B-HAR: An Open-Source Baseline Framework for In-Depth Study of Human Activity Recognition Datasets and Workflows

CRISTIAN TURETTA (Graduate Student Member, IEEE), FLORENC DEMROZI (Member, IEEE), AND GRAZIANO PRAVADELLI (Senior Member, IEEE)

1Department of Computer Science, University of Verona, 37129 Verona, Italy
2Department of Electrical Engineering and Computer Science, University of Stavanger, 4021 Stavanger, Norway
3Department of Engineering for Innovation Medicine, University of Verona, 37129 Verona, Italy

Corresponding author: Cristian Turetta (cristian.turetta@univr.it)

This work was supported in part by the “PREPARE” Project under Grant n. F/310130/05/X56, Grant CUP: B3923001730005, and Grant D.M. MiSE 31/12/2021; and in part by the “UNISCO” Project-PR Veneto Fondo Sociale Europeo (FSE)+ 2021-2027 under Grant CUP: 1695-0026-553-2023.

ABSTRACT Human Activity Recognition (HAR) plays a pivotal role in diverse fields such as healthcare, performance monitoring, and risk prevention, employing Machine Learning (ML) and Deep Learning (DL) algorithms. This paper introduces B-HAR (Baseline-HAR), an open-source framework based on Service-Oriented Architecture to facilitate the engineering and evaluation of different ML/DL-based HAR methodologies. By automating the creation and implementation of a baseline workflow, B-HAR enables researchers to assess and compare HAR methods effectively. It integrates prevalent data-processing techniques and popular machine and deep learning models, ensuring consistency in data preprocessing while allowing for custom model integration. The framework’s efficacy is demonstrated across nine prominent HAR datasets, encompassing various sensor types and placements, showcasing its utility in engineering applications, particularly in healthcare, where it aids in diagnosis, rehabilitation, and treatment optimization for neurological and physiatric disorders, as well as assisting individuals with special needs.

INDEX TERMS Human activity recognition, sensor data, machine learning, deep learning, open-source framework.

I. INTRODUCTION

With the dawn of Artificial Intelligence (AI) and the Internet of Things (IoT), embedded sensors featuring AI capabilities have started to be incorporated into personal devices such as smartphones, smartwatches, clothes, and objects of daily life. This trend is driving the development of new research directions and the development of intelligent applications for Human Activity Recognition (HAR). HAR has become a widespread topic due to its significance in many areas, including wellbeing, health care, gaming, sports, and Activities of Daily Life (ADLs) monitoring [1], [2], [3].

Among the various application fields, HAR is becoming more and more relevant, in particular, for monitoring and coaching the older population, since, according to the 2019 World Population Prospects [4], in 2018, for the first time in human history, persons aged 65+ years outnumbered children under five years. By 2050, people aged 65+ (1.5 billion) will outnumber adolescents and youth aged 15 to 24 (1.3 billion). The increase in the population of people over 65 also impacts the number of persons affected by pathologies. Thus, despite an increase in life expectancy, most people lose self-efficacy as they age, which consequently reduces their quality of life. This decrease in autonomy, in conjunction with the necessity of modifying daily habits and developing new behaviors (e.g., taking medicine at periodic intervals), has pushed towards the definition of...
new HAR technologies and techniques for diagnosis [5], rehabilitation [6], virtual coaching [7], optimization of the treatment [8], and in general to monitor the health status and implement telemedicine and teleassistance support systems [9], [10], [11].

The information that HAR algorithms process derives from cameras, environmental sensors (such as temperature, humidity, light, and seismic activity), and wearable embedded sensors (such as inertial, environmental, and physiological sensors) [12], [13], [14], [15]. Although cameras are widely utilized in HAR, gathering video data raises privacy concerns and necessitates considerable computational resources. Due to these factors, several researchers favor using embedded and ambient sensors [16], [17]. Particularly in HAR, inertial sensors have produced excellent results, and their application in conjunction with other kinds of sensors is expanding quickly. The ability of these sensors to capture human movement [18] and psychological state [19] directly correlates with how widely their usage. Additionally, inertial sensors are less expensive and integrated into most individuals’ wearable items [20], [21].

A. PROBLEM STATEMENT

Despite the devices used to sense the environment and collect data related to human behaviors, the gathered information is then generally processed through Machine Learning (ML) or Deep Learning (DL) based workflows to recognize and classify the target activities. However, the definition and implementation of such workflows are particularly challenging, and evaluating a HAR pipeline design is an error-prone task, which unquestionably influences the accuracy and correctness of the achievable results [22]. Furthermore, the increasing volume of data collected for HAR applications, local computing is often inadequate to handle the computational demands these data require. In this context, microservices architectures have become essential, as they enable the distribution of computational tasks across multiple, cloud-based services, thus allowing a more efficient infrastructure [23].

Figure 1 shows the standard strategy for HAR algorithmic design. It often entails the following steps: 1) identification of data sources, such as sensors and devices used to perceive data; 2) data collection and preprocessing; 3) AI model selection, training, and testing; and 4) HAR model performance evaluation. Based on this overall workflow, there are several non-standardized procedures, particularly in the data preprocessing (step 2) and AI model selection and training (step 3), which are crucial to the success of the HAR method.

Data preprocessing, in particular, covers a variety of elaborations, including normalization, noise reduction, balancing, or feature extraction. Their use has a significant impact on how well the recognition model performs. Unfortunately, neither a well-defined strategy for applying data processing processes nor specific guidelines for model selection and training exist. Additionally, even when using the same dataset, examining, comparing, and evaluating the quality of various HAR techniques is challenging due to the need for a clearly defined procedure. For example, some of the most frequent mistakes made when implementing a HAR methodology include:

- data errors (i.e., null values, infinite values, or corrupted values) are not handled;
- normalization applied on the initial dataset instead of applying it to the training dataset and then to the validation and testing datasets;
- incorrect training and testing (train/test split) procedures are applied;
- performance evaluation of multi-class problems is provided only in terms of accuracy, leaking the usage of sensitivity, specificity, precision, and f1-score metrics.

The lack of an instrument that unifies the process phases and clarifies their application to the user typically causes the mistakes above.

In addition, concerning the model selection step, HAR has noticed a rise in interest in DL techniques, primarily because of their capacity to handle raw data that, in theory, does not require explicit data pretreatment procedures like normalization, filtering, or feature extractions [24]. However, large datasets of human behavior are necessary to construct DL models. Due to the lack of such datasets, several processing methods must be used [18]. Determining the DL models’ ideal configuration is another salient characteristic affecting usability. In DL, the use of hyperparameters searching techniques only offers the chance to dynamically set model parameters (such as learning rate, training epochs, batch size, weight initialization, activation functions, and dropout regularization) rather than the architecture of the DL model (such as type of layers, connection between layers, and their arrangement, etc.) [25].

B. PAPER CONTRIBUTION

To the best of our knowledge, only a few libraries/frameworks (such as Scikit-Learn [26], Orange [27], Weka [28], TSFEL [29], or Lazy Predict [30]) integrate the capabilities to design and test HAR pipelines easily. However, it is essential to note that these libraries/frameworks are not exclusively designed for HAR purposes, and more importantly, they only include some of the processing steps needed. Thus, there are currently no articles or tools that propose a framework that can be utilized as a baseline to guide researchers in the correct and effective definition and implementation of HAR workflows and their fair comparison.

To fill in the gap, this paper proposes B-HAR, a new open-source framework1 based on Service-Oriented Architecture (SOA) principles to support researchers in implementing HAR workflows. The tool utilizes modular and interoperable features of SOA, providing increased flexibility and scalability. By adopting SOA principles, B-HAR ensures seamless

1B-HAR GitHub page.
orchestration and utilization of its various micro-services, depicted in Figure 2 and detailed in Section II. The main characteristics of B-HAR are:

- unification of the main workflow of Figure 1 into a single service, making the application of the various preprocessing steps transparent to the user;
- definition of a precise order of application of the data preprocessing steps to minimize errors;
- provision to the users with baseline information about the most famous HAR datasets such that new HAR approaches can be exhaustively compared to existing results;
- automatic application, given a new HAR dataset, of the grid search technique on the most used pattern recognition models to provide the user with baseline results;
- provision to the users with the possibility of implementing their own HAR (ML or DL-based) models and integrating them in B-HAR, on which they can reuse previously implemented preprocessing pipelines already tested on other models and/or datasets.

In conclusion, according to the aforementioned user’s needs and B-HAR features, it is worth noting that this work does not intend to present any new HAR model or standard methodology. Instead, we aim to provide the scientific community with a flexible and customizable framework to implement and compare existing and new HAR models without making mistakes.

C. ORGANIZATION OF THE PAPER
The rest of the paper’s structure is the following. Section II describes B-HAR and the underpinning methodology. Section III presents the results of an exhaustive experimental campaign. Section IV discusses HAR design workflow issues and recommendations. Finally, Section V concludes the papers and provides an overview of future extensions.

II. METHODOLOGY
B-HAR employs an SOA design, including seven distinct services, each consisting of various micro-services, as illustrated in Figure 2. Each service within B-HAR can operate independently or orchestrated to form a comprehensive analysis pipeline. In Figure 2, red dots highlight computationally intensive and time-consuming operations. The green dot denotes a computation service that activates only one of its micro-services. Micro-services marked in grey represent optional computational elements that users of B-HAR may choose to utilize. B-HAR orchestrates the internal micro-services to execute them in the sequence denoted by the dashed lines. The following sections provide detailed descriptions of these services.

A. INPUT
In order to start the analysis, the first step is to perform the B-HAR object instantiation on the server. It takes several input parameters, which are further detailed in this section. In particular, the dataset input can be a single file, a directory with different dataset files, or it can be a link to a dataset on Kaggle [31]. In the case of a directory, B-HAR will merge all files inside it to create the entire dataset, i.e., values of the signals perceived by sensors during human activities, enriched by information concerning the testing subject identity, the data collection session, and the performed activity at a specific timestamp. On the other hand, it will directly load the dataset using Kaggle’s API. The only constraint is that the dataset format must be compliant with B-HAR-allowed formats.

1) DATASET FORMAT
Since HAR datasets present different structures and information types, B-HAR can handle different dataset formats flexibly. Table I shows the generic, customizable structure of the input dataset that B-HAR can handle; ✓ identifies...
C. Turetta et al.: B-HAR: An Open-Source Baseline Framework for In-Depth Study

FIGURE 2. Overview of the B-HAR micro-services and structure.

mandatory data, while [✓] identifies optional data. B-HAR requires a dataset composed of signals from one or more sensors (Sensor\textsubscript{i}), a label (A) indicating the human activity performed by the testing subject, and his/her identifier (T). In general, these data are collected during supervised collection sessions (S) where subjects (T) perform only one activity (e.g., walking) or a series of consecutive activities (e.g., sitting, standing up, standing, walking, sitting down, and sitting again). B-HAR handles both data collection approaches (i.e., one activity/one session, many activities/one session).

Moreover, our framework can handle input datasets where data appears in different orders as follows:

- Ti, Data, A, T, S;
- Data, A, T, S, Ti;
- A, T, S, Data, Ti;
- A, T, S, Ti, Data;
- S, A, T, Ti, Data;
- and other configurations.

Besides, if time (Ti) information is lacking, B-HAR uses the discrete sampling frequency the sensors adopt. From now on, all the processing steps are performed by considering the applied data collection methodology (i.e., one activity/one session or many activities/one session). In the many activities/one session approach, B-HAR provides the possibility to generate transition classes that generally are not handled by the users (e.g., a sequence of activities as [sitting, standing, walking, standing up, walking] is represented as [sitting, sitting_to_standing, standing, standing_to_walking, walking, walking_to_standing, standing, standing_to_walking, walking]).

2) CONFIGURATION PARAMETERS
The B-HAR object empowers users with complete control over the computation modules and related sub-modules depicted in Figure 2. By adjusting its initialization parameters, B-HAR users can effortlessly tailor the data processing workflow and experiment with various configuration
TABLE 1. B-HAR dataset structure.

| Column 1 | Column 2 | Column 3 | Column 4 |
|----------|----------|----------|----------|
| Time(T)  | →        | →        | →        |
| Sensor   | →        | →        | →        |
| y         | →        | →        | →        |
| Sensor   | →        | →        | →        |
| y         | →        | →        | →        |
| A        | →        | →        | →        |
| T        | →        | →        | →        |
| S        | →        | →        | →        |

B-HAR incorporates four distinct types of filters [33]:

- **low-pass filters** allow signals with a frequency lower than a chosen cutoff to pass while attenuating the remainder;
- **high-pass filters** permit signals with a frequency higher than a chosen cutoff to pass while attenuating the rest;
- **band-pass filters** enable signals within a designated range to pass while attenuating frequencies outside of it;
- **band-stop filters** allow most frequencies to pass unchanged but attenuate those within a specific range.

B-HAR users can also choose not to apply noise removal techniques, relying entirely on the capabilities of pattern recognition models.

B-HAR does not apply the data handling and noise removal to the entire dataset. It applies them to a single data collection session or activity based on the dataset type.²

C. DATA REPRESENTATION

In general, data can manifest in various forms:

- as raw data without any transformations;
- following the application of feature extraction techniques;
- after segmentation, i.e., grouped into time windows.

Since single data observations in a time series are not statistically informative, a common approach is to apply segmentation or feature extraction. Moreover, HAR models resonate with ongoing activities over time and not on single observations. With such an aim, segmentation (especially in deep learning) and feature extractions (especially in classical machine learning) have become state-of-the-art procedures for HAR.

B-HAR allows users to select one of the data as mentioned above treatment procedures.

1) RAW DATA

In this form of representation, B-HAR does not conduct any treatment, and the pre-processing module receives the dataset generated by the cleaning module as input. This representation applies when the user has already pre-processed the data offline and wishes to utilize B-HAR to analyze various HAR models.

2) SEGMENTATION

Given a $n \times 1$ time series dataset $HAR_D$, sampled at a fixed frequency $S_f$ (expressed in Hertz), and a time-window segment of dimension $T_w$ (expressed in seconds), the basic segmentation process implemented in B-HAR returns a dataset $SHAR_D$ of dimension $(n/(T_w * S_f)) \times (T_w * S_f)$. For example, given a $100 \times 1$ time series dataset $HAR_D$, a sampling frequency $S_f = 25Hz$, and a time window $T_w = 2$ seconds, the segmentation process returns a dataset $SHAR_D$ of dimension $(100/(25 * 2)) \times (2 * 25)$, i.e., $2 \times 50$. Nevertheless, many methodologies maintain an overlapping fragment between two consecutive segments. Such a fragment provides the model with information on the preceding segment context. B-HAR can also handle this overlapping if the user specifies it.

3) FEATURES EXTRACTION

The feature extraction process delves into the raw data’s time and frequency domains. The utilization of time-domain features is predominant because extracting them requires less computational effort than extracting frequency-domain features, as indicated by [29] and [34].

²$M\%$ of the samples in a time series column. By default $M = 5$. ³One activity/session, or many activities/one session.
TABLE 2. B-HAR json file configuration parameters. (gray lines define the configuration used in the experimental results section.)

| Parameter                  | Description                                                                 | Values Range                                                                 |
|----------------------------|-----------------------------------------------------------------------------|------------------------------------------------------------------------------|
| dataset_path               | Path to the dataset directory.                                              | String, Kaggle URL.                                                          |
| dataset_sampling_freq      | Sampling frequency of loaded dataset in Hz.                                  | Integer                                                                     |
| header_format              | Header format of the input dataset.                                         | TIDA, TIDAT, DA, DATS, etc.                                                  |
| derive_header             | The input dataset file contains also the header or should it be derived.    | Bool                                                                        |
| separator                  | .csv separator                                                              |同期字符 continuous, fragmented                                               |
| dataset_type               | The type of data in the dataset.                                            | Float                                                                       |
| segment_time               | Time for desired window length, in seconds.                                 | CLASS, P_ID                                                                  |
| overlap                    | Overlap between time windows, in seconds.                                   |                                                                             |
| group_by                   | Show stats by attribute.                                                   |                                                                             |
| representation             | Data treatment type.                                                        | segmentation, raw, features_extraction                                      |
| features_domain           | Available domains for features extraction.                                  | statistical, spectral, temporal, all                                          |
| features_selection        | Toggle features selection.                                                  | Bool (true, false)                                                           |
| models                     | Implemented ML and DL models.                                               | KNN, WNN, LDA, QDA, SVM, RF, DT, CNN, NN, LSTM, RNN                         |
| user_models                | User defined ML and DL models.                                              | Must be implemented with sklearn or tensorflow                               |
| scaling_method            | Normalize data.                                                             | none, minmax, robust, standard                                              |
| split_method               | Train/Test split method.                                                    | intra, inter                                                                |
| feature_selection_mtd      | Features selection technique.                                               | variance, 11, tree-based, recursive                                          |
| n_features_to_select       | Number of features to select, available only with recursive selection method.| Integer                                                                     |
| balancing_method           | For unbalanced dataset, balancing techniques.                              | random under, near miss, edited nn, random over, smote, adasyn, kmmeans smote.|
| replace_error_method       | Substitute NaN and Inf values method.                                       | mean, forward, backward, constant, interpolate                              |
| constant_value             | The value which substitutes NaN and ±oo values.                             | Float                                                                       |
| filter_type                | Data filtering techniques.                                                  | lowpass, highpass, bandpass, bandstop                                        |
| filter_order               | The order of the applied filter.                                            | Integer (e.g., 4)                                                 |
| filter_cutoff              | Low/High-Cutoffs frequency in Hz.                                           | Integer (e.g., 20 Hz)                                                       |
| test_size                  | The size of test data.                                                      | Float (e.g., 25%)                                                            |
| use_features               | If features selection is enabled, extracted features are used as input of the CNN.| Bool (true, false)                                                            |

B-HAR utilizes the original time series dataset, along with specifications such as the type of features to extract in the desired domain (either time or frequency), the size of the time window, and the degree of overlap, to generate the corresponding feature-based representation in the time and/or frequency domains. B-HAR integrates the Time Series Feature Extraction Library (TSFEL), as introduced in [29].

Feature extraction is of primary importance if the sensors sample at a non-fixed frequency; B-HAR allows the creation of windows with a fixed time dimension (e.g., 2 seconds) regardless of the system sampling frequency. For example, suppose we have a time window of 2 seconds composed of 127 samples and another time window of 2 seconds of 100 samples. In that case, these time windows will be represented by one snapshot of 150 features each after the feature extraction step.

D. DATA ENGINEERING

This service employs methods to minimize the dataset size by eliminating redundant data, choosing a subset of extracted features, and ensuring dataset balance. Additionally, it standardizes the data using scaling techniques.

1) DROP OFF UNNECESSARY DATA

In general, one of the primary challenges in HAR and pattern recognition techniques is the presence of imbalanced class distribution within the dataset, which may be associated with a particular tester (subject), activity (class), or session. This service implements the capability to exclude data from the input dataset. Specifically, this includes excluding data related to:

- a single subject or a group of subjects;
- a single activity or a group of activities;
- a single data collection session or a group of sessions.

2) TRAINING/TESTING APPROACH

When designing ML/DL models, data is usually split into two (training and testing) or three (training, validation, and testing) subsets [35]. The model is then trained on the training dataset (and cross-checked with the validation dataset when required) to make classifications/predictions for the testing dataset. However, in HAR, train and test subsets are usually partitioned by considering the presence of data collected from several different testing subjects. For such a purpose, B-HAR presents three different types of train/test-splitting approaches:

- Inter-subjects: The inter-subject train/test partitioning approach operates as follows: B-HAR users select a subset of subjects to create the testing dataset (e.g., 10 out of 100 total subjects), while the remaining subjects constitute the training dataset (e.g., 90 out of 100 total subjects). This approach ensures no overlap between the training and test datasets, thereby minimizing the possibility of overfitting and allowing for in-depth testing of the model’s generalizability capabilities.
• **Intra-subjects:** The intra-subject train/test partitioning approach functions as follows: B-HAR users employ a traditional hold-out method over the activity dataset. Initially, the dataset is divided into training and testing datasets (e.g., 75% train and 25% test). Subsequently, a classic k-fold cross-validation approach is applied solely to the training dataset. Finally, the trained model is evaluated on the testing dataset.

• **Inter-sessions:** In the inter-session train/test approach, B-HAR utilizes a subset of the data collection sessions performed by specific subjects as the training set and the remaining sessions as the testing set. However, this testing approach necessitates more sessions for each subject.

When dealing with HAR data, the best testing methodologies are inter-subjects or inter-session.

3) **SCALING**

During the training phase, features with higher values dominate the training process. However, these features do not necessarily reflect the dataset’s characteristics or the final accuracy of the pattern recognition model. Data scaling transforms multiscale data to a uniform scale, where all variables positively impact the model, thereby enhancing the stability and performance of the learning algorithm [35]. Besides, when working with datasets with different features representing every sample, the datasets perform independent scaling for every feature. B-HAR provides the following scaling techniques:

• **Robust scaling:** it scales each feature of the dataset $HAR_D$ by subtracting the median ($HAR_D^{Q_2}$) and then dividing by the Interquartile Range (IQR). The IQR is defined as the difference between the third and the first quartile ($HAR_D^{Q_3} - HAR_D^{Q_1}$). The robust scaler of a dataset $HAR_D$ is expressed as:

$$HAR_D^{norm} = \frac{HAR_D - HAR_D^{Q_2}}{HAR_D^{Q_3} - HAR_D^{Q_1}} \quad (1)$$

This scaler uses robust statistics for outliers, in contrast with the other scalers, which are highly affected by outliers statistics such as the maximum, the minimum, the mean, and the standard deviation.

• **Standard scaling** (z-Score): it maps the data into a distribution with a mean of 0 and a standard deviation of 1. Each normalized value is computed by subtracting the corresponding feature’s mean and dividing it by the standard deviation. The standard scaler of a dataset $HAR_D$ is expressed as:

$$HAR_D^{norm} = \frac{HAR_D - HAR_D^\text{mean}}{HAR_D^\text{std}} \quad (2)$$

where $HAR_D^\text{mean}$ is the mean of the training dataset, and $HAR_D^\text{std}$ is the standard deviation of the training datasets.

• **Min-Max normalization:** it rescales the feature to a fixed range by subtracting the minimum value of the feature and then dividing by the range:

$$HAR_D^{norm} = \frac{HAR_D - HAR_D^{\min}}{HAR_D^{\max} - HAR_D^{\min}} \quad (3)$$

In particular, a normalizer is applied to the training dataset, and subsequently, the same normalizer is employed to normalize the testing data. However, a common mistake made by researchers new to the HAR and ML/DL field is performing scaling on the original dataset before the train/test split.

4) **FEATURES SELECTION/REDUCTION**

A higher number of features does not necessarily correlate to better outcomes in the pattern recognition model. This is because features can either positively or negatively affect the recognition process. To address this issue, feature selection techniques are employed to identify and prioritize features based on their importance. These techniques automatically choose features from a training dataset that contribute most to prediction accuracy, thereby decreasing the model’s reliance on irrelevant features. Additionally, eliminating features reduces the time required for the training and testing phases. The primary advantages of feature selection techniques include: i) mitigating overfitting by removing redundant data, thereby reducing noise-related errors; ii) enhancing accuracy by eliminating misleading data; and iii) decreasing training time due to fewer data points [35], [36].

Within this service, B-HAR integrates the following feature selection/reduction techniques:

• **Variance:** It removes all features whose variance is lower than a defined threshold. By default, it removes zero-variance features, i.e., features with the same value in all samples.

• **Recursive Features Elimination (RFE):** It fits a model and removes the less critical feature (or features) until a defined number of features is reached without knowing how many features are valid. Features are ranked based on their importance and by recursively eliminating a small number of features per loop. Besides, RFE attempts to eliminate dependencies and collinearity that may exist in the model. Moreover, to determine the optimal number of features, k-fold cross-validation is used with RFE to score different feature subsets and select the best-scoring collection.

• **Lasso regularization (L1):** The Lasso regularization linear model estimates sparse coefficients starting from the non-zero coefficients returned by a linear regression model, thus effectively reducing the number of features the given solution depends on.

• **Tree-based:** This function calculates impurity-based feature importance, identifying and removing irrelevant features in conjunction with other feature selection techniques. Tree-based estimators inherently establish an ordering of the features based on the training dataset, rendering them highly suitable for feature selection methods.
B-HAR offers various techniques to address such challenges:  

- **Highly Correlated Features**: This technique involves calculating the correlation matrix of the input features, providing insights into the pairwise correlation between features, and allowing users to identify which features are highly correlated. Eliminating highly correlated features helps to reduce redundancy and improve the efficiency of the HAR model. It also helps to alleviate the curse of dimensionality, which can lead to overfitting and reduced generalization performance.

- **Adaptive Synthetic (ADASYN) Oversampling**: ADASYN employs a weighted distribution for different minority class observations based on their learning phase difficulty. It then enriches less populated classes with new synthetic observations that are easier to learn compared to the original observations from the most populated class.

5) **BALANCING**

Training a HAR model on an imbalanced dataset can pose unique challenges throughout the model training process, yielding a biased recognition model, mainly when the data are not linearly separable. However, two categories of techniques address this concern in the state-of-the-art: i) oversampling the less populated classes to match the number of observations of the most populated class, and ii) undersampling the most populated classes to match the number of observations of the less populated class [37]. B-HAR offers various techniques to address such challenges:

- **Random Undersampling**: This technique randomly eliminates a subset of observations from the most populated classes so that all classes have the same population equal to that of the initially least populated class. Its main limitation is that the removed observations can be more informative than the ones kept.

- **Near Miss Undersampling**: When given two observations belonging to different classes and are very similar, this method eliminates the observation from the most populated class.

- **Edited Nearest Neighbors Undersampling**: This approach removes observations whose actual class label differs from the class of at least \( k \) nearest neighbors. It can undersample indiscriminately across all existing classes or a subset of the most populated classes.

- **Random Oversampling**: It randomly duplicates observations from the less populated classes so that all classes have the same population, equal to that of the initially most populated class. However, the main limitation is that the duplicated observations may lead to an overfitting problem.

- **Synthetic Minority Over-sampling Technique (SMOTE) Oversampling**: This technique randomly selects an observation \( a \) from the less populated class and identifies its \( k \) nearest neighbors from the same class. A synthetic observation \( s \) is then generated as a convex combination of \( a \) and an observation \( b \) randomly selected from its \( k \) nearest neighbors.

- **K-means SMOTE Oversampling**: This method utilizes the well-known k-means unsupervised learning algorithm to generate new observations of the less populated class in safe and crucial input dataset areas. It avoids generating noise and effectively addresses imbalances between and within classes.

**E. MODEL TRAINING AND TESTING**

In such a service, B-HAR, starting from the pre-processed input dataset, takes a known set of input data and responses (training dataset) and trains a model to generate reasonable predictions of new data (testing dataset). B-HAR offers users a comprehensive variety of pattern recognition models, including seven traditional machine learning models: k-Nearest Neighbours (kNN), weighted kNN (wkNN), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Support Vector Machine (SVM), Random Forests (RF), Decision Trees (DT) and three deep learning models: Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Recurrent Neural Network (RNN). Beyond simply selecting a model, B-HAR introduces hyper-models. These predefined deep learning architectures can be tailored by users, which can define their hyperparameters (e.g., number of nodes per layer, activation functions, loss function, dropout rate, etc.), providing the models the capability to suit specific Human Activity Recognition tasks.

To enhance model performance, B-HAR incorporates a grid search feature. This functionality systematically explores various hyperparameter combinations, reducing the need for manual tuning. In addition, B-HAR allows user-defined deep learning models, increasing and allowing comparison between different user-defined and state-of-the-art models. Furthermore, the framework allows users to download trained models easily.

Nevertheless, it is essential to clarify that B-HAR’s primary objective is not to pinpoint the optimal recognition model or attain the highest performance outcomes. Instead, it aims to serve as a valuable tool in delineating the HAR data processing workflow, enabling a fair comparison between various models and workflows.

**F. PERFORMANCE MEASURES**

Typically, the performance evaluation of a HAR model relies heavily on accuracy as its primary metric. However, accuracy may not provide precise insights, especially in unbalanced training or multi-class scenarios. Thus, to address this limitation, more representative metrics derived from the confusion matrix are often employed [39]. The confusion matrix offers a clear visualization of the model’s performance. In the context of multi-class classification, the mathematical representation of a confusion matrix is as follows: rows correspond to instances predicted for each class. In contrast, columns
correspond to instances belonging to each actual class.

\[ C \equiv \begin{bmatrix} c_{11} & \ldots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \ldots & c_{nn} \end{bmatrix} \]

The confusion elements for each class are given by:
- true positives: \( tp = \sum c_{ii} \);
- false positives: \( fp = \sum_{i=1}^{n} c_{i} - tp_i \);
- false negatives: \( fn = \sum_{i=1}^{n} c_{i} - tp_i \);
- true negatives: \( tn = \sum_{i=1}^{n} \sum_{k=1}^{n} c_{ik} - tp_i - fp_i - fn_i \).

Based on the confusion matrix, B-HAR computes the following model quality metrics:

\[
P = \frac{tp}{tp + fp}, \quad S_p = \frac{tn}{fp + tn}, \quad S_e = \frac{tp}{tp + fn}, \\
A = \frac{tp + tn}{p + n}, \quad F1 = 2 \times \frac{P \times S_e}{P + S_e}.
\]

where \( P \) represent precision, \( S_p \) stands for specificity, \( S_e \) is sensitivity, \( A \) is accuracy and \( F1 \) is F1-Score. Additionally, to provide a clearer explanation of the results and the behavior of the tested models, B-HAR presents the trend of the loss function and the overall accuracy at each stage of the model training.

### G. DATA EXPLORATION

This service furnishes information and statistics regarding the input dataset. Specifically, this service enables users to:
- obtain a correlation matrix for the features of the dataset.
- analyzes the distribution of observations related to testers and activities, facilitating the detection of potential imbalance issues.
- access statistics for each sensor in the dataset if multiple sensors are present.
- visualize the boxplot distribution for all or specific activities or testers, aiding in identifying outlier distributions or noisy data.

### III. EXPERIMENTAL RESULTS

This section presents the results of an extensive experimental campaign conducted using B-HAR on seven of the most popular open-source HAR datasets and two dedicated datasets (i.e., Channel State Information (CSI) and Received Signal Strength Indicator (RSSI) used to recognize the occupancy (empty or occupied) status of an environment based on the radio signal propagation patterns). The campaign has two main objectives: i) testing the flexibility and ease of use of B-HAR on heterogeneous datasets concerning the number of subjects, types of activities, and types of sensors, and ii) evaluating the performance achieved by machine and deep learning models implemented in B-HAR. Table 3 presents the main characteristics of the datasets considered. Column 1 displays the datasets’ names [reference]. Columns 2, 3, and 4 indicate the number of testing subjects, activities, and used sensors, where A refers to accelerometers and G to gyroscopes. Column 5 denotes the data sampling frequency, and finally, column 6 presents the dimension of the time window and overlapping fragment in seconds and references the article where such window and overlap dimension were utilized.

Concerning the experimental analysis, Table 4 presents the results obtained using B-HAR on the datasets listed in Table 3, employing six out of the ten pattern recognition models introduced in Section II-E. The gray rows in Table 2 highlight the configuration of B-HAR utilized in the experiments. The initial block of Table 4 reports the results employing data segmentation. Conversely, the second block of Table 4 utilized feature extraction in both the time and frequency domains.

Upon examining the results in Table 4, users can discern which model performs optimally for each metric across the various datasets considered. Naturally, users can furnish B-HAR with new datasets and HAR models, configuring them according to different workflows. Thus, B-HAR is proposed as a foundational framework for impartially comparing existing and novel HAR approaches, facilitating the identification of the most accurate approach tailored to the target dataset.

### IV. HAR DESIGN WORKFLOW ISSUES

In this article, we presented an overview of the fundamental processing steps with a primary focus on the definition of the correct order in which such steps have to be applied (Figure 2) since many methodologies published in the existing literature are complex to be replicated and do not provide precise information regarding the processing steps.

In the following, we discuss the essential information that HAR methodologies need to provide since comparing solutions with different experimental setups is challenging because accuracy and other factors depend on the performed experimental processing steps.

Generally, detailed information related to the collected data and the experimental setup is not provided. However, a complete description of the collected data is of primary importance. The following information concerning the input dataset needs to be provided:
- number of testing subjects, gender, age, height, weight;
- existing pathologies, left-handed or right-handed;
- number of testing subjects, gender, age, height, weight;
B-HAR results on segment/features data representation of the most famous HAR datasets.

| Dataset | kNN | LDA | QDA | RF | DT | CNN |
|---------|-----|-----|-----|----|----|-----|
| WISDM v1 | [98.61, 62.61, 62] | [95.12, 12, 12, 12] | [95.20, 17.20, 17.20] | [99.76, 76, 76, 76] | [97.55, 55, 55, 55] | [96.26, 26, 26, 26] |
| WISDM v2 | [94.69, 78, 69, 78] | [64.40, 34, 40, 34] | [86.58, 58, 68, 58] | [96.90, 91, 90, 91] | [92.77, 77, 77, 77] | [60.62, 62, 62, 60] |
| DAPHFNET | [16.90, 87, 90, 88] | [90.91, 89, 31] | [92.15, 19, 15, 19] | [93.80, 83, 80, 83] | [91.60, 60, 60, 60] | [90.73, 76, 73, 73] |
| PAPAM | [90.65, 66, 65, 66] | [90.45, 45, 45, 45] | [98.89, 89, 88, 89] | [93.67, 66, 66, 66] | [96.84, 84, 84, 84] | [96.84, 84, 84, 84] |
| HHAR (phone) | [96.83, 85, 83, 83] | [88.43, 43, 43, 43] | [88.40, 40, 50, 50] | [96.49, 49, 49, 49] | [96.83, 83, 83, 83] | [96.67, 67, 67, 67] |
| HHAR (watch) | [95.78, 82, 78, 82] | [90.54, 52, 54, 52] | [94.26, 26, 26, 26] | [97.85, 85, 85, 85] | [94.69, 69, 69, 69] | [96.83, 83, 83, 83] |
| mHealth | [81.76, 81, 76, 81] | [75.38, 38, 58, 39] | [91.82, 82, 91, 82] | [73.85, 85, 85, 85] | [74.77, 77, 77, 77] | [65.80, 80, 80, 80] |
| RSSI | [91.91, 91, 91, 91] | [91.91, 91, 91, 91] | [91.91, 91, 91, 91] | [91.91, 91, 91, 91] | [91.91, 91, 91, 91] | [91.91, 91, 91, 91] |
| CSI | [93.93, 93, 93, 93] | [93.93, 93, 93, 93] | [92.92, 92, 92, 92] | [93.93, 93, 93, 93] | [93.93, 93, 93, 93] | [92.92, 92, 92, 92] |

Classification results are shown in terms of [Specificity, Sensitivity, Precision, Accuracy, F1-Score].

Recommended HAR methodologies characteristics.

| Rec. | Info. related to | Info. to provide |
|------|-----------------|-----------------|
| R1   | Data source devices | Manufacturer, model, cost, OS/industrial version |
| R2   | Sensor model | Manufacturer, model, static characteristics, sampling frequency |
| R3   | Dataset information | Number of samples per subject, activities, and sessions |
| R4   | Data errors | Used handling errors technique |
| R5   | Noise | Detailed filtering information |
| R6   | Segmentation | Window/overlap size and discussion concerning such choice |
| R7   | Feature extraction | Features domain and reference to the used features |
| R8   | Train/Test approach | Inter-subject, intra-subject/session and precise split parameters (i.e., number of subjects, samples per subject and class) |
| R9   | Normalization | Definition of the used technique and discussion concerning such choice |
| R10  | Feature selection | Definition of the used technique and discussion concerning such choice |
| R11  | Balancing | Definition of the used technique and discussion concerning such choice |
| R12  | HAR model | Precise description concerning tested model and its configuration |
| R13  | Performance metrics | Definition of the performance metrics and discussion of the achieved results |

This last point is crucial since the datasets present many activities for each session, so the transition between activities needs to be handled.

Beyond the issues concerning the data and the data collection approach, a significant issue in the proposed methodologies concerns the missing details. Table 5 provides 13 basic recommendations related to the essential information authors should provide when proposing a HAR methodology. Column 2 presents the item of interest. Instead, Column 3 presents the recommendations related to the item.

B-HAR aims to not standardize the HAR workflow since the performance of the generated model is strongly related to the data source (i.e., R1 to R3), the single processing steps (i.e., R4 to R11), and the configuration of the HAR model (i.e., R12). As for the performance measures (i.e., R13), it is widely acknowledged that new performance measure metrics (e.g., robustness against adversarial attacks or true generalizability across datasets) that measure the generalization capabilities of the models are needed. Based on such recommendations, B-HAR can reproduce the analysis workflow easily and minimize user errors, such as due to wrong preprocessing steps in application order.

V. CONCLUSION AND FUTURE EXTENSIONS

HAR based on time series data includes several processing steps that affect the results’ quality and correctness. However, no clear pipeline illustrates how such steps have to be applied. Thus, researchers might make elementary mistakes that affect the quality of the activity recognition model. This article proposed B-HAR, a service that facilitates the study of the behavior of the most famous pattern recognition models in the context of HAR. B-HAR allows the users to define elaboration pipelines in the correct order, including different steps that affect the results’ quality and correctness. However, no clear pipeline illustrates how such steps have to be applied. Thus, researchers might make elementary mistakes that affect the quality of the activity recognition model. This article proposed B-HAR, a service that facilitates the study of the behavior of the most famous pattern recognition models in the context of HAR. B-HAR allows the users to define elaboration pipelines in the correct order, including different
pre-processing steps, such as noise removal, segmentation, feature extraction, normalization, feature selection, and balance. Moreover, users can utilize B-HAR to test self-designed HAR models by reusing already-designed HAR workflows. This reduces the likelihood of erroneous design in HAR pipelines and furnishes users with a foundational framework for equitably comparing resultant outcomes. Future works will encompass the development of a Graphical User Interface (GUI), augmentation of each micro-service with an expanded array of cutting-edge data processing techniques, incorporation of new recognition, regression, and transfer learning models, and integration of novel performance metrics.

REFERENCES

[1] 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. 2019.

[2] A. Poli, G. Cosoli, L. Scalise, and S. Spinsante, “Impact of wearable measurement properties and data quality on ADLs classification accuracy,” IEEE Sensors J., vol. 21, no. 13, pp. 14221–14231, Jul. 2021.

[3] F. Demozri, C. Turetta, P. H. Kindt, F. Chiarani, R. Bacchin, N. Valè, F. Pascucci, P. Cesari, N. Smian, T. Tamburin, and G. Pravadelli, “A low-cost wireless body area network for human activity recognition in healthy life and medical applications,” IEEE Trans. Emerg. Topics Comput., vol. 11, no. 4, pp. 839–850, Oct./Dec. 2023.

[4] World Population Prospects 2019: Highlights, Dept. Econ. Social Affairs, United Nations, San Francisco, CA, USA, 2019.

[5] J. R. Villar, S. González, J. Sedano, C. Chira, and J. M. Trejo-Gabriel-Galan, “Improving human activity recognition and its application in early stroke diagnosis,” Int. J. Neural Syst., vol. 25, no. 4, Jun. 2015, Art. no. 1450036.

[6] J. Goyal, P. Khandnor, and T. C. Aseri, “Classification, prediction, and model personalization in sensor-based human activity recognition,” IEEE Trans. Emerg. Topics Comput., vol. 7, no. 11, pp. 11137–11146, Nov. 2020.

[7] J. Lu, X. Zheng, M. Sheng, J. Jin, and S. Yu, “Efficient human activity recognition using a single wearable sensor.” IEEE Internet Things J., vol. 7, no. 11, pp. 11317–11346, Nov. 2020.
[39] D. M. Powers, “Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation,” 2011, arXiv:2010.16061.

[40] 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, Mar. 2011.

[41] G. M. Weiss, K. Yoneda, and T. Hayajneh, “Smartphone and smartwatch-based biometrics using activities of daily living,” IEEE Access, vol. 7, pp. 133190–133202, 2019.

[42] J. W. Lockhart, G. M. Weiss, J. C. Xue, S. T. Gallagher, A. B. Grosner, and T. T. Pulickal, “Design considerations for the WISDM smart phone-based sensor mining architecture,” in Proc. 5th Int. Workshop Knowl. Discovery Sensor Data, Aug. 2011, pp. 25–33.

[43] M. Bachlin, M. Plotnik, D. Roggen, I. Maidan, J. M. Hausdorff, N. Giladi, and G. Troster, “Wearable assistant for Parkinson’s disease patients with the freezing of gait symptom,” IEEE Trans. Inf. Technol. Biomed., vol. 14, no. 2, pp. 436–446, Mar. 2010.

[44] F. Demrozi, R. Bacchin, S. Tamburin, M. Cristani, and G. Pravadelli, “Toward a wearable system for predicting freezing of gait in people affected by Parkinson’s disease,” IEEE J. Biomed. Health Inform., vol. 24, no. 9, pp. 2444–2451, Sep. 2020.

[45] A. Reiss and D. Stricker, “Introducing a new benchmarked dataset for activity monitoring,” in Proc. 16th Int. Symp. Wearable Comput., Jun. 2012, pp. 108–109.

[46] O. Banos, R. Garcia, J. A. Holgado-Terriza, M. Damas, H. Pomares, I. Rojas, A. Saez, and C. Villalonga, “mHealthdroid: A novel framework for agile development of mobile health applications,” in Proc. Int. Workshop Ambient Assist. Living. Belfast, U.K.: Springer, Dec. 2014, pp. 91–98.

[47] L. T. Nguyen, M. Zeng, P. Tague, and J. Zhang, “Recognizing new activities with limited training data,” in Proc. ACM Int. Symp. Wearable Comput., 2015, pp. 67–74.

[48] F. Demrozi, F. Chiarani, and G. Pravadelli, “A low-cost BLE-based distance estimation, occupancy detection and counting system,” in Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE), Feb. 2021, pp. 1430–1433.

[49] F. Demrozi, C. Turetta, F. Chiarani, P. H. Kindt, and G. Pravadelli, “Estimating indoor occupancy through low-cost BLE devices,” IEEE Sensors J., vol. 21, no. 15, pp. 17053–17063, Aug. 2021.

CRISTIAN TURETTA (Graduate Student Member, IEEE) received the B.S. and M.E. degrees in computer science and engineering from the University of Verona, Italy, in 2017 and 2020, respectively, where he is currently pursuing the Ph.D. degree with the Department of Computer Science, where he is a member of the IoT4Care research group working on Ambient Assisted Living and the Internet of Things.

FLORENC DEMROZI (Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science and engineering and the Ph.D. degree in computer science from the University of Verona, Italy, in 2014, 2016, and 2020, respectively. Currently, he is an Associate Professor of biomedical engineering with the Department of Electrical Engineering and Computer Science, University of Stavanger, Norway. In his current role, he focuses on research areas, such as the Artificial Intelligence of Things (AIoT), human activity recognition (HAR), active and assisted living (AAL), and the Internet of Medical Things (IoMT).

GRAZIANO PRAVADELLI (Senior Member, IEEE) received the Ph.D. degree in computer science. He has been an IFIP 10.5 WG Member and a Full Professor of information processing systems with the Computer Science Department, University of Verona, Italy, since 2018. In 2007, he cofounded EDALab s.r.l., an SME working on the design of IoT-based monitoring systems. His main interests include system-level modeling, simulation and semi-formal verification of embedded systems, and on their application to develop IoT-based virtual coaching platforms for people with special needs. In the previous contexts, he collaborated in several national and European projects and he published more than 120 papers in international conferences and journals.

Open Access funding provided by ‘Università degli Studi di Verona’ within the CRUI CARE Agreement