et al. [41]	2012	Bagging & Boosting	5	95.7
	16	Lara et al. [34]	2012	DT	3	92.74
	17	Fallmann & Kropf [45]	2016	HMM	14	88.75
	18	Padar et al. [46]	2016	HMM	4	90
Deep learning techniques	19	Fang & Hu [82]	2014	RBM	3	93
	20	Chen & Xue [53]	2015	CNN	8	93.8
	21	Lane & Georgiev [84]	2015	RBM	14	73
	22	Vepakomma et al. [47]	2015	DNN	22	90
	23	Zhang et al. [48]	2015	DNN + HMM	5	93.52
	24	Zhang et al. [85]	2015	RBM	7	99.98
	25	Bhattacharya & Lane [86]	2016	RBM	3	93
	26	Edel & Köppe [77]	2016	RNN	14	83
	27	Gjoreski et al. [58]	2016	CNN	8	75.5
	28	Liu et al. [60]	2016	CNN, RBM	11	98.9
	29	Wang et al. [65]	2016	CNN	4	96.67
	30	Zebin et al. [66]	2016	CNN	6	97.01
	31	Kim & Li [68]	2017	CNN	6	98.8
	32	Lee et al. [69]	2017	CNN	3	92.71
	33	Mohammed & Tashev [70]	2017	CNN	18	90.48
	34	Panwar et al. [71]	2017	CNN	20	99.8
	35	Zhang et al. [50]	2017	DNN	11	98.86

the use of publicly available datasets to enable further comparisons [17]–[25], [28], [33], [35], [36], [43], [49], [51], [52], [54]–[59], [61], [62], [64], [67], [72], [74]–[81], [83], [86]–[88]. A deep survey of the evolution of modern datasets for HAR is available in [93]. The performances of a given HAR systems depend on several parameters including: the type of sensors, the number of participants, the protocol of data collection, the selected features and the selected models. Tables 1 and 2 present the performances of relevant existing work with experiments on publicly available and custom datasets. In these tables, the column entitled '*#HA*' contains the number of human activities involved in the considered work and the last column contains the *accuracy* of each reviewed work (except for [36], [49], [61], [79], [88] where the *F1-measure* is rather provided).

Among the datasets experimented by existing deep learning techniques listed in Table 1, *OPPORTUNITY*, *PAMAP2* and *UCI-HAD* are the most experimented datasets. However these datasets have not been considered in the current work for several reasons. *OPPORTUNITY* and *PAMAP2* were respectively designed with 4 and 9 participants, which are low values. Additionally, the subset of human activities selected by the authors for these two datasets varies from one work to the other. The dataset *UCI-HAD* was design with enough participants (30 participants) but it only enables us to identify 6 human activities. The advantages of the dataset *UniMiB SHAR* (selected here) compared to the other publicly available datasets listed in Table 1 are thoroughly analyzed in [23].[8] A summarized description of the UniMiB SHAR datatset is given in Section V-A.

B. PROBLEM STATEMENT

Existing approaches for HAR rely on conventional pattern recognition techniques or deep learning techniques for deriving feature vectors from the raw data acquired by diverse sensors. These feature vectors are used for classification purposes. Conventional pattern recognition techniques only enable to learn shallow features extracted via handcrafted processes and require a large amount of well-labeled data. Deep learning techniques for HAR overcome these

[8]Cf. Sections 1 and 2 of [23]

FIGURE 3. HAR using the proposed HMM-based learning technique.

drawbacks, but they require lots of computing resources and they generate less interpretable feature vectors. Additionally, it is challenging to adjust the parameters of the resulting deep models.

The current paper attempts to provide a solution to these limitations. This solution is based on the following observation: *Every human activity is a sequential process; hence it embeds a natural temporality that is meaningful for its characterization*. For this reason, human activity is generally captured at a precise sampling frequency by dedicated sensors which sequentially record several values in a raw data vector. Unfortunately, the natural temporality embedded in the raw data vectors is actually ignored during the computation of existing feature vectors. Our opinion is that, the sequential variations of spatial locations within each raw data vector enables the derivation of relevant feature vectors which may provide a better characterization of the considered human activity.

The current paper follows the principle presented in Figure 3 to analyze these sequential variations in order to generate one new feature vector $\overrightarrow{w}$ from each raw data vector w through a machine learning process. HMMs have been selected in this paper on the one hand because they are suitable for sequential data and their training time is reasonable compared to deep models. On the other hand, these models are managed by algorithms whose robustness and efficiency are widespread. HMMs have already been used in HAR systems as classifiers [36], [42], [45], [46]. They have also been combined with deep learning techniques [43], [44]. But they are used here in a different way and for a very different purpose (i.e: feature vectors are extracted from their meta-data).

III. PRESENTATION OF HMMs

A. HMM DEFINITION

A HMM $\lambda = (A, B, \pi)$ is fully characterized by [94]:

1) The number N of states of the model. The set of states is $S = \{s_1, s_2, \ldots, s_N\}$. The state of the model at time x is generally noted $q_x \in S$.
2) The number M of symbols. The set of symbols is $\vartheta = \{v_1, v_2, \ldots, v_M\}$. The symbol observed at time x is generally noted $o_x \in \vartheta$.
3) The state transition probability distribution $A = \{A[s_i, s_j]\}$ where $A[s_i, s_j] = Prob(q_{x+1} = s_j | q_x = s_i)$ with $1 \leq i, j \leq N$.
4) The symbols probabilities distributions $B = \{B[s_i, v_k]\}$ where $B[s_i, v_k] = Prob(v_k \text{ at time } x | q_x = s_i)$ with $1 \leq i \leq N$ and $1 \leq k \leq M$.

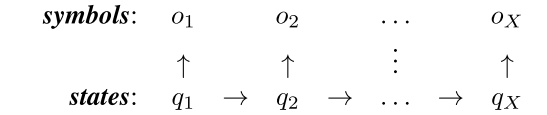

FIGURE 4. HMM used as sequence generator.

5) The initial state probability distribution $\pi = \{\pi[s_i]\}$ where $\pi[s_i] = Prob(q_1 = s_i)$ with $1 \leq i \leq N$.

B. HMM USED AS SEQUENCE GENERATOR

A HMM $\lambda = (A, B, \pi)$ can be used to generate a sequence $O = o_1 o_2 \ldots o_X$ composed of X symbols observed by the sequence of states $q = q_1 q_2 \ldots q_X$ as described in the *Markov chain* (MC) shown in Figure 4. In order to obtain the MC presented in Figure 4, the following algorithm is executed:

1) Select the initial state $s_j \in S$ according to the distribution π and set $x = 0$.
2) Set $x = x + 1$ and change the current state to $q_x = s_j$
3) Select the symbol $o_x \in \vartheta$ to be observed at state q_x according to the distributions in B.
4) If $(x < X)$ **go to** step 5, else **terminate**.
5) Select the state transition to be realized from the current state q_x to another state $s_j \in S$ according to the distribution A, then **go to** step 2.

C. MANIPULATION OF A HMMs

Consider a sequence of symbols $O = o_1 o_2 \ldots o_X$ and a HMM $\lambda = (A, B, \pi)$. The probability $Prob(O|\lambda)$ to observe O given λ is efficiently calculated by the *Forward-Backward* algorithm [94] which runs in $\theta(X.N^2)$. Given a sequence of symbols $O = o_1 o_2 \ldots o_X$, it is possible to iteratively re-estimate the parameters of a HMM $\lambda = (A, B, \pi)$ in order to maximize the value of $Prob(O|\overline{\lambda})$, where $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$ is the re-estimated model. The *Baum-Welch* algorithm [94] is generally used to perform this re-estimation. This algorithm runs in $\theta(\gamma.X.N^2)$ where γ is the user-defined maximum number of iterations. In this paper, the value $\gamma = 100$ is selected following [95].

D. STATIONARY DISTRIBUTION OF A HMM

A vector $\varphi = (\varphi[s_1], \ldots, \varphi[s_N])$ is a stationary distribution of a HMM $\lambda = \{A, B, \pi\}$ if:

1) $\sum_j \varphi[s_j] = 1$
2) $\forall j, \varphi[s_j] \geq 0$
3) $\varphi = \varphi.A \Leftrightarrow \left(\varphi[s_j] = \sum_i \varphi[s_i] \times A[s_i, s_j], \forall j\right)$

$\varphi[s_j]$ estimates the overall proportion of time spent by λ in state s_j over a long time span. φ can be extracted from any line

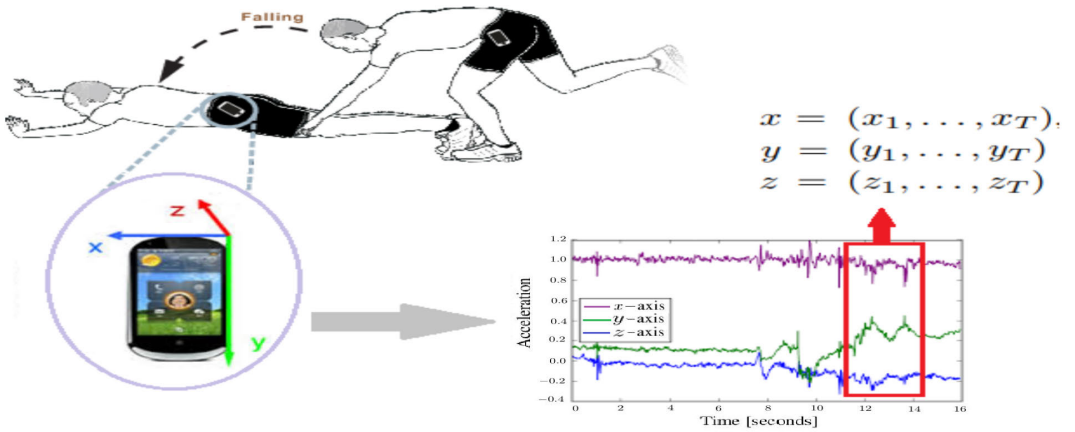

FIGURE 5. Extraction of the 3 data vectors x, y and z from the raw data generated by the triaxial accelerometer of a smartphone located inside subject's waist pocket during a fall.

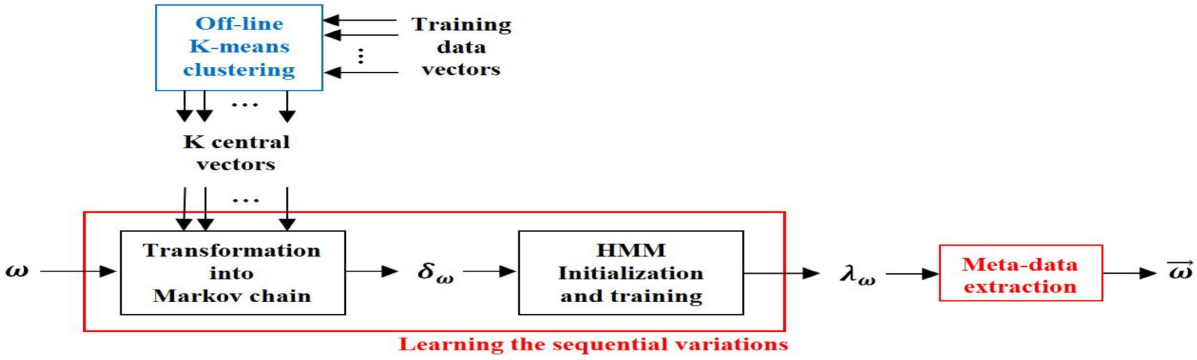

FIGURE 6. Methodology for deriving the features vector $\vec{w}$ associated with the signal windows w.

of the matrix $A^r = A \times A \times \ldots \times A$ (r times) when $r \to +\infty$. Therefore, the computation of φ requires $\theta(r.N^3)$ arithmetic operations.

IV. THE PROPOSED APPROACH

A. MAIN IDEA

Given a human activity, we assume that the experimental data are recorded by a triaxial smartphone accelerometer which generates 3 vectors $x = (x_1, \ldots, x_T)$, $y = (y_1, \ldots, y_T)$ and $z = (z_1, \ldots, z_T)$ for each signal window w. These vectors are respectively obtained after sampling the signal on each Cartesian axis at a unique sampling frequency. Figure 5 depicts this process for fall detection where the smartphone is located inside subject's waist pocket. In that figure, a signal window extracted from the raw data generated by the smartphone triaxial accelerometer is sampled on each Cartesian axis to derive the 3 data vectors x, y and z.

Existing techniques for HAR generally concatenate these 3 raw data vectors to obtain a unique vector $w = (x_1, \ldots, x_T, y_1, \ldots, y_T, z_1, \ldots, z_T)$. All the $(3T)$-dimensional raw data vectors resulting from the application of this principle on all the signal windows of the database are then used for handcrafted feature extraction or for deep feature extraction.

$$\begin{array}{rcccccc} x & = & (& \boxed{x_1} & , & \boxed{x_2} & , \ \ldots \ , & \boxed{x_T} &) \\ y & = & (& \boxed{y_1} & , & \boxed{y_2} & , \ \ldots \ , & \boxed{y_T} &) \\ z & = & (& \boxed{z_1} & , & \boxed{z_2} & , \ \ldots \ , & \boxed{z_T} &) \\ & & & \downarrow & & \downarrow & & \downarrow & \\ w & = & & w_1 & & w_2 & \ldots & w_T & \end{array}$$

FIGURE 7. Proposed representation of a signal window $w = w_1 \ldots w_T$ derived from the raw data vectors.

The main idea of this work is related to the fact that the former raw data vectors x, y and z can also be viewed as one sequence $w = w_1 w_2 \ldots w_T$ composed of 3-dimensional data vectors as depicted in Figure 7 where each $w_i = (x_i, y_i, z_i)$ with $(1 \leq i \leq T)$. Hence, the *sequential variations of the spatial locations* within the T vectors composing w can be captured into a MC δ_w which can later be used to initialize and train a dedicated HMM λ_w. Thereafter, one single feature vector $\vec{w}$ derived from the model's meta-data can finally be associated with w for classification purposes.

B. METHODOLOGY

As it is summarized in Figure 6, the proposed methodology for deriving the feature vector $\vec{w}$ associated with the

sequence $w = w_1 w_2 \ldots w_T$ is composed of the three following steps:

1) The spatial locations of the T data vectors composing the input sequence $w = w_1 w_2 \ldots w_T$ are captured by transforming w into the MC δ_w through a calculation involving each w_i and the K central vectors derived from the off-line *K-means* clustering of the training data vectors. More explanations about this step are provided in Section IV-C.
2) The content of δ_w is used for initializing a HMM which is then trained using the *Baum-Welch* algorithm to learn the sequential variations occurring inside δ_w (and consequently, inside w). The resulting model is λ_w. Section IV-D is devoted to the presentation of this step.
3) Meta-data are finally extracted from λ_w to derive the features vector $\overrightarrow{w}$. More precisely, $\overrightarrow{w}$ has M components, where M is the number of symbols of λ_w. The k^{th} component $\overline{w}_k$ of $\overrightarrow{w}$ being the overall proportion of time spent by λ_w observing the symbol $v_k \in \vartheta$ over a long time span, irrespective of the state from which v_k is observed. This step is fully presented in Section IV-E.

C. TRANSFORMATION INTO MARKOV CHAIN

As shown in Figure 4, a MC is composed of symbols and states, both belonging to finite sets. In order to transform a signal window into a MC, these two finite sets must be defined.

To determine the set of symbols, we are initially going to cluster the data vectors derived from all the signal windows found in the training database. More formally, let $H = \{H_1, \ldots, H_n\}$ be the set of activities found in the training database, each activity being represented by $|H_j|$ signal windows, with $(1 \leq j \leq n)$. Given that each signal window w is now considered as a sequence $w_1 \ldots w_T$ of 3-dimensional data vectors, the experimental database becomes a collection composed of $T \times \left(\sum_{j=1}^{n} |H_j|\right)$ data vectors. The *k-means* clustering algorithm [96] is then executed off-line for organizing this collection of training data vectors into K clusters, where K is a positive user-defined integer. The resulting set $\vartheta = \{v_1, \ldots, v_K\}$ of clusters is finally considered as the set of symbols of the model in such a way that all the vectors found inside a given cluster are associated with the same symbol. If we note $v(w_i)$ the cluster containing the data vector w_i, then the signal window $w = w_1 \ldots w_T$ is associated with the sequence of symbols $v(w_1) \ldots v(w_T)$. The *k-means* clustering algorithm is preferred in this work due its simplicity of implementation and the quality of its resulting clusters.

To determine the set of states, we focus on the spatial locations of the data vectors inside each cluster. More formally, consider a data vector w_i and let $\tilde{v}(w_i)$ be the center vector of cluster $v(w_i)$. We first evaluate the distance between w_i and $\tilde{v}(w_i)$, then we compare the resulting distance to the highest distance between any data vector of cluster $v(w_i)$ and $\tilde{v}(w_i)$. This comparison leads to the computation of a percentage $\alpha(w_i)$ which spatially characterizes each data vector w_i inside its cluster. Given a selected distance measure *dist* between vectors, the computation scheme of $\alpha(w_i)$ is shown in (1).

$$\begin{aligned} \alpha(w_i) &= 100 \times (U/V) \text{ in } (\%) \quad \text{where} \\ U &= dist(w_i, \tilde{v}(w_i)) \quad \text{and} \\ V &= \max\{dist(w_j, \tilde{v}(w_i)), \forall w_j \in v(w_i)\} \end{aligned} \tag{1}$$

Our objective is to consider all the possible values of $\alpha(w_i)$ as the set of states. In these conditions, Figure 8 depicts the resulting '*pseudo*' MC $\overline{\delta}_w$ associated with $w = w_1 w_2 \ldots w_T$.

clusters:	$v(w_1)$		$v(w_2)$		$\ldots$		$v(w_T)$
	$\uparrow$		$\uparrow$		$\vdots$		$\uparrow$
locations:	$\alpha(w_1)$	$\rightarrow$	$\alpha(w_2)$	$\rightarrow$	$\ldots$	$\rightarrow$	$\alpha(w_T)$

FIGURE 8. Pseudo MC $\overline{\delta}_w$ associated with w.

symbols:	$v(w_1)$		$v(w_2)$		$\ldots$		$v(w_T)$
	$\uparrow$		$\uparrow$		$\vdots$		$\uparrow$
states:	$\beta(w_1)$	$\rightarrow$	$\beta(w_2)$	$\rightarrow$	$\ldots$	$\rightarrow$	$\beta(w_T)$

FIGURE 9. MC δ_w associated with w.

$\overline{\delta}_w$ is not a valid MC because the spatial locations can take any value in the continuous interval [0, 100]. However, the states of a MC must always belong to a finite set. To overcome this limitation, the interval [0, 100] is first split into $(m+1)$ slices $\{s_0, s_1, \ldots, s_m\}$ following [97][9] as shown in (2), where m is a user-defined integer.

$$s_0 = \{0\} \text{ and } s_j = \left]\frac{100}{m} \times (j-1), \frac{100}{m} \times j\right], \quad (1 \leq j \leq m) \tag{2}$$

If the value of m is very high, the width of each slice s_j becomes tiny and all the elements in s_j converge to one unique value which is $\frac{100}{m} \times j$. In that case, the elements of s_j can be approximated by this single value which we identify here by the index j of slice s_j. In these conditions, the finite set $\{s_0, s_1, \ldots, s_m\}$ of slices can be considered here as the set of states. This reasoning enables defining the valid MC δ_w associated with $w = w_1 w_2 \ldots w_T$ by replacing every $\alpha(w_i)$ appearing in $\overline{\delta}_w$ by the value $\beta(w_i)$ which is the index j of the slice s_j containing $\alpha(w_i)$ as shown in (3). Figure 9 shows the resulting MC.

$$(\beta(w_i) = j) \Leftrightarrow \left(\alpha(w_i) \in s_j\right), \quad (0 \leq j \leq m) \tag{3}$$

Proceeding this way, δ_w effectively embeds information related to the sequential variations of spatial locations within w because:

1) The sequence $v(w_1) \ldots v(w_T)$ of symbols embeds information related to the sequential variations of clusters within w.

[9] See Equation 7 of [97]

2) The corresponding sequence $\beta(w_1)\ldots\beta(w_T)$ of states embeds information related to the sequential variations of spatial locations inside the various clusters within w.

Consequently, if a HMM λ_w is initialized and trained according to the content of δ_w, this model will learn all the information related to these sequential variations.

D. HMM INITIALIZATION AND TRAINING

1) DESIGN OF THE INITIAL HMM

Given a positive user-defined constant ε, the parameters of each the initial HMM λ_w^0 associated with w, are set to statistically capture the state transitions and the symbols probability distributions from the content of δ_w as follows:

1) The set of symbols is the set $\vartheta = \{v_1, \ldots, v_K\}$ of clusters generated by the *k-means* clustering algorithm, the number $K = M$ of clusters (symbols) being user-defined.
2) The set of states is $S = \{s_0, s_1, \ldots, s_m\}$ whose content is computed in (2) where m is the user-defined number of slices used to split the interval [0, 100]. Consequently, the number of states is $N = m + 1$.
3) The probability of transiting from state s_j to state s_k is calculated in (4), where $transit(s_j, s_k, \delta_w)$ is the number of transitions from state s_j to state s_k in δ_w and $transit(s_j, -, \delta_w)$ is the number of transitions from state s_j to any destination in δ_w.

$$A_w^0[s_j, s_k] = \frac{transit(s_j, s_k, \delta_w)}{transit(s_j, -, \delta_w) + \varepsilon} \tag{4}$$

4) The probability to observe symbol v_k at state s_j is calculated in (5), where $observe(v_k, s_j, \delta_w)$ is the number of times symbol v_k is observed at state s_j in δ_w, and $observe(-, s_j, \delta_w)$ is the number of occurrences of state s_j in δ_w.

$$B_w^0[s_j, v_k] = \frac{observe(v_k, s_j, \delta_w)}{observe(-, s_j, \delta_w) + \varepsilon} \tag{5}$$

5) The probability that the observation starts with state s_j is calculated in (6), where $start(s_j, \delta_w) = 1$ if δ_w starts with state s_j, 0 otherwise.

$$\pi_w^0[s_j] = \frac{start(s_j, \delta_w)}{1 + \varepsilon} \tag{6}$$

2) READJUSTMENT OF THE INITIAL HMM

The parameters of λ_w^0 are not probability distributions. This inconvenience is intentionally introduced by adding ε to the denominators of its various components in order to avoid eventual divisions by zero and zero probabilities. In this work, we experimentally fixed $\varepsilon = 1$. An equitable redistribution of the missing quantity is applied to each element of each line in $\lambda_w^0 = (A_w^0, B_w^0, \pi_w^0)$ to obtain the readjusted initial model $\lambda_w^1 = (A_w^1, B_w^1, \pi_w^1)$ whose parameters are:

1) $A_w^1[s_j, s_k] = A_w^0[s_j, s_k] + \frac{1}{m+1}\left(1 - \sum_{l=0}^{m} A_w^0[s_j, s_l]\right)$
2) $B_w^1[s_j, v_k] = B_w^0[s_j, v_k] + \frac{1}{K}\left(1 - \sum_{l=1}^{K} B_w^0[s_j, v_l]\right)$
3) $\pi_w^1[s_j] = \pi_w^0[s_j] + \frac{1}{m+1}\left(1 - \sum_{l=0}^{m} \pi_w^0[s_l]\right)$

3) TRAINING OF THE HMM

The readjusted initial HMM λ_w^1 is trained to learn the sequential variations occurring inside δ_w using the *Baum-Welch* algorithm. The resulting HMM λ_w is the final model associated with w. During this training phase, the training sequences are exclusively composed of symbols appearing in δ_w.

E. FEATURES VECTOR COMPUTATION

The features vector $\overrightarrow{w} = (\overline{w}_0, \overline{w}_1, \ldots, \overline{w}_m)$ associated with w is finally derived from $\lambda_w = (A_w, B_w, \pi_w)$ by analyzing the behavior of λ_w regarding each symbol v_k. More precisely, we propose to consider $\overline{w}_k$ as the overall proportion of time spent by λ_w at observing symbol v_k over the long term, irrespective of the state from which this observation is realized. In order to compute $\overline{w}_k$, one must first evaluate the overall proportion of time spent by λ_w at observing v_k in each state s_i over the long term as follows:

1) Evaluate the overall proportion of time spent by λ_w in state s_i over the long term. This proportion is given by the i^{th} component $\varphi_w[s_i]$ of the stationary distribution of λ_w.
2) Multiply the result obtained at step 1 by the probability of observing v_k in state s_i which is $B_w[s_i, v_k]$.

The value of $\overline{w}_k$ is finally obtained by repeating this process for every state s_i and summing the resulting proportions (7).

$$\begin{aligned} &\overrightarrow{w} = (\overline{w}_0, \overline{w}_1, \ldots, \overline{w}_m) \quad \text{where} \\ &\overline{w}_k = \sum_{i=1}^{K} (\varphi_w[s_i] \times B_w[s_i, v_k]) \text{ with } (0 \leq k \leq m) \end{aligned} \tag{7}$$

V. EXPERIMENTAL RESULTS

A. EXPERIMENTAL DATASET

Among the publicly available databases recorded with smartphones listed in Table 1, the dataset *UniMiB SHAR* has been selected in this work [23]. It is database of acceleration patterns measured by smartphones to be used as a common benchmark for the objective evaluation of both, ADLs recognition and fall detection techniques. This dataset contains 17 human activities including 9 different types of ADLs and 8 different types of falls. Table 3 presents the description of each ADL/fall.

During the construction of the selected database, human activities where performed by 30 subjects (including 24 females) between 18 and 60 years of age. Each ADL/fall type was performed twice by each subject: the first time with the smartphone in the right pocket and the second time with the smartphone in the left pocket. Signal windows of 3s each were saved during every experimental trial of a given ADL/fall performed by each subject. For each signal window w, the accelerometer recorded three data vectors (samples) x, y and z, each having $T = 151$ components. The database contains a total of 11.771 samples not equally distributed across activity types: 7.759 samples describing ADLs and 4.192 samples describing falls. Further details about the data acquisition, the experimental protocols, the characteristics of

TABLE 3. Descriptions of the 17 human activities of *UniMiB SHAR*.

Category	Label	Description
ADLs	StandingUpFL	From laying on the bed to standing
	LyingDownFS	From standing to lying on a bed
	StandingUpFS	From standing to sitting on a chair
	Running	Moderate running
	SittingDown	From standing to sitting on a chair
	GoingDownS	Climb the stairs moderately
	GoingUpS	Down the stairs moderately
	Walking	Normal walking
	Jumping	Continuous jumping
Falls	FallingBackSC	Fall backward while trying to sit on a chair
	FallingBack	Generic fall backward from standing
	FallingWithPS	Falls using compensation strategies to prevent the impact
	FallingForw	Fall forward from standing, use of hands to dampen fall
	FallingLeft	Fall left from standing
	FallingRight	Fall right from standing
	HittingObstacle	Falls with contact to an obstacle before hitting the ground
	Syncope	Getting unconscious

the subjects, the signal segmentation and the signal processing are available in [23].

B. EXPERIMENTAL SETTINGS

The classification experiments performed in this work were realized on a personal computer having 16 GB of main memory and the following processor: *Intel(R) Core(TM) i7-8665U CPU @ 1.90 GHz 2.11 GHz*. We evaluated four different classification tasks, following [23]:

1) **AF-17** which contains 17 classes (9 ADL classes and 8 FALL classes).
2) **A-9** which contains 9 ADL classes.
3) **F-8** contains 8 FALL classes.
4) **AF-2** which contains 2 classes obtained by considering all the ADLs as one class and all the FALLs as one class.

The *Euclidean* distance was selected in this work as the distance *dist* between two vectors required in (1). (2) was used to split the interval [0, 100] into 51 slices (i.e: we fixed $m = 50$) following [97] where the authors used analog reasoning to split the same interval to perform the comparison of finite sets of histograms using HMMs. Hence, the number of states of the HMMs designed in the current work is 51.

To analyze the impact of the user-defined number K of clusters discovered by the *k-means* clustering algorithm on the performances of the proposed approach, we experimented with the 5 following values of K: 20, 40, 60, 80 and 100. Consequently, the number of symbols of the HMMs designed in this work also varies accordingly. The first step of the machine learning process presented in Figure 6 and corresponding to the transformation of every signal w into the MC δ_w was entirely developed in Matlab. This choice was dictated by the fact that the database files were available as Matlab tables. Therefore, we executed a Matlab version of the *k-means* clustering algorithm during this step. Depending on the classification task and on the selected number of clusters, the off-line clustering step could sometimes take over an hour.

The two remaining steps of the proposed machine learning process were both developed in C language. Given a signal window w and its associated HMM $\lambda_w = (A_w, B_w, \pi_w)$, the stationary distribution φ_w of λ_w is obtained in this work by extracting the first line of the matrix $(A_w)^r$ with $r = 100$. After discovering the K clusters for each classification task, the computation of each feature vector $\overrightarrow{w}$ associated with w took between 50 and 3500 ms, depending on the content of the signal window. The overall time taken for the computation of the feature vectors associated with all the signal windows varied from one classification task to another depending on the considered number of signal windows. For each classification task in {AF-2, F-8, A-9, AF-17} and for each number of clusters in {20, 40, 60, 80, 100}, the resulting descriptor vectors have been saved into one online available '*arff*' files[10] which are taken as inputs by WEKA. The Matlab items (codes and tables) required to perform the off-line *k-means* clustering and the transformation into of Markov chains are also available through this same URL.

C. CLASSIFICATION PERFORMANCES

Classification experiments were realized with the soft WEKA [98] through 5-fold cross-validation following [23]. The following classifiers have been selected in this paper and their corresponding names in WEKA are shown in brackets:

1) *k-NN* (IBk) with $k = 1$, was used for the *Euclidean* and the *Manhattan* distances.
2) *SVMs* with polynomial kernel (SMO).
3) *Multilayer Perceptron* (MLP).
4) *Decision trees* (J48).
5) *Random Forest* (RF), these are bootstrap-aggregated decision trees with 300 bagged classification trees.

Table 4a presents the best classification accuracies for each classification task using the proposed feature vectors, irrespective of the number of clusters generating

[10]http://perso-etis.ensea.fr/sylvain.iloga/index.html

TABLE 4. Classification results. Accuracies are in (%). The best accuracies are in bold.

(a)- *Best performances*

	IBk		SMO	MLP	J48	RF
	Eucli	*Manha*				
AF-2	97.17	97.55	89.21	93.97	95.79	**98.85**
F-8	84.35	87.71	45.06	49.59	75.07	**92.00**
A-9	91.81	92.57	72.56	82.20	84.91	**95.28**
AF-17	87.05	89.61	53.58	62.88	79.39	**93.22**

(b)- *20 clusters*

	IBk		SMO	MLP	J48	RF
	Eucli	*Manha*				
AF-2	96.86	97.19	87.57	93.97	95.79	98.59
F-8	77.36	80.84	27.50	36.45	71.13	88.23
A-9	90.52	91.87	62.06	74.28	83.48	93.20
AF-17	84.16	86.22	46.30	57.24	75.88	90.50

(c)- *40 clusters*

	IBk		SMO	MLP	J48	RF
	Eucli	*Manha*				
AF-2	97.17	97.55	89.18	93.37	95.79	98.62
F-8	81.15	84.78	34.06	44.53	74.54	90.67
A-9	91.81	92.57	72.56	82.20	84.91	95.28
AF-17	87.05	89.61	50.92	61.86	79.39	93.22

(d)- *60 clusters*

	IBk		SMO	MLP	J48	RF
	Eucli	*Manha*				
AF-2	96.85	97.14	88.44	87.51	95.62	98.70
F-8	82.56	85.28	38.14	49.59	75.07	91.53
A-9	85.88	87.08	69.60	76.90	77.59	93.45
AF-17	85.10	87.26	53.58	62.88	77.02	92.77

(e)- *80 clusters*

	IBk		SMO	MLP	J48	RF
	Eucli	*Manha*				
AF-2	96.61	97.12	88.11	89.00	94.10	98.85
F-8	83.42	87.11	40.17	34.42	74.57	92.00
A-9	81.39	82.99	64.34	61.24	68.53	92.71
AF-17	80.87	83.67	52.35	51.53	71.92	92.09

(f)- *100 clusters*

	IBk		SMO	MLP	J48	RF
	Eucli	*Manha*				
AF-2	96.23	96.53	89.21	77.66	93.60	98.63
F-8	84.35	87.71	45.06	29.46	74.83	91.81
A-9	76.54	78.79	56.73	42.68	58.20	91.66
AF-17	78.03	81.69	49.97	33.26	65.82	91.09

these accuracies. According to Table 4a, the selected classifiers can be organized in descending order of performances for all the classification tasks as follows: RF, IBk (*Manhattan*), IBk (*Euclidean*), J48, MLP and SMO. The content of Table 4a demonstrates the high quality of the proposed descriptor vectors derived from the proposed HMM-based learning process with best accuracies always above 75%, 84% and 92% for the J48, the IBk and the RF classifiers respectively. This table also reveals the unsatisfactory performances exhibited by the SMO and the MLP classifiers for the AF-17 classification task and the very poor performances of these same classifiers for the F-8 classification task. Detailed classification performances for each classification task are presented in Tables 4b to 4f respectively for 20, 40, 60, 80 and 100 clusters. According to these tables, the variations of the user-defined number K of clusters do not significantly influence the performances of the proposed technique. Indeed, the gaps between the classification accuracies for the 5 experimental values of K are low. Consequently, it is not worth selecting a high value of K. We therefore recommend low values between 20 and 40.

D. COMPARISONS WITH RELATED WORK

We have compared the best classification performances obtained in this paper with those obtained in [23] where the authors conducted classification experiments on the same database and for the same classification tasks, using the following classifiers:

1) *k-NN* with $k = 1$.
2) *SVMs* with a radial basis kernel.
3) *Artificial Neural Networks*.
4) *RF* with 300 bagged classification trees.

Comparison results presented in Table 5 reveal that the approach proposed in this paper always outperforms [23] with positive accuracy gains reaching +13.45% and +10.36% respectively for F-8 and AF-17.

TABLE 5. Comparisons with [23]. Accuracies are in (%). The best accuracies are in bold.

	Micucci et al. 2017	This work	Accuracy gains
AF-2	98.71	**98.85**	+0.14
F-8	78.55	**92.00**	+13.45
A-9	88.41	**95.28**	+6.87
AF-17	82.86	**93.22**	+10.36

E. TIME COST

1) THEORETICAL TIME COST

The main contribution of the current paper is the computation of the proposed feature vector $\overrightarrow{w}$ associated with an input sequence w of raw data vectors as it can be observed in Figure 6. This computation embeds an offline *k-means* clustering whose time cost is not considered in this evaluation because it is realized '*offline*'. The remaining steps of the computation of $\overrightarrow{w}$ are:

1) The transformation of w into the Markov chain δ_w (See Section IV-C).
2) The HMM initialization using the content of δ_w to obtain the initial model λ_w^0 (See Section IV-D1).
3) The readjustment of λ_w^0 to obtain λ_w^1 (See Section IV-D2).
4) The HMM training of λ_w^1 with the *Baum-Welch* algorithm to obtain the final model λ_w (See Section IV-D3).

5) The extraction of meta-data from λ_w to derive $\overrightarrow{w}$ (See Section IV-E).

The time cost of the 3 first steps was experimentally low and very tiny compare to the time cost of the two last steps. Consequently, only the HMM training and the meta-data extraction are time consuming. According to Section III-C, the HMM training phase runs in $\theta(\gamma.T.(m+1)^2)$. The main operation realized during the meta-data extraction is the computation of the stationary distribution of the HMM which runs in $\theta(r.(m+1)^3)$ as stated in Section III-D. Therefore, the overall time cost of our contribution is approximated by $\theta(r.(m+1)^3 + \gamma.T.(m+1)^2)$ where:

1) r is the user-defined number of matrix products needed to compute the stationary distribution. In this paper $r = 100$.
2) γ is the user-defined maximum number of iterations of the algorithm. In this paper $\gamma = 100$.
3) T is the number of vectors in the sequence w. In this paper $T = 151$.
4) m is the user-defined number of slices used to split the interval [0, 100]. In this paper $m = 50$.

This time cost can be further reduced by gradually reducing the values of parameters like r or γ without negatively impacting the classification results. If the stationary distribution is discovered after r_0 iterations with ($r_0 < r$), this stationary distribution will not change during $(r - r_0)$ remaining iterations. Similarly, if the *Baum-Welch* algorithm reaches its local optimum after γ_0 iterations with ($\gamma_0 < \gamma$), this local optimum will not change during the $(\gamma - \gamma_0)$ remaining iterations. The experimental value $T = 151$ cannot be modified because it was fixed during the design of the experimental database. Similarly, the experimental value $m = 50$ cannot be changed here because it was fixed after several experiments performed in [97].

2) EXPERIMENTAL TIME COST

In order to measure the speed-up of the two most time-consuming stages of the processing chain (i.e. the stages of HMM training and the meta-data extraction), we executed and benchmarked the program on two different architectures: a desktop and the *Nvidia JETSON TX2* [99] comparable in terms of hardware to an embedded platform.

The main characteristics of the experimental desktop are the following:

- *CPU: AMD Ryzen 5 5600X (6 Cores) with dedicated core frequency when used at 4649.877 MHz*
- *RAM: 32GB DDR4 3600MHz G SKill Trident Z Neo*
- *GPU: RTX 3080 Gaming X Trio MSI*
- *Motherboard: Asus ROG STRIX B550-F GAMING Wifi*
- *Power supply: RMX 850W Corsair (Certified 89% of efficiency)*

The System-On-a-Chip (SOC) used on the *Nvidia JETSON TX2* has in common 4 cores used on the SOC of the *Exynos 7 Octa (5433)* [100] clocked at 1.9 GHz which is used on the *Galaxy Note 4* and the *Galaxy Tab S2*. The experiment presented in this section has been performed using the 11.771 MCs obtained when the k-means algorithm is executed with $k = 20$ (our smallest experimental value of k).

The *Nvidia Jetson TX2* has 8 GB L128 bit DDR4 of main memory. It runs on L4T (Linux for Tegra, i.e. Linux Kernel 4.9) and has a Parker SOC consisting of:

1) A Pascal GPU at 256 Cuda cores (not used in this experiment).
2) One HMP (6 cores) including 2 Denver cores (custom core designed by Nvidia to run the ARMv8 ISA) and 4 Arm Cortex A57 cores (to run the ARMv8 ISA). All the cores are compatible with ISA ARMv8 which is the 64 bits architecture of Arm.

The *Nvidia Jetson TX2* has several power supply modes that influence the frequency of the different cores. The modes used during our tests were the '*Max-N*' mode which allows to reach 2 GHz on each core and the '*Max-P Core-All*' mode allowing each core to be used at 1.4 GHz (i.e. a trade-off between performance and energy consumption). During the current experiment, the program was executed on cores excluded from the Linux scheduler allowing to dedicate the core entirely to the program and thus, not to distort the performance measurements. The following units were selected to measure the performance of our program using the *Performance Monitoring Unit* and *Syscall*:

1) Milliseconds (ms)
2) Instructions per cycle (IPC)
3) Instructions per second (IPS)
4) Floating point operations per second (FLOPS)

Tables 6 and 7 present the performances of our program on the experimental devices. According to these tables, the best performance is obtained by the desktop with a mean time cost of less than 2 seconds. Nevertheless, the performance on *Nvidia Jetson TX2* is also very interesting with a mean time cost of around 12 seconds when the *Core Denver 2 GHz* is used.

TABLE 6. Time cost in (ms) on the experimental devices.

Device	Worst case	Best case	Mean
Core ARM 1.4 GHz	40749.07	882.29	23837.94
Core ARM 2 GHz	28424.88	612.08	16647.94
Core Denver 1.4 GHz	26586.49	457.24	16565.96
Core Denver 2 GHz	18494.03	300.81	11490.03
Desktop 4.6 GHz	1408.53	50.81	1228.7

TABLE 7. Means of IPC, IPS and FLOPS on the experimental devices.

Device	IPC	IPS	FLOPS
Core ARM 1.4 GHz	1.52	2150141967.96	3051739.08
Core ARM 2 GHz	1.52	3079014192.13	4358501.28
Core Denver 1.4 GHz	2.22	3122548664.66	3295845.56
Core Denver 2 GHz	2.23	4509673864.60	4751731.28
Desktop 4.6 GHz	2.15	9995826825.13	98253721.03

A wattmeter was additionally used for measuring the energy consumption in order to deduce the *Average Power*

TABLE 8. APC and FOM.

Device	APC	FOM
Desktop 4.6 GHz	120 W	147.444 J
Nvidia Jetson TX2	6 W	68.94018 J

Consumption (APC) of each experimental device. This enabled us to calculate the *Figure Of Merit* (FOM) of each device by multiplying the *Average Execution Time* (AET) by the APC as shown in Equation (8). According to the FOM presented in Table 8, the *Nvidia Jetson TX2* is 2.138 times more efficient than the *Desktop 4.6 GHz* for the execution of our program.

$$\text{Efficiency(Time,Power)} = (AET) \times (APC) \tag{8}$$

F. MAIN ASSETS

The technique proposed in this paper:

1) Considers the sequential variations of spatial locations inside the raw data vectors unlike existing techniques.
2) Uses HMMs for the extraction of feature vectors, unlike existing techniques which only use these same models during the classification step.
3) Generates feature vectors whose components are interpretable, unlike existing techniques generating handcrafted or less interpretable feature vectors.
4) Performs the feature vectors extraction in reasonable time compared to deep learning techniques.
5) Efficiently performs HAR and outperforms prior work for the selected database.
6) The algorithm has been demonstrated viable on embedded platform processors from 2014 mobile phones and tablets, namely, *ARM* and *Denver* cores clocked at 1.4GHz and 2GHz. This shows that current mobile phones and tablets would be much closer to the performances of a PC.

VI. CONCLUSION

This paper addresses the problem of HAR based on acceleration data from Smartphones. Existing approaches for this purpose either rely on conventional pattern recognition techniques or on deep learning techniques. Conventional pattern recognition techniques generate shallow handcrafted feature vectors which heavily rely on the human experience/expertise. Deep learning techniques are preferable, but they require lots of computing resources while generating less interpretable feature vectors. The current paper attempts to overcome these limitations by proposing an efficient HMM-based technique that generates interpretable feature vectors while requiring a reasonable time cost with a demonstrated feasibility of implementation on embedded processors, namely, *Denver* and *ARM* cores.

Four different classification tasks have been tested on the *UniMiB SHAR* dataset containing 17 human activities including 9 types of ADLs and 8 types of falls. Classification results have demonstrated the efficiency of the proposed approach with the best accuracies between 92% and 98.85% for all the classification tasks. This performance is more than 10% better than state of the art for two classification tasks.

The main contribution of the current work is the HMM-based sequential learning of the sample (raw data vector) w associated with a human activity. Meta-data extracted from the resulting HMM λ_w are then used for deriving the corresponding feature vector $\overrightarrow{w}$. Consequently, the number of samples in the experimental database does not impact the components of $\overrightarrow{w}$ since each sample in the database is handled individually, irrespective of the other samples in the database. For this reason, the proposed approach will still exhibit good classification results even for large scale databases. Only the overall computation time for all the samples in the database will increase in these conditions. Parallel computation of all the feature vectors in the data base can also be implemented to reduce this overall computation time.

The current work has a dual impact on further researches in HAR. Firstly, it has been theoretically and experimentally demonstrated that learning a human activity as a sequential process enhances the quality of the resulting feature vectors and consequently, induces better classification results. A lot of the research in HAR only considers discrete activities as opposed to activities in a continuum. The proposed method enables extracting salient features from a stream of data using HMM. Secondly, the proposed method is an advance towards real-time implementation since this method can be efficiently ported on embedded platforms. Indeed, the current work uses HMMs for generating the feature vectors and the resulting feature vectors exhibit good classification performance, even with a basic classifier like the *k-NN*. Given that efficient hardware implementations of the *Baum-Welch* [101] and the *k-NN* [102] algorithms on Field-Programmable Gate Array (FPGA) chips are available, this method can therefore be deployed using hardware platforms with a lower footprint than GPUs in terms of energy and using fewer resources on a FPGA due to the simplicity of the implementation compared to CNN or DNN approaches.

REFERENCES

[1] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1192–1209, 3rd Quart., 2013.
[2] Y. Wang, S. Cang, and H. Yu, "A survey on wearable sensor modality centred human activity recognition in health care," *Expert Syst. Appl.*, vol. 137, pp. 167–190, Dec. 2019.
[3] C. Dhiman and D. K. Vishwakarma, "A review of state-of-the-art techniques for abnormal human activity recognition," *Eng. Appl. Artif. Intell.*, vol. 77, pp. 21–45, Jan. 2018.
[4] S. Vishwakarma and A. Agrawal, "A survey on activity recognition and behavior understanding in video surveillance," *Vis. Comput.*, vol. 29, no. 10, pp. 983–1009, Oct. 2013.
[5] L. Onofri, P. Soda, M. Pechenizkiy, and G. Iannello, "A survey on using domain and contextual knowledge for human activity recognition in video streams," *Expert Syst. Appl.*, vol. 63, pp. 97–111, Nov. 2016.
[6] D. R. Beddiar, B. Nini, M. Sabokrou, and A. Hadid, "Vision-based human activity recognition: A survey," *Multimedia Tools Appl.*, vol. 79, nos. 41–42, pp. 30509–30555, Nov. 2020.

[7] V. Ghate, "Hybrid deep learning approaches for smartphone sensor-based human activity recognition," *Multimedia Tools Appl.*, vol. 2021, pp. 1–20, Feb. 2021.

[8] S. Xu, R. Chen, Y. Yu, G. Guo, and L. Huang, "Locating smartphones indoors using built-in sensors and Wi-Fi ranging with an enhanced particle filter," *IEEE Access*, vol. 7, pp. 95140–95153, 2019.

[9] H. Ju, S. Y. Park, and C. G. Park, "A smartphone-based pedestrian dead reckoning system with multiple virtual tracking for indoor navigation," *IEEE Sensors J.*, vol. 18, no. 16, pp. 6756–6764, Aug. 2018.

[10] S. Qiu, Z. Wang, H. Zhao, K. Qin, Z. Li, and H. Hu, "Inertial/magnetic sensors based pedestrian dead reckoning by means of multi-sensor fusion," *Inf. Fusion*, vol. 39, pp. 108–119, Jan. 2018.

[11] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *Pattern Recognit. Lett.*, vol. 119, pp. 3–11, Mar. 2019.

[12] A. Jarraya, A. Bouzeghoub, A. Borgi, and K. Arour, "DCR: A new distributed model for human activity recognition in smart Homes," *Expert Syst. Appl.*, vol. 140, Feb. 2020, Art. no. 112849.

[13] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Proc. Int. Conf. Pervas. Comput.* Berlin, Germany: Springer, 2004, pp. 1–17.

[14] L. C. Jatoba, U. Grossmann, C. Kunze, J. Ottenbacher, and W. Stork, "Context-aware mobile health monitoring: Evaluation of different pattern recognition methods for classification of physical activity," in *Proc. 30th Annu. Int. Conf. Eng. Med. Biol. Soc.*, Aug. 2008, pp. 5250–5253.

[15] T. Brezmes, J.-L. Gorricho, and J. Cotrina, "Activity recognition from accelerometer data on a mobile phone," in *Proc. Int. Work-Conf. Artif. Neural Netw.* Berlin, Germany: Springer, 2009, pp. 796–799.

[16] K. Altun and B. Barshan, "Human activity recognition using inertial/magnetic sensor units," in *Proc. Int. Workshop Hum. Behav. Understand.* Berlin, Germany: Springer, 2010, pp. 38–51.

[17] M. Shoaib, H. Scholten, and P. J. M. Havinga, "Towards physical activity recognition using smartphone sensors," in *Proc. IEEE 10th Int. Conf. Ubiquitous Intell. Comput.*, Dec. 2013, pp. 80–87.

[18] C. Medrano, R. Igual, I. Plaza, and M. Castro, "Detecting falls as novelties in acceleration patterns acquired with smartphones," *PLoS ONE*, vol. 9, no. 4, Apr. 2014, Art. no. e94811.

[19] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. M. Havinga, "Fusion of smartphone motion sensors for physical activity recognition," *Sensors*, vol. 14, no. 6, pp. 10146–10176, 2014.

[20] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, "Transition-aware human activity recognition using smartphones," *Neurocomputing*, vol. 171, pp. 754–767, Jan. 2016.

[21] G. Vavoulas, C. Chatzaki, T. Malliotakis, M. Pediaditis, and A. M. Tsiknakis, "The mobiact dataset: Recognition of activities of daily living using smartphones," in *Proc. ICT4AgeingWell*, 2016, pp. 143–151.

[22] T. Sztyler and H. Stuckenschmidt, "On-body localization of wearable devices: An investigation of position-aware activity recognition," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. (PerCom)*, Mar. 2016, pp. 1–9.

[23] D. Micucci, M. Mobilio, and P. Napoletano, "UniMiB SHAR: A new dataset for human activity recognition using acceleration data from smartphones," 2016, *arXiv:1611.07688*. [Online]. Available: http://arxiv.org/abs/1611.07688

[24] G. Vavoulas, M. Pediaditis, C. Chatzaki, E. G. Spanakis, and A. M. Tsiknakis, "The mobifall dataset: Fall detection and classification with a smartphone," in *Artificial Intelligence: Concepts, Methodologies, Tools, and Applications*. Hershey, PA, USA: IGI Global, 2017, pp. 1218–1231.

[25] J. Santoyo-Ramón, E. Casilari, and J. Cano-García, "Analysis of a smartphone-based architecture with multiple mobility sensors for fall detection with supervised learning," *Sensors*, vol. 18, no. 4, p. 1155, Apr. 2018.

[26] Z.-Y. He and L.-W. Jin, "Activity recognition from acceleration data using AR model representation and SVM," in *Proc. ICMLC*, vol. 4, Jul. 2008, pp. 2245–2250.

[27] Z. He and L. Jin, "Activity recognition from acceleration data based on discrete consine transform and SVM," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Oct. 2009, pp. 5041–5044.

[28] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Proc. ESANN*, vol. 3, 2013, p. 3.

[29] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions," in *Proc. Int. Workshop Wearable Implant. Body Sensor Netw.*, 2006, p. 4.

[30] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen, "Activity classification using realistic data from wearable sensors," *IEEE Trans. Inf. Technol. Biomed.*, vol. 10, no. 1, pp. 119–128, Jan. 2006.

[31] E. M. Tapia, S. S. Intille, W. Haskell, K. Larson, J. Wright, A. King, and R. Friedman, "Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor," in *Proc. 11th IEEE Int. Symp. Wearable Comput.*, Oct. 2007, pp. 37–40.

[32] M. Ermes, J. Parkka, and L. Cluitmans, "Advancing from offline to online activity recognition with wearable sensors," in *Proc. 30th Annu. Int. Conf. Eng. Med. Biol. Soc.*, Aug. 2008, pp. 4451–4454.

[33] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explor. Newslett.*, vol. 12, no. 2, pp. 74–82, Dec. 2010.

[34] S. D. Lara and M. A. Labrador, "A mobile platform for real-time human activity recognition," in *Proc. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2012, pp. 667–671.

[35] K. H. Walse, R. V. Dharaskar, and V. M. Thakare, "PCA based optimal ANN classifiers for human activity recognition using mobile sensors data," in *Proc. 1st Int. Conf. Inf. Commun. Technol. Intell. Syst.*, vol. 1. Cham, Switzerland: Springer, 2016, pp. 429–436.

[36] M. H. Kabir, M. R. Hoque, K. Thapa, and S.-H. Yang, "Two-layer hidden Markov model for human activity recognition in home environments," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 1, Jan. 2016, Art. no. 4560365.

[37] Y.-P. Chen, J.-Y. Yang, S.-N. Liou, G.-Y. Lee, and J.-S. Wang, "Online classifier construction algorithm for human activity detection using a tri-axial accelerometer," *Appl. Math. Comput.*, vol. 205, no. 2, pp. 849–860, 2008.

[38] T.-P. Kao, C.-W. Lin, and J.-S. Wang, "Development of a portable activity detector for daily activity recognition," in *Proc. IEEE Int. Symp. Ind. Electron.*, Jul. 2009, pp. 115–120.

[39] M. Berchtold, M. Budde, D. Gordon, H. R. Schmidtke, and M. Beigl, "ActiServ: Activity recognition service for mobile phones," in *Proc. Int. Symp. Wearable Comput. (ISWC)*, Oct. 2010, pp. 1–8.

[40] D. Minnen, T. Westeyn, D. Ashbrook, P. Presti, and T. Starner, "Recognizing soldier activities in the field," in *4th Int. workshop wearable Implant. body sensor Netw. (BSN)*, pp. 236–241. Springer, 2007.

[41] D. Lara, A. J. Pérez, M. A. Labrador, and J. D. Posada, "Centinela: A human activity recognition system based on acceleration and vital sign data," *Pervasive Mobile Comput.*, vol. 8, no. 5, pp. 717–729, Oct. 2012.

[42] C. Zhu and W. Sheng, "Human daily activity recognition in robot-assisted living using multi-sensor fusion," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 2154–2159.

[43] M. Abu Alsheikh, A. Selim, D. Niyato, L. Doyle, S. Lin, and H.-P. Tan, "Deep activity recognition models with triaxial accelerometers," 2015, *arXiv:1511.04664*. [Online]. Available: http://arxiv.org/abs/1511.04664

[44] L. Zhang, X. Wu, and D. Luo, "Recognizing human activities from raw accelerometer data using deep neural networks," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2015, pp. 865–870.

[45] S. Fallmann and J. Kropf, "Human activity recognition of continuous data using hidden Markov models and the aspect of including discrete data," in *Proc. Int. Conf. Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput.*, Jul. 2016, pp. 121–126.

[46] M. O. Padar, A. E. Ertan, and C. Candan, "Classification of human motion using radar micro-Doppler signatures with hidden Markov models," in *Proc. IEEE Radar Conf. (RadarConf)*, May 2016, pp. 1–6.

[47] P. Vepakomma, D. De, S. K. Das, and S. Bhansali, "A-wristocracy: Deep learning on wrist-worn sensing for recognition of user complex activities," in *Proc. IEEE 12th Int. Conf. Wearable Implant. Body Sensor Netw. (BSN)*, Jun. 2015, pp. 1–6.

[48] L. Zhang, X. Wu, and D. Luo, "Human activity recognition with HMM-DNN model," in *Proc. IEEE 14th Int. Conf. Cognit. Informat. Cognit. Comput.*, Jul. 2015, pp. 192–197.

[49] N. Y. Hammerla, S. Halloran, and T. Ploetz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," 2016, *arXiv:1604.08880*. [Online]. Available: http://arxiv.org/abs/1604.08880

[50] S. Zhang, W. W. Ng, J. Zhang, and C. D. Nugent, "Human activity recognition using radial basis function neural network trained via a minimization of localized generalization error," in *Proc. Int. Conf. Ubiquitous Comput. Ambient Intell.* Cham, Switzerland: Springer, 2017, pp. 498–507.
[51] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, "Convolutional neural networks for human activity recognition using mobile sensors," in *Proc. 6th Int. Conf. Mobile Comput., Appl. Services*, 2014, pp. 197–205.
[52] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *Proc. Int. Conf. Web-Age Inf. Manage.* Cham, Switzerland: Springer, 2014, pp. 298–310.
[53] Y. Chen and Y. Xue, "A deep learning approach to human activity recognition based on single accelerometer," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2015, pp. 1488–1492.
[54] S. Ha, J.-M. Yun, and S. Choi, "Multi-modal convolutional neural networks for activity recognition," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2015, pp. 3017–3022.
[55] W. Jiang and Z. Yin, "Human activity recognition using wearable sensors by deep convolutional neural networks," in *Proc. 23rd ACM Int. Conf. Multimedia*, Oct. 2015, pp. 1307–1310.
[56] J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *Proc. IJCAI*, vol. 15, 2015, pp. 3995–4001.
[57] Y. Chen, K. Zhong, J. Zhang, Q. Sun, and X. Zhao, "LSTM networks for mobile human activity recognition," in *Proc. Int. Conf. Artif. Intell., Technol. Appl.*, 2016, pp. 50–53.
[58] H. Gjoreski, J. Bizjak, M. Gjoreski, and M. Gams, "Comparing deep and classical machine learning methods for human activity recognition using wrist accelerometer," in *Proc. Workshop Deep Learn. Artif. Intell.*, New York, NY, USA, vol. 10, 2016, p. 970.
[59] S. Ha and S. Choi, "Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 381–388.
[60] C. Liu, L. Zhang, Z. Liu, K. Liu, X. Li, and Y. Liu, "Lasagna: Towards deep hierarchical understanding and searching over mobile sensing data," in *Proc. 22nd Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2016, pp. 334–347.
[61] F. J. O. Morales and D. Roggen, "Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations," in *Proc. ACM Int. Symp. Wearable Comput.*, Sep. 2016, pp. 92–99.
[62] D. Ravi, C. Wong, B. Lo, and G.-Z. Yang, "Deep learning for human activity recognition: A resource efficient implementation on low-power devices," in *Proc. IEEE 13th Int. Conf. Wearable Implant. Body Sensor Netw. (BSN)*, Jun. 2016, pp. 71–76.
[63] D. Ravì, C. Wong, B. Lo, and G.-Z. Yang, "A deep learning approach to on-node sensor data analytics for mobile or wearable devices," *IEEE J. Biomed. Health Inform.*, vol. 21, no. 1, pp. 56–64, Jan. 2017.
[64] C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Syst. Appl.*, vol. 59, pp. 235–244, Oct. 2016.
[65] J. Wang, X. Zhang, Q. Gao, H. Yue, and H. Wang, "Device-free wireless localization and activity recognition: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 66, no. 7, pp. 6258–6267, Jul. 2017.
[66] T. Zebin, P. J. Scully, and K. B. Ozanyan, "Human activity recognition with inertial sensors using a deep learning approach," in *Proc. IEEE SENSORS*, May 2016, pp. 1–3.
[67] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Exploiting multi-channels deep convolutional neural networks for multivariate time series classification," *Frontiers Comput. Sci.*, vol. 10, no. 1, pp. 96–112, Feb. 2016.
[68] Y. Kim and Y. Li, "Human activity classification with transmission and reflection coefficients of on-body antennas through deep convolutional neural networks," *IEEE Trans. Antennas Propag.*, vol. 65, no. 5, pp. 2764–2768, May 2017.
[69] S.-M. Lee, S. Min Yoon, and H. Cho, "Human activity recognition from accelerometer data using convolutional neural network," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2017, pp. 131–134.
[70] S. Mohammed and I. Tashev, "Unsupervised deep representation learning to remove motion artifacts in free-mode body sensor networks," in *Proc. IEEE 14th Int. Conf. Wearable Implant. Body Sensor Netw. (BSN)*, May 2017, pp. 183–188.
[71] M. Panwar, S. Ram Dyuthi, K. Chandra Prakash, D. Biswas, A. Acharyya, K. Maharatna, A. Gautam, and G. R. Naik, "CNN based approach for activity recognition using a wrist-worn accelerometer," in *Proc. 39th Annu. Int. Conf. Eng. Med. Biol. Soc. (EMBC)*, Jul. 2017, pp. 2438–2441.
[72] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "DeepSense: A unified deep learning framework for time-series mobile sensing data processing," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 2017, pp. 351–360.
[73] M. Dong, J. Han, Y. He, and X. Jing, "HAR-Net: Fusing deep representation and hand-crafted features for human activity recognition," in *Int. Conf. Signal Inf. Process., Netw. Comput.* Singapore: Springer, 2018, pp. 32–40.
[74] F. M. Rueda, R. Grzeszick, G. A. Fink, S. Feldhorst, and M. T. Hompel, "Convolutional neural networks for human activity recognition using body-worn sensors," *Informatics*, vol. 5, no. 2, p. 26, 2018.
[75] S. Wan, L. Qi, X. Xu, C. Tong, and Z. Gu, "Deep learning models for real-time human activity recognition with smartphones," *Mobile Netw. Appl.*, vol. 25, pp. 743–755, Dec. 2019.
[76] K. Xia, J. Huang, and H. Wang, "LSTM-CNN architecture for human activity recognition," *IEEE Access*, vol. 8, pp. 56855–56866, 2020.
[77] M. Edel and E. Koppe, "Binarized-BLSTM-RNN based human activity recognition," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat. (IPIN)*, Oct. 2016, pp. 1–7.
[78] A. Murad and J.-Y. Pyun, "Deep recurrent neural networks for human activity recognition," *Sensors*, vol. 17, no. 11, p. 2556, 2017.
[79] Y. Guan and T. Plötz, "Ensembles of deep LSTM learners for activity recognition using wearables," *Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 1, no. 2, pp. 1–28, 2017.
[80] C. Xu, D. Chai, J. He, X. Zhang, and S. Duan, "InnoHAR: A deep neural network for complex human activity recognition," *IEEE Access*, vol. 7, pp. 9893–9902, 2019.
[81] T. Plötz, N. Y. Hammerla, and P. L. Olivier, "Feature learning for activity recognition in ubiquitous computing," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, 2011, pp. 1729–1734.
[82] H. Fang and C. Hu, "Recognizing human activity in smart home using deep learning algorithm," in *Proc. 33rd Chin. Control Conf.*, Jul. 2014, pp. 4716–4720.
[83] T. Hayashi, M. Nishida, N. Kitaoka, and K. Takeda, "Daily activity recognition based on DNN using environmental sound and acceleration signals," in *Proc. 23rd Eur. Signal Process. Conf. (EUSIPCO)*, Aug. 2015, pp. 2306–2310.
[84] N. D. Lane and P. Georgiev, "Can deep learning revolutionize mobile sensing?" in *Proc. 16th Int. Workshop Mobile Comput. Syst. Appl.*, pp. 117–122, 2015.
[85] L. Zhang, X. Wu, and D. Luo, "Real-time activity recognition on smartphones using deep neural networks," in *Proc. IEEE 12th Int. Conf. Ubiquitous Intell. Comput.*, Aug. 2015, pp. 1236–1242.
[86] S. Bhattacharya and N. D. Lane, "From smart to deep: Robust activity recognition on smartwatches using deep learning," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops*, Mar. 2016, pp. 1–6.
[87] V. Radu, N. D. Lane, S. Bhattacharya, C. Mascolo, M. K. Marina, and F. Kawsar, "Towards multimodal deep learning for activity recognition on mobile devices," in *Proc. ACM Int. Joint Conf. Pervas. Ubiquitous Comput., Adjunct*, Sep. 2016, pp. 185–188.
[88] B. Almaslukh, J. Almuhtadi, and A. Artoli, "An effective deep autoencoder approach for online smartphone-based human activity recognition," *Int. J. Comput. Sci. Netw. Secur.*, vol. 17, no. 4, pp. 160–165, 2017.
[89] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, and K.-R. Müller, "Explaining deep neural networks and beyond: A review of methods and applications," *Proc. IEEE*, vol. 109, no. 3, pp. 247–278, Mar. 2021.
[90] E. Sansano, R. Montoliu, and Ó. Belmonte Fernández, "A study of deep neural networks for human activity recognition," *Comput. Intell.*, vol. 36, no. 3, pp. 1113–1139, Aug. 2020.
[91] F. Li, K. Shirahama, M. A. Nisar, L. Köping, and M. Grzegorzek, "Comparison of feature learning methods for human activity recognition using wearable sensors," *Sensors*, vol. 18, no. 2, p. 679, 2018.
[92] S. Hosseini, S. Hee Lee, and N. Ik Cho, "Feeding hand-crafted features for enhancing the performance of convolutional neural networks," 2018, *arXiv:1801.07848*. [Online]. Available: http://arxiv.org/abs/1801.07848
[93] R. Singh, A. Sonawane, and R. Srivastava, "Recent evolution of modern datasets for human activity recognition: A deep survey," *Multimedia Syst.*, vol. 26, no. 2, pp. 83–106, 2019.

[94] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.
[95] S. Iloga, "Customizable HMM-based measures to accurately compare tree sets," *Pattern Anal. Appl.*, vol. 5, pp. 1–22, Mar. 2021.
[96] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognit.*, vol. 36, no. 2, pp. 451–461, Feb. 2003.
[97] S. Iloga, O. Romain, and M. Tchuenté, "An accurate HMM-based similarity measure between finite sets of histograms," *Pattern Anal. Appl.*, vol. 22, no. 3, pp. 1079–1104, Aug. 2019.
[98] I. H. Witten and F. Eibe. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. [Online]. Available: http://weka.sourceforge.net/
[99] *Nvidia Jetson Tx2 User Manual*. Accessed: Jul. 21, 2021. [Online]. Available: https://www.manualslib.com/manual/1634693/Nvidia-Jetson-Tx2.html
[100] *Mobile Processor Exynos 7 Octa (5433)*. Accessed: Jul. 21, 2021. [Online]. Available: https://www.samsung.com/semiconductor/minisite/exynos/products/mobileprocessor/exynos-7-octa-5433/
[101] A. A. López and M. O. Arias-Estrada, "Implementing hidden Markov models in a hardware architecture," in *Proc. Int. Meeting Comput. Sci.*, vol. 2, 2001, pp. 1007–1016.
[102] A. Lu, Z. Fang, N. Farahpour, and L. Shannon, "CHIP-KNN: A configurable and high-performance k-nearest neighbors accelerator on cloud FPGAs," in *Proc. Int. Conf. Field-Program. Technol. (ICFPT)*, Dec. 2020, pp. 139–147.

SYLVAIN ILOGA received the Ph.D. degree in computer science from the University of Yaoundé 1, Cameroon, in January 2018. Since January 2010, he has been a Teacher with the Department of Computer Science, High Teachers' Training College, Maroua, Cameroon. From September 2017 to August 2019, he completed a research and teaching internship with the Department of Electronic Engineering and Industrial Computing, IUT of Cergy-Pontoise, Neuville University, France. He was promoted to the rank of a Lecturer in Cameroon, in May 2018. Subsequently in January 2019, he obtained his qualification for the functions of a Lecturer in France, section 27 (Computer Science). His research interests include design of taxonomies for hierarchical classification, sequential data mining, machine learning using hidden Markov models, and the implementation of reconfigurable architectures based on the FPGA technology.

ALEXANDRE BORDAT graduated from the École Nationale Supérieure de l'Électronique et de ses Applications (ENSEA), France. He is currently pursuing the Ph.D. degree in embedded system with ENSEA. He is also an Engineer of embedded system.

JULIEN LE KERNEC (Senior Member, IEEE) received the B.Eng. and M.Eng. degrees in electronic engineering from Cork Institute of Technology, Ireland, in 2004 and 2006, respectively, and the Ph.D. degree in electronic engineering from University Pierre and Marie Curie, France, in 2011. He is currently a Senior Lecturer with the School of Engineering, University of Glasgow. He is also a Senior Lecturer with the University of Electronic Science and Technology of China and an Adjunct Associate Professor with the ETIS Laboratory, University of Cergy-Pontoise, France. His research interests include radar system design, software-defined radio/radar, signal processing, and health applications.

OLIVIER ROMAIN (Member, IEEE) received the Engineering degree in electronics from ENS Cachan, the master's degree in electronics from Louis Pasteur University, and the Ph.D. degree in electronics from Pierre and Marie Curie University, Paris. From 2012 to 2019, he was the Head of the Department of Architecture, ETIS-UMR8051 Laboratory. Since January 2020, he has been the Director of the ETIS-UMR8051 Laboratory. He is currently a University Professor of electrical engineering with CY Cergy Paris University. His research interests include systems on chips for diffusion and biomedical applications.