University of Milano - Bicocca, Milan, ITALY
Department of Informatics, Systems and Communication
Master Degree in Computer Science

Model-Centric and Data-Centric AI for Personalization in Human Activity Recognition

Supervisor: Prof. Paolo Napoletano

Co-Supervisor: Prof. Daniela Micucci

Master's Degree Thesis of:
Hamza Amrani
Mat. 807386

Academic Year 2020-2021

Computer science is no more about computers than astronomy is about telescopes.

Edsger Dijkstra

Abstract

Current sensor-based Human Activity Recognition (HAR) techniques that rely on a user-independent model struggle to generalize to new users and on to changes that a person may do over time to his or her way of carrying out activities. Moreover, the performances of subject-independent models are proportional to the size and the variability of the dataset used to generate it. Incremental Learning is a technique that allows obtaining personalized models that may improve the classifiers' performance thanks to continuous Learning based on user data. However, continuous user interaction is essential to provide ground-truth labels but could harm his/her experience and fall into some incorrect data annotation.

This work aims to: *i*) combine Deep Learning techniques with incremental Learning to obtain personalized models that perform better concerning user-independent models and personalized models obtained using traditional machine learning techniques; *ii*) homogenize existing and heterogeneous datasets into a common measurement configuration and standard annotation, in order to train an optimal user-independent model and to make available to the scientific community large datasets of homogeneous signals, enriched, when possible, with context information; *iii*) define an Unsupervised end-to-end learning approach for the problem of HAR from mobile devices to annotate unlabelled data without user cooperation. The experimentations were done separately, for each proposed approach, by comparing the obtained results with similar methods in the literature and showed the effectiveness of Model-Centric and Data-Centric strategies applied to personalize models in Human Activity Recognition.

This work is the result of the collaboration between the *Imaging and Vision Laboratory (IVL)* and the *Software Architecture Laboratory (SAL)*.

This labour has contributed to the scientific community with the following publications:

[1] Hamza Amrani, Daniela Micucci, and Paolo Napoletano. Personalized models in human activity recognition using deep learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 9682–9688, 2021. doi: 10.1109/ICPR48806.2021.9412140.

[2] Hamza Amrani, Daniela Micucci, Marco Mobilio, Paolo Napoletano. Homogenization of Existing Inertial-Based Datasets to Support Human Activity Recognition. In *NeurIPS Data-Centric AI Workshop (DCAI 2021), Virtual.*
(currently under review)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

A journey of a thousand miles begins with a single step.

Confucius

Artificial Intelligence (AI) is increasingly applied in different domains, such as healthcare and signal processing, making it one of the most active research globally. According to *Andrew Ng*, in his Seminary [1], almost all AI research focuses more on *Model-Centric AI* instead of *Data-Centric AI*, affirming that Data is the critical infrastructure necessary to build Artificial Intelligence (AI) system. In his opinion, to achieve a good AI solution, a careful balance between Model-Centric a Data-Centric perspective is fundamental. To prove it, he and his team have demonstrated, with a couple of experiments with real-world data, that investing in improved existing data quality is as effective as collecting the triple amount of the data. So, a combination of Model-Centric and Data-Centric AI is needed and can be summarized as follow:

$$AI \;\; System = Code + Data \tag{1.1}$$

where *Code* means model/algorithm.

To achieve a Data-Centric approach, a few questions can be answered, for example: *Is the Data complete? Is the data relevant for the use case? If labels are available,*

are they consistent? Is the presence of heterogenous differences impacting the performance? Do I have enough data?
These interrogatories are expected to be answered *iteratively*, just like in a Model-Centric approach.

In this *thesis*, Model-Centric and Data-Centric AI are analyzed and used to propose a procedure able to personalize models in Human Activity Recognition with Deep Learning approaches.

Human Activity Recognition (HAR) is an active research field that studies methods and techniques able to identify activities of Daily Living (ADLs) automatically. Recently, research on sensor-based HAR techniques has focused on the use of inertial signals acquired through mobile devices such as smartphones [2, 4, 7, 8, 9, 10]. This research shift is due to several factors. Modern mobile devices are characterized by high computing capabilities, large storage capacity, built-in sensors that measure motion, orientation, and various environmental conditions, and wide network connectivity. Moreover, smartphones are largely used by the population, and thus, no additional cost is required for setting up the hardware equipment. Recent statistics show that more than 5.19 billion people (67% of the world's population) use mobile phones, with user numbers up by 124 million (2.4%) over the past year [11]. Due to the growing use of mobile and wearable devices, HAR can be extensively used in several domains. For example, in healthcare to keep track of older people or the rehabilitation process after accidents and injuries, or in the guide to safe travel [12].

In literature, three approaches are mainly adopted: *subject-independent*, *subject-dependent*, and *hybrid*.

The *subject-independent model (or impersonal)* does not use the end-user data to develop the activity recognition model. It is based on the definition of a single activity recognition model that must be flexible enough to generalize the diversity between users. It should be able to have good performance once a new user is to be classified.

The *subject-dependent (or personal)* model only uses the end-user data to develop

the activity recognition model. The specific model, being built with the final user's data, can capture her/his characteristics; thus, it should well generalize in the real context. The defect is that it must be implemented for each end-user.

The *hybrid model* uses the end-user data and the other users' data to develop the activity recognition model. The classification model is trained both on the users' data and on the final user data. The idea is that the classifier should recognize easier the activity performed by the final user.

Generally, the recognition of Human Activities is carried out using the called *user-independent recognition models*. These models are usually generated by machine learning techniques that exploit data available from multiple users for the training phase without considering the end-user data.

Initially, traditional Machine Learning techniques were used to generate user-independent models. Support Vector Machines, k-Nearest Neighbor, and Decision Tree are only a few examples [13]. During the last years, due to the successful application of Deep Learning techniques in the computer video area [6], researchers start experimenting with deep learning techniques in the sensor-based HAR area [14].

Deep Learning techniques have proven to be more efficient than traditional ones in classifying ADLs [14]. In particular, it has been demonstrated the ability of Convolutional Neural Networks (CNNs) to detect spatial and temporal dependencies between signals. However, these methods require a specific hardware setup, Graphical Processing Units (GPUs) to speed up computation, and a significant amount of data to avoid overfitting during the training phase [5].

Whether traditional Machine Learning or Deep Learning techniques are used, the end-user data used to train the model must be known in advance to get a model with good performances. Indeed, user-independent models struggle to generalize 1) to new users and 2) to changes in how a pre-existing user performs an activity. In the first case, the difficulty in generalizing is due to the *inter-subject variability*, which either refers to anthropometric differences of body parts or to personal styles in accomplishing the activity (in other words, different subjects may differently perform

the same activity); in the second case it is due to the *intra-subject variability*, which represents the random nature of each class of activity due to pathological conditions or environmental factors (in other words, a subject never performs the same activity in the same exact way) [15, 9, 16, 17].

These factors make a constantly updated *subject-dependent model* the ideal solution so that both inter- and intra-subject variability can be addressed effectively [14].

An interesting approach to get a subject-dependent model is *Incremental Learning* [18, 19, 20]. It consists in learning new information to personalize HAR models without losing previously achieved knowledge. The application of Incremental Learning in sensor-based Machine Learning algorithms leads to several advantages. Among them, Incremental Learning may improve the recognition of individual users, even if they change how they perform ADLs. It allows to train models from existing ones, thus decreasing computational times and hardware setup limits. So, a personal updated recognition model can be obtained by training the subject-independent model with the new user data (inter-variability) and keeping the personal model constantly updated with new data for the existing user (intra-variability).

Although personal models work better than user-independent models, each user must have his/her own model (loss of generality) and must actively cooperate in the generation and maintenance phases with the risk of having a negative impact on his/her experience [21].

An optimal subject-independent model must generalize well due to the presence of inter-and intra-subject variability data. So, a model must be trained on a dataset with a high number of samples and users to have a significant inter and intra-subject variability of data. Some researchers proposed their dataset related to HAR by collecting data from different users and different contexts. However, these datasets are *heterogeneous*, and their standardization in a single unified dataset requires considerable effort. For example, signals are expressed in different units of measure, they may include gravity or not, and signals have different acquisition frequencies. Furthermore, labels are not aligned with a standard dictionary and sometimes have different meanings among other datasets. The lack of large inertial datasets on several sub-

jects [22, 3] is a well-known problem in sensor-based HAR. Since acquiring labelled time series is a costly procedure in terms of resource, time, and people involved, *integrating* existing datasets may be a solution despite the substantial heterogeneity of the data.

The *Continuous Learning Platform (CLP)* [3] is a platform that semi-automatically integrates heterogeneous data and provides them in a homogeneous form. It proposes to:

- harmonize heterogeneous data from inertial sensors, being them already existing datasets or coming from online acquisitions;
- distribute sets of labelled signals according to specific requests, such as data related to a specific activity, data from subjects with specific physical characteristics, or, more generally, data from subjects that performed a set of specified activities;
- distribute activity recognition models;
- provide an online service by associating labels to series of signals in real-time.

The functionalities proposed by the CLP platform can be helpful in several contexts. For example, researchers can use labelled inertial signals to validate new ADLs recognition techniques. Activity recognition models can be integrated into existing domain-dependent applications that require ADLs recognition in order to provide application functionalities (e.g., estimation of energy expenditure [23, 24], monitoring the development of Parkinson's disease [25], and early detection of dementia [26]). Finally, suppose an application that requires identifying the ADLs performed by the user and that does not include an activity recognition model. In that case, it can rely on the service the platform provides each time a new series of Data is acquired.

However, despite the use of Continuous Learning Platform to train an optimal subject-independent, the problem of *inter-subject variability* remains due to anthropometric differences of body parts or personal styles in accomplishing the activity.

So, user cooperation in acquiring correct data labels to help during the personalization procedure is essential. However, the continuous interaction could harm his/her experience and fall into some incorrect data annotation. A valid approach in these circumstances is *unsupervised learning* [27], which permits the use of collected unlabelled raw data without any interaction with the user. The system labels them at the expense of lower recognition performances. However, this approach allows focusing more deeply on raw signals instead of their labels.

In Unsupervised Learning [27], no accurate annotations are associated with the training data. The focus is on drawing inferences from the data and modelling the underlying structure and the distribution. It is assumed that similar patterns occur more often than others related to the output values to be predicted. When hidden patterns are found in the groups of the training data, groups of similar physical activities may have been recognized, identifying a *cluster*. Therefore, good representative features are essential in order to build an effective clustering algorithm. *AutoEncoder(AE)* [28] is an Unsupervised technique of Neural Networks (NN) that can learn compressed data representations of input data. First, an encoder encodes the input data to a latent state representation of the data, and a decoder reconstructs the representation back to the input data through the network. The representative embedded features can be used to identify clusters of similar data instances. *Centroid-based K-means Clustering* [29, 30] is one of the most used approaches due to its speed and accuracy in several domains. Similarly, *Hierarchial Agglomerative Clustering (HIER)* [31, 32] builds clusters, first considering each data instance as a separate cluster. Differently from K-means, in HIER, the number of clusters is not needed.

The aim of this work is to solve the problem of personalization models in HAR using Deep Learning approaches, using Model-Centric and Data-Centric solutions. *In this thesis,* the definition of a semi-automatic procedure that combines Incremental Learning and Deep Learning techniques, considering their effectiveness in solving individually the problem for which they have been designed, is proposed. Conse-

quently, an analysis of the procedure has been done, revealing the presence of some crucial points: the definition of an optimal subject-independent model and the continuous user interaction to annotate unlabelled data. After exploiting the heterogeneity problem of existing datasets and why the use of a single dataset is not a good choice, the Continuous Learning Platform has been employed to produce a uniform dataset with good inter and intra-subject variability. The aim is to define and implement a platform that integrates datasets of inertial signals to make large datasets of homogeneous signals available to *the scientific community*, enriched, when possible, with context information (e.g., characteristics of the subjects and device position). Finally, Unsupervised Learning has been adopted to annotate unlabelled data without user interaction and integrate unlabelled datasets or users' recordings to feed the uniformed dataset provided by the Continuous Learning Platform. In particular, a Deep Clustering architecture based on a recurrent AutoEncoder and a specific Clustering criterion has been defined and developed.

In *this labour*, three different problems can be identified: *i*) *the personalization of Human Activity Recognition model*; *ii*) *the homogenization of existing heterogeneous inertial-based HAR datasets*; *iii*) *the problem of Unsupervised Learning applied to inertial HAR signals.*

Those three contents were individually studied and analyzed, providing valid approaches to solve them. Consequently, this *thesis* has been divided into three *Chapters*: *Chapter 2* describes the procedure to obtain a personalized recognition model; *Chapter 3* define the homogenization procedure and the Continuous Learning Platform; *Chapter 4* provides the proposed Unsupervised Learning approach.

Each Chapter presents a literature review of state of the art, then describes the materials, methods, and procedures adopted to validate the approach, and finally discusses the experimentation results.

Chapter 2

Personalized Models using Deep Learning

You should have personalized genomics, personalized physiology, personalized medicine, where each person's different, and each body is an integrated whole.

George M. Church

Commonly, the recognition of human activities is carried out using the user-independent recognition models. However, they struggle to generalize 1) to new users and 2) to changes in how a pre-existing user performs an activity. These factors make a constantly updated *subject-dependent model* the ideal solution for effectively addressing inter and intra-subject variability. The former means that different people perform the same activity in different ways, so a bijective association between signal and activity performed does not exist; the latter means that fundamentally different classes show very similar characteristics in terms of sensor data [15, 9, 16, 17].

Aim of this work is to experiment with the effectiveness of an approach that combines Incremental Learning and Deep Learning techniques, considering their effectiveness in solving individually the problem for which they have been designed.
In particular, the results obtained by Siirtola et al. in [21] have been replicated to consider them the baseline. Siirtola et al. used Learn++ to personalize the models,

three classifiers (to make a comparison), and three different datasets. Since the features were not sufficiently described to replicate them, the ones described by Ferrari et al. have been used [5].
The neural networks chosen for the experimentation are a Residual Neural Network (ResNet) and a Simplified Convolutional Neural Network (S-CNN). ResNet was chosen because it is one of the most performing CNN in literature [33]. The S-CNN has been designed ad hoc to recognize HAR signals and has been chosen because it is computationally less expensive than the ResNet network.
Finally, the data of the three datasets have been augmented to balance them and obtain a reasonable amount of signals necessary for training the CNNs.
The experimentation showed that neural networks adapt faster to a new user with respect to Learn++.

The Chapter is organized as follow. First, a review of the state of the art of literature is provided. It follows an in-depth description of incremental learning and the Learn++ algorithm, followed by a description of the used datasets and models. Finally, the incremental learning procedure adopted is discussed and related experiments.

2.1 State of the Art

Recently, researchers have started studying solutions able to generate sensor-based recognition models that are *personalized* to the end-user. *Personalized recognition models* are a combination of user-independent and personal models that may improve the performance of the classifiers when news users enter the stage or previously known users change the way they carry out the activity [34, 21].

Several studies show that personalization contributes to improving recognition rates, and several methods have been proposed in the literature. For example, Fallahzadeh et al. [35] proposed an approach based on transfer learning algorithms and unsupervised learning. They demonstrated the improvement of personalization, show-

ing that the recognition performances of personalized models are much better than the recognition performances obtained using models that are not personalized.

Besides, Akbari et al. [36] suggested a personalization model based on deep learning and retraining methods. In their study, supervised personalization increased the recognition accuracy of a new user by 25

Ferrari et al. [2] experimented with Adaboost classifier after extracting 18 handcrafted features from signal data to improve the classification accuracy. They demonstrated that a classifier trained on personalized data is more powerful in terms of recognition accuracy than a classifier trained without personalization. Hossain et al. [37] proposed deep learning methods that use a deep reinforcement learning approach and a novel annotator selection model. Transfer learning techniques that require only a small amount of data from the target user have also been employed to generate personalized models [38]. Rokni et al. [39] demonstrated how the transfer learning approach obtains significant improvements on the accuracy of ADLs recognition with only a few labelled instances.

A very interesting approach is that of Siirtola et al. [21] that exploit *incremental learning* to generate a personalized model without requiring data from the target user. Incremental learning was previously used to personalize HAR models in Yu et al. [40], basing the approach on logistic regression models. In their study, a user-independent model was trained and then updated and personalized on personal data. The model update was based on supervised learning in the study, making the approach laborious for the user; in fact, much personal data is needed as their annotation. Similarly, Siirtola et al. [21] used the Learn++ algorithm [41] to personalize human activity recognition models with a different personalization approach. They proposed a fully autonomous personalization method based on unsupervised learning where the model update was based on personal data and predicted labels. This approach is easy for the user and permits continuous learning. However, it can quickly learn wrong due to unsupervised learning. In order to reduce the problem in the study, they also propose a supervised approach that consists in asking the user the correct annotation only if the posterior prediction is under a certain threshold.

2.2 Incremental Learning

Incremental Learning (see Figure 2-1) aims to develop artificially intelligent systems that can continuously address new tasks from new data while preserving knowledge learned from previously learned tasks. This approach is inspired by how humans acquire and integrate new knowledge. In the presence of new tasks to learn, we use knowledge from previous ones and integrate newly learned awareness into previous tasks.

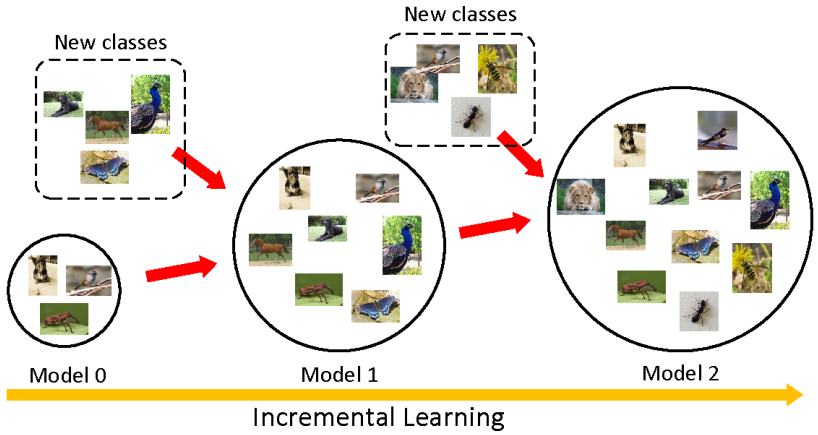

Figure 2-1: Example of Incremental Learning in Multiclass Classification on images.

In Incremental Learning, only data from a single task are presented to a learner in a sequence of training sessions. After each training session, the learner should perform all previously seen tasks on unseen data. So, IL methods learn from the current task data and prevent forgetting of previously learned tasks. The Figure 2-2 shows the process of Incremental Learning. The continuously collected data are divided into tasks, represented as $\{T_1, T_2, ..., T_n\}$. T_i indicates the *i-th* data task, and $T_i = \{(x_1^i, ..., z_1^i), (x_2^i, ..., z_2^i), ...\}$. The IL model M_i is trained with data coming from T_i and model M_{i-1}, as shown in Equation 2.1.

$$M_i = f(T_i, M_{i-1}) \tag{2.1}$$

In deep learning approaches, learning models incrementally may result in *catastrophic forgetting* (see Figure 2-3), a phenomenon where the performance of the original (old) task degrades dramatically [42]. So, a new task will override the previous

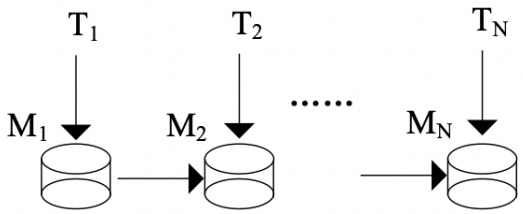

Figure 2-2: Incremental Learning procedure.

weights, degrading the model performance for the past tasks. Without fixing this problem, a neural network will not adapt itself to a continuous learning scenario because it forgets the existing knowledge when it learns new tasks. In Incremental Learning, Young Luo et al. [43] demonstrates that IL approaches can reduce and prevent catastrophic learning. They analyzed three Incremental Learning strategies to mitigate the problem, demonstrating their efficiency against deep learning approaches.

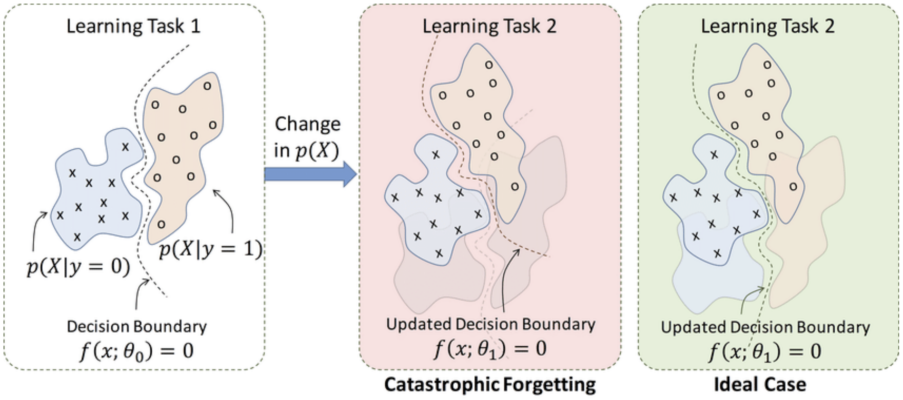

Figure 2-3: Catastrophic Forgetting.

The application of Incremental Learning in sensor-based machine learning algorithms leads to several advantages. Among them, it may improve the recognition of individual users, adapting it to new and changing conditions. Indeed, in HAR, this adaption refers to a new person's moving style and adapting to changes in a person's moving over time. Another advantage is that this adaption relies on model updating. It allows to train models from existing ones, thus decreasing computational times and hardware setup limits, avoiding model retraining.

In literature, different Incremental Learning approaches have been proposed [44]. One popular state of the art approach is importance weighting, that is a strategy that explicitly or implicitly reweight the observed samples to achieve greater robustness [44].

Also, several incremental SVM models exist [45]. Ensemble learning algorithms based on SVM [45] achieve Incremental Learning by training new classifiers for new batches of data and combining all existing classifiers for decision making. Many modern incremental approaches rely on a local partitioning of the input space and a separate classification/regression model for each partition. Different techniques can be used for partitioning, from kd-trees [46] to generic algorithms and adaptive Gaussian receptive fields [47]. Similarly, it is possible to use different local models, such as linear models [47], Gaussian mixture regression [46] or Gaussian Processes [48].

Another interesting approach relies on ensemble methods. They combine a collection of different models by a weighting strategy. They have proved to be particularly useful against the loss of a model's generalization ability after being trained on a new task. Some examples are incremental random forests [49] and ensembles of bipartite graph classifiers [50]. Learn++ algorithm [41] belongs to this category, and it will be described in the following subsection.

2.2.1 Learn++ Algorithm

The Learn++ algorithm [41] (see Figure 2-5) is based on ensemble of classifiers. It was inspired by the AdaBoost (adaptive boosting) algorithm [51] developed to improve the classification performance of weak classifiers. Learn++ generates an ensemble of weak classifiers. A combining ensemble of classifiers is specifically aimed at achieving Incremental Learning. Each new classifier added to the ensemble is trained using a set of examples belonging to a specific distribution. These classifiers are combined, Figure 2-4, with a weighted majority algorithm to obtain the final classification result. In particular, if a vast majority of weak classifiers agree on the class of a particular instance, considering their *posteriori* probabilities, this can be

defined as the algorithm having high or low confidence in the final decision.

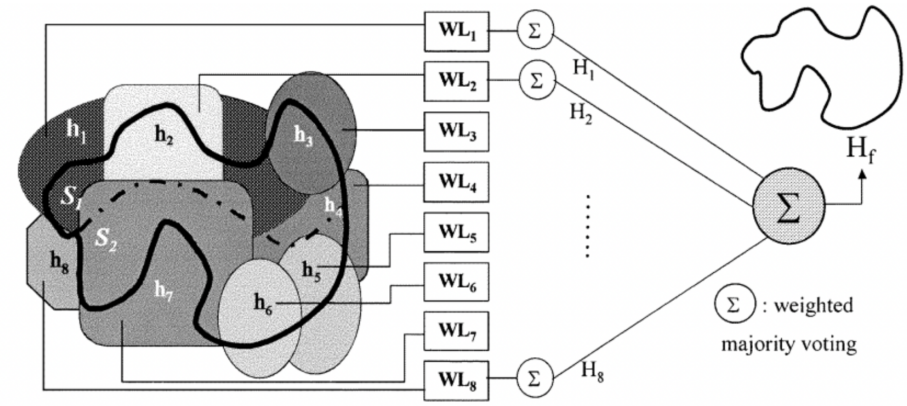

Figure 2-4: Combining Classifiers for Incremental Learning.

Learn++ can learn from new data even when the data introduces new classes. It does not require access to previously used data during subsequent training sessions and can retain previously acquired knowledge. So, to achieve Incremental Learning new classifiers are trained with new samples and combined to adapt the classification on new data.

Moreover, any classifier can be chosen as a base classifier, thus making it flexible compared to other Incremental Learning algorithms that force the choice of the classifier to be used. Among the Incremental Learning algorithms, Learn++ [41] results to be less complex. Although less complex, Learn++ is one of the most accurate online learning algorithm [52]. Because of its low complexity, Learn++ can also be implemented in devices with limited memory and reduced computational capacity, such as smartphones.

Algorithm Learn++

Input: For each database drawn from $\mathcal{D}_k$ $k=1,2,\ldots,K$

- Sequence of m training examples $S=[(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)]$.
- Weak learning algorithm WeakLearn.
- Integer T_k, specifying the number of iterations.

Do for $k=1,2,\ldots,K$:

Initialize $w_1(i) = D(i) = 1/m$, $\forall i$, unless there is prior knowledge to select otherwise.

Do for $t = 1,2,\ldots,T_k$:

1. Set $\boldsymbol{D}_t = \mathbf{w}_t \Big/ \sum_{i=1}^{m} w_t(i)$ so that $\boldsymbol{D}_t$ is a distribution.
2. Randomly choose training TR_t and testing TE_t subsets according to $\boldsymbol{D}_t$.
3. Call WeakLearn, providing it with TR_t.
4. Get back a hypothesis $h_t : X \rightarrow Y$, and calculate the error of h_t : $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$ on $S_t = TR_t + TE_t$. If $\varepsilon_t > 1/2$, set $t = t-1$, discard h_t and go to step 2. Otherwise, compute normalized error as $\beta_t = \varepsilon_t/(1-\varepsilon_t)$.
5. Call weighted majority, obtain the composite hypothesis $H_t = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} \log(1/\beta_t)$, and compute the composite error $E_t = \sum_{i:H_t(x_i)\neq y_i} D_t(i) = \sum_{i=1}^{m} D_t(i)[\![H_t(x_i)\neq y_i]\!]$. If $E_t > 1/2$, set $t = t-1$, discard H_t and go to step 2.
6. Set $B_t = E_t/(1-E_t)$ (normalized composite error), and update the weights of the instances:

$$w_{t+1}(i) = w_t(i) \times \begin{cases} B_t, & \text{if } H_t(x_i) = y_i \\ 1, & \text{otherwise} \end{cases} = w_t(i) \times B_t^{1-[\![H_t(x_i)\neq y_i]\!]}$$

Call weighted majority on combined hypotheses H_t and **Output** the final hypothesis:

$$H_{final} = \arg\max_{y\in Y} \sum_{k=1}^{K} \sum_{t:H_t(x)=y} \log\frac{1}{B_t}$$

Figure 2-5: Learn++ Algorithm.

2.3 Datasets and Pre-processing

In recent years, some researchers have published their own dataset related to HAR. However, these datasets are heterogeneous, and their standardization in a single unified dataset requires considerable effort. The availability of a dataset containing a large number of samples is a well-known issue both in the field of ADLs recognition from inertial sensors and in other domains [53].

Since benchmark datasets are not available, the experimentation has been done on three different public datasets, the same used by Siirtola et al. [21].

- *Human Activities and Postural Transitions* [54] (referred as *Anguita* in the following), which includes 3-axial linear acceleration, 3-axial angular velocity, and gyroscope sensor data of 11 ADLs. The signals were recorded with Samsung Galaxy S II smartphones, and the activities were performed by 30 volunteers (19-48 years). They performed six basic activities: three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs and walking upstairs). The experiment also included postural transitions that occurred between the static postures. All the participants were wearing a smartphone on the waist during the experiment execution. This is an updated version of UCI HAR [55].

- *Sensors Activity Recognition* [23] (referred as *Shoaib* in the following), which includes 3-axial acceleration, gyroscope, magnetometer, and linear acceleration sensor data of 7 ADLs. The signals were recorded with Samsung Galaxy SII smartphones, and the activities were performed by 10 volunteers (male, 25-30 years). Each of these participants was equipped with five smartphones on five body positions, in Figure 2-6.

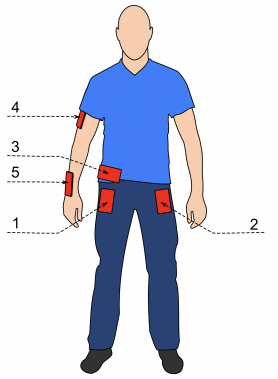

Figure 2-6: Shoaib Dataset: Overview of the phone positions.

- *Daily Activities* [56] (referred as *Siirtola* in the following), which includes 3-axial acceleration sensor data of 5 ADLs recorded with Nokia N8 smartphones.

The activities have been performed by 8 volunteers, whose ages varied from 25 to 34 years. They performed activities outside the laboratory. Subjects walked inside and outside, mainly on flat surfaces but also in a staircase. Streets, where subjects walked, ran, drove a car, and cycled, were normal tarmac roads, and subjects themselves determined the route and speed.

Anguita contains a high number of users (30) but is smaller than the other two datasets in terms of the number of activities for each user. Shoaib is more balanced because each user has the same number of samples. Siirtola instead has data from 8 users, but the number of activities per user is unbalanced.

Each dataset signal is composed of three accelerometer components along the x, y, and z-axis and the relative magnitude. Anguita and Shoaib datasets have been acquired at a frequency rate of 50Hz, instead the Siirtola dataset uses a sampling rate of 40Hz. No homogenization was necessary on the datasets because the experiments were performed on the individual datasets. All the signals have been divided in windows of 3.0s with an overlap between subsequent segments of 1.5s (50%).

Table 2.1: Number of segments and classes for each Dataset.

Dataset	**Segment size**	**# Segments**	**# Per user**	**# Classes**
Anguita	150x4	9,712	$\sim$ 324	11
Shoaib	150x4	41,930	4,193	7
Siirtola	120x4	6,921	$\sim$ 865	5

Table 2.2: Activities of Daily Living for each Dataset.

Dataset	**Activities of Daily Living**
Anguita	Standing, sitting, lying, walking, walking downstairs and walking upstairs
Shoaib	Walking, sitting, standing, jogging, biking, walking upstairs and walking downstairs
Siirtola	Idling, walking, cycling, driving a car, running

Table 3.1 shows the size and the total number of segments in the three datasets, and Table 2.2 reports the activities for each dataset. Except for Shoaib, the number of samples is low. For this reason, a data augmentation technique has been employed.

Moreover, the hand-crafted features have been extracted from each window to get more representative information between the time-series data than raw data.
OpenHAR [57] has been used to download the three datasets. OpenHAR is a MATLAB toolbox that provides access to data of different open human activity recognition datasets. In particular, OpenHAR provides quick access to data by providing off-the-shelf scripts for some of the most popular datasets.

2.3.1 Data Augmentation

Data augmentation techniques allow deep learning models to be more robust by producing data variations that the model may recognize in the real world. In fact, in HAR, the domain continues to diversify and increase rapidly due to inter and intra-subject variability of subjects. Instead of traditional data augmentation techniques, which usually include noise injection, flipping, etc., a new data augmentation technique based on multiple window segmentation has been designed. The idea is to apply a Δ time-shifting multiple times to obtain different windows from a given signal. This technique allows increasing the original samples more realistically compared to traditional techniques. Without augmentation, a given signal of length T is segmented in windows of $S = 3s$ with 50% of overlap. With augmentation, a window of $S = 3s$ is taken every $\Delta = 0.2s$ until $T - \Delta$. For example, considering a given signal of length 10s and a $\Delta = 0.2s$, this approach extracts windows considering multiple time intervals, such as 0-10s, 0.2-10s, 0.4-10s and so on. This technique allows increasing the original samples more realistically compared to traditional techniques.

Table 2.3: Number of segments and classes for each Dataset, after Data Augmentation process.

Dataset	**Segment size**	**# Segments**	**# Per user**	**# Classes**
Anguita	150x4	57,372	$\sim$ 1,912	11
Shoaib	150x4	251,230	25,123	7
Siirtola	120x4	41,351	$\sim$ 5,169	5

Table 2.3 shows the size and the total number of segments in the three datasets after the data augmentation process.

2.3.2 Hand-Crafted Features

Table 2.4: Hand-Crafted Features.

Features	
Minimum	$min = \min_{j=1,...n}(x_j)$
Maximum	$max = \max_{j=1,...n}(x_j)$
Mean	$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
Median	$Me = x_{0.5} : F(x_{0.5}) = 0.5$
Standard Deviation (SD)	$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$
Variance	$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$
Fourth central moment	$m_4 = \frac{1}{n}\sum_{j=1}^{n}(x - \bar{x})^4$
Fifth central moment	$m_5 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^5$
Skewness	$S = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{s^3}$
Kurtosis	$K = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{s^4}$
Root Mean Square (RMS)	$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$
Interquartile Difference	$ID = x_{0.75} - x_{0.25}$
Total Sum	$TS = \sum_{i=1}^{n} x_i$
Range	$R = max - min$
Entropy	$H(x) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$
SD of the intervals between two successive peaks	$SDNN = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(PP_j - \overline{PP})^2}$
RMS of the differences between two successive peaks	$RMSSD = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n-1}(PP_{j+1} - PP_j)^2}$
Number of pairs of successive peaks intervals that differ by more than 50 ms	$pNN50 = p(\lvert PP_{j+1} - PP_j\rvert > 50)$
Sum of the spectral power components	$SP = \frac{1}{n}\sum_{j=1}^{f} \lvert FFT_j\rvert^2$
Mean of the spectral components	$\mu_f = \frac{1}{n}\sum_{j=1}^{n} FFT_j$
Median of the spectral components	$Me_f = FFT_{0.5} : F(f_{0.5}) = 0.5$

The Hand-Crafted features introduced by Ferrari et al. [5, 58] are extracted from the x, y, and z accelerometer segments and the magnitude of the segments. They demonstrated that deep learning solutions reach optimal performances, even when large scale datasets are not available. Moreover, hand-crafted features are preferable in cases where the hardware is low cost and thus does not permit deep learning solutions to run in a short time.

Table 2.4 reports details about the 21 features. The total number of features extracted from the x, y, z and magnitude segments is 84.

2.4 Convolutional Neural Networks

Classical approaches to Human Activity Recognition (see Figure 2-7) problem involve hand crafting features from the time series data based on fixed-sized windows and training machine learning models, such as SVMs and decision trees. The difficulty is in extracting good representative features.

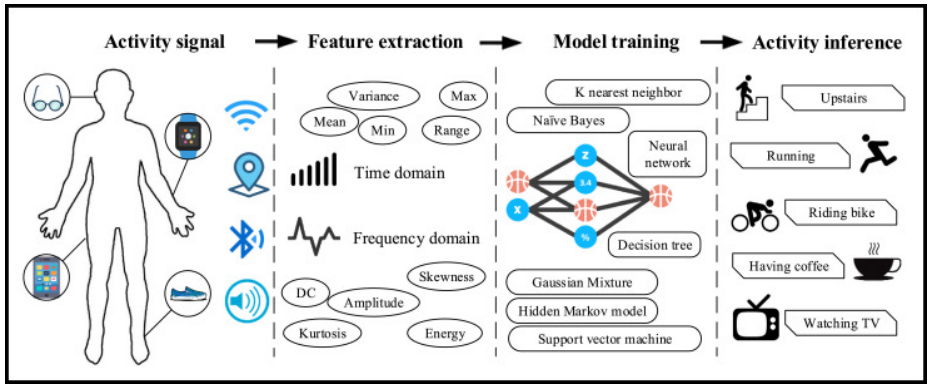

Figure 2-7: Traditional Approach for Sensor-Based Activity Recognition.

Recent deep learning approaches (see Figure 2-8) such as *convolutional neural networks*, CNNs, and *recurrent neural networks*, RNNs, have been shown to provide state-of-the-art results on HAR tasks using both raw data and features data [59].

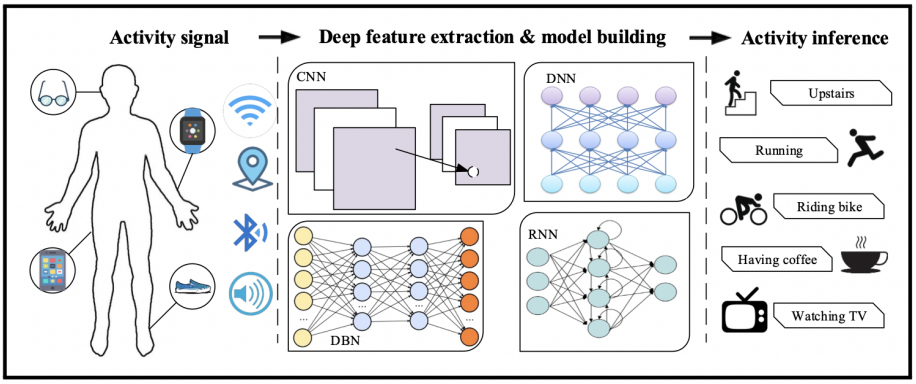

Figure 2-8: Deep Learning for Sensor-Based Activity Recognition.

Convolutional neural network models were designed for image classification problems, where the model learns an internal representation of a two-dimensional input. This process can be extended on time-series data, such as acceleration and gyroscopic data for HAR. The model extracts neural features from observations or handcrafted

features and maps them to different activity types. The one-dimensional data sequences are considered two-dimensional data in the form 1xn, where n is the sequence length. The three axes, x,y and z, are considered like RGB channels.

The benefit of using CNNs for sequence classification is that they can learn from the raw time series data directly and do not require domain expertise to extract input features. Moreover, CNN has one important benefit over other models: *local dependency*. Local dependency means the nearby signals are likely to be correlated. The model can learn an internal representation of the time series data and achieve optimal performances.

The following two neural networks, employed in the experimentations, are described: *ResNet* and *S-CNN*.

Both CNNs have been optimized for all the datasets through the Stochastic Gradient Descent with Momentum (SGDM), using a piecewise learning update strategy with an initial value of 0.1 and a drop factor of 0.1. The batch size was 128, the total number of epochs was 80, and the early stopping has been used to avoid overfitting. The Matlab Deep Learning Toolbox has been used for training and testing.

2.4.1 Residual Network (ResNet)

The *Residual Network (ResNet)* adopted is a 1-D CNN based on the traditional architecture proposed by He et al. [33], which demonstrated to be very effective on the ILSVRC 2015 (ImageNet Large Scale Visual Recognition Challenge) validation set with a top 1- recognition accuracy of about 80%. Residual architectures are based on the idea that each layer of the network learns *residual functions* regarding the layer inputs instead of learning unreferenced functions.

An input $\mathbf{x}$ is subject to a sequence of operation *convolution-ReLU-convolution*, obtaining $\mathbf{F(x)}$, summed by $\mathbf{x}$. The output is $\mathbf{H(x)}=\mathbf{F(x)}+\mathbf{x}$. Adding the input x preserves the information in the deepest layers, contrasting the gradient degradation. Through the concatenation of several *residual blocks*, in Figure 2-9, ResNet learns to predict a specific output by learning the term $\mathbf{F(x)}$ added to the input $\mathbf{x}$ by

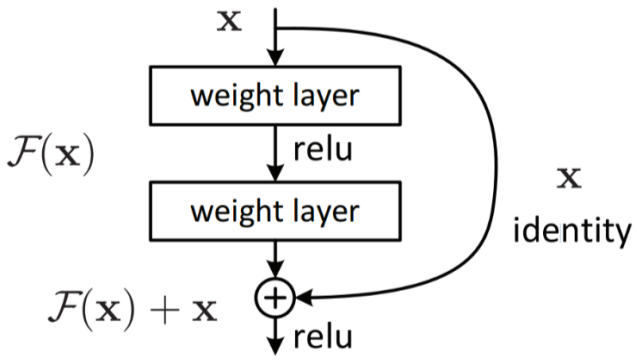

Figure 2-9: Architecture of Residual Block.

minimizing the error, called *residual error*. This approach represents *deep residual learning* and permits building deep networks.

Moreover, He et al. [33] demonstrate that such architectures are easier to optimize, and they gain accuracy also when the depth increases considerably. The input size of the network is the segment size ($1 \times 150 \times 4$ for Anguita and Shoaib and $1 \times 120 \times 4$ for Siirtola), which corresponds to 4 segments along the three axes x, y, z, and the magnitude. The network architecture, in Figure 2-10, is made of an initial convolutional block, 3 residual stages, each containing a variable number n of residual blocks, average pooling layer, fully connected layer, and softmax layer. A convolutional block is made of three layers: convolutional, batch normalization, and ReLU (Rectified Linear Unit) [5]. A residual block is made of 2 subsequent convolutional blocks and an addition operator that sums the input of the residual block with the output of the residual block itself.

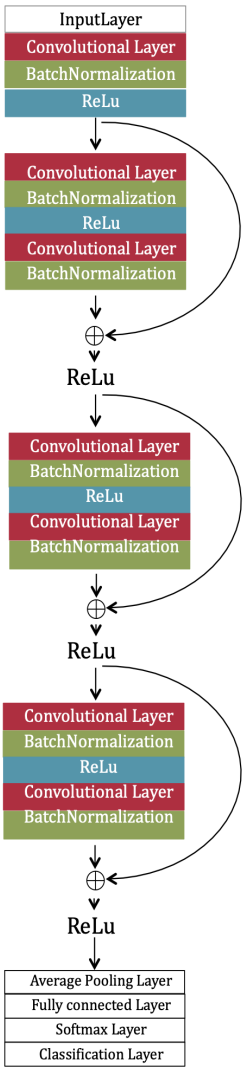

Figure 2-10: Residual Neural Network architecture.

2.4.2 Simplified Convolutional Neural Network (S-CNN)

Since ResNet is computationally and data demanding [6] a simpler architecture (S-CNN) has been designed. Krizhevsky demonstrates, in [60], that even networks with low layers can obtain excellent results and a series of advantages linked to their size. Thanks to their small size, training on cheap GPUs is possible, even relatively quickly and with the testing phase quite fast. A reduction in the parameters used corresponds to a reduction in the total training/training time.

Having a smaller size, CNNs are more energy-efficient, making them an excellent choice for mobile device applications.

The proposed architecture is shown in Table 2.5. The S-CNN is faster than the ResNet (Subsection 2.4.1) in both forward time and training time [60]. The input size of the network is the segment size ($1 \times 150 \times 4$ for Anguita and Shoaib and $1 \times 120 \times 4$ for Siirtola), which corresponds to 4 segments along the three axes x, y, z, and the magnitude. The network architecture is made of an initial convolutional block, 3 convolutional blocks, average pooling layer, fully connected layer, and softmax layer. A convolutional block is made of 2 subsequent convolutional and batch normalization layers and ReLU (Rectified Linear Unit).

Table 2.5: Architecture of S-CNN.

	Layers
1	Conv 16 filters of 1x5
2	Batch normalization
3	Conv 16 filters of 1x5
4	Batch normalization
5	Conv 16 filters of 1x5
6	ReLU
7	Conv 32 filters of 1x5
8	Batch normalization
9	Conv 32 filters of 1x5
10	ReLU
11	Conv 64 filters of 1x5
12	Batch normalization
13	Conv 64 filters of 1x5
14	ReLU
15	Fully connected
16	Softmax

2.5 Incremental Learning Procedure

The procedure consists in three phases: *(1) data preparation*, *(2) model generation*, and *(3) personalization*.

Each dataset contains data acquired by N subject, with N varying according to the dataset as reported in Tables 3.1 and 2.3. For each subject x, the following procedure has been iterated. All the subjects, except subject x, are used to train the base model in the *model generation* phase. The subject x is used in the *personalization* phase to adapt the model trained in the previous phase.

All the samples of user x are split into three equal partitions: D_{x1}, D_{x2}, and D_{x3}. D_{x1} and D_{x2} are used to personalize the model, whereas D_{x3} is used for testing. In the case of Learn++, all the samples, except those of the user x, are used for training the initial model. D_{x1} and D_{x2} are used separately to adapt the initial model for the user x. In the case of CNNs, all the samples, except those of the user x, are divided into two portions, 80% and 20%, which are used, respectively, for training and evaluating the initial model. For CNNs, D_{x1} and D_{x2} are merged to perform the adaptation for the user x.

The *model generation* phase consists of obtaining an initial model from all the samples of the dataset except those from the user x. This model is called *user-independent model*. The *user-independent model* is then tested on the D_{x3} in order to assess the performance on user x without personalization.

For the personalization phase three different approaches (see Figure 2-11) have been experimented: *non-supervised approach*, *semi-supervised*, and *supervised*. These approaches differ in the management of data labels of the two sets D_{x1} and D_{x2} used for the personalization process. In the non-supervised approach, data labels of D_{x1} and D_{x2} are not available, so the user-independent model is used to predict them. Once the labels are available, they are used to adapt the classifier, which is in our experiments the Learn++ and the CNNs. The semi-supervised approach considers a threshold (th) that allows to select predicted labels by the user-independent model. In particular, the labels with a higher score are selected, so those that are more likely well

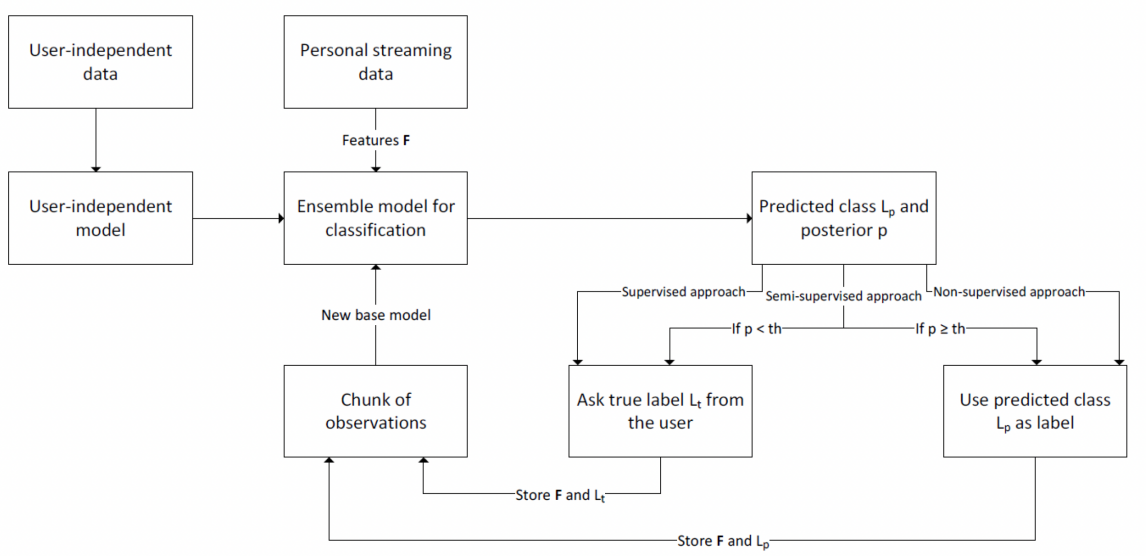

Figure 2-11: Procedure to personalize recognition model.

predicted by the classifier. If the score of a given label is higher than the threshold, the label is taken, otherwise, the correct label is taken from the ground-truth. This procedure is called semi-supervised because the ground-truth is provided by the users. The threshold adopted here is 0.90, which means that the labels predicted by the initial model are taken only if their score is higher than 90%. The supervised approach uses the samples from D_{x1} and D_{x2} with labels taken directly from the ground-truth.

In Learn++, the personalization process of the user-independent model is the following: the samples in D_{x1} are used to train a new model and then samples in D_{x2} are used to train another model. Both models are then added to the user-independent model, thus increasing the number of classifiers. Different training has been done on D_{x1} and D_{x2} data to avoid over-fitting and to make the final model more robust.

For what concerns the neural networks, the personalization process has been obtained by applying a transfer learning technique that allows preserving the knowledge acquired during the training phase of the user-independent model. In particular, a light fine-tuning approach has been used. It consists of freezing all the layers up to the classification layer (excluded). The light fine-tuning permits to quickly adapt the network to a new user, thus making this approach feasible in real applications.

2.5.1 Experiment Results

Table 2.6 shows the average of the experimental results obtained by all models with and without data augmentation over the three datasets. Performance is measured in terms of macro average accuracy (i.e., the average of each class accuracy). The accuracy of each class is computed as ratio between the number of segments correctly classified and the total number of segments of that class. Each number in the table shows the average over each user x of the macro average accuracy measured on the test set D_{x3}. Numbers clearly show that the employment of data augmentation increases the performance whatever is the method employed for incremental learning.

Table 2.6: Average of the macro average accuracy of all users whatever is the method adopted with and without the data augmentation procedure proposed. Best results are reported in bold.

Augmentation	User-Independent	Non-Supervised	Semi-Supervised	Supervised
no	89.10 (7.56)	89.41 (7.76)	94.78 (5.95)	98.01 (1.98)
yes	**90.32 (6.24)**	**90.73 (6.24)**	**96.98 (3.46)**	**99.53 (0.88)**

Table 2.7 shows the comparison between Learn++ and CNNs. The bold font represents the best method for each model (user-independent, non-supervised, semi-supervised, and supervised). Both CNNs overcome Learn++ whatever is the dataset and the model employed. ResNet and S-CNN achieve similar performance, thus confirming that even a less complex architecture can achieve good performance. Both CNNs, in the case of the user-independent model, improve with respect to Learn++ of about 12%. This permits to start from a more robust model that can be adapted to a new user more quickly (Fig. 2-12). In the Siirtola dataset, both CNNs achieve lower results than Learn++ in the case of semi-supervised. It is mainly due to the number of samples which is too low and not enough to train both networks.

Table 2.7: Macro average accuracy of all users: Learn++ vs ResNet and S-CNN. For each model (user independent, non supervised, semi-supervised and supervised), the bold font represents the best method.

Dataset / Method	User-Independent	Non-Supervised	Semi-Supervised	Supervised
Anguita / Learn++	85.60 (7.74)	86.72 (7.81)	96.89 (4.52)	99.53 (0.61)
Shoaib / Learn++	86.79 (4.10)	87.95 (4.06)	97.71 (1.91)	99.02 (0.48)
Siirtola / Learn++	75.84 (17.51)	76.24 (17.53)	95.59 (6.08)	97.66 (5.73)
Mean	82.74 (9.78)	83.64 (9.80)	96.73 (4.17)	98.74 (2.27)
Anguita / ResNet	97.30 (2.53)	97.33 (2.67)	98.78 (2.13)	99.81 (0.53)
Shoaib / ResNet	98.45 (0.76)	98.52 (0.76)	99.07 (0.56)	99.96 (0.05)
Siirtola / ResNet	86.67 (9.41)	87.31 (9.25)	92.72 (7.31)	99.94 (0.04)
Mean	**94.14 (4.23)**	**94.39 (4.23)**	96.86 (3.33)	99.90 (0.21)
Anguita / S-CNN	97.49 (2.75)	97.61 (2.79)	99.06 (1.87)	99.99 (0.27)
Shoaib / S-CNN	98.32 (0.78)	98.37 (0.83)	99.22 (0.64)	99.93 (0.05)
Siirtola / S-CNN	86.54 (10.61)	86.50 (10.48)	93.74 (6.11)	99.90 (0.12)
Mean	94.09 (4.71)	94.16 (4.70)	**97.34 (2.87)**	**99.94 (0.15)**

Figure 2-12 shows how fast is the process to adapt the user-independent model to a new user x for Learn++ and S-CNN. In particular, there is an increment of accuracy concerning the number of labels modified by the user. Except for the Siirtola dataset, S-CNN adapts much faster than Learn++, which requires, as in the case of the Shoaib dataset, more than two times the labels required by the S-CNN.

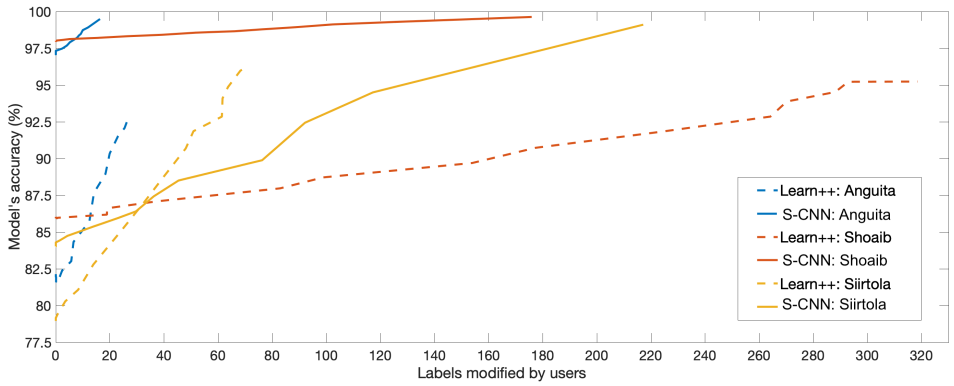

Figure 2-12: Learn++ Algorithm and S-CNN comparison in Semi-Supervised cases. The plot shows the trend of classifiers' accuracy in relation to the number of labels adjusted by a user.

Chapter 3

Homogenization of Existing Inertial-Based Datasets

The stakeholder approach to business sees integration rather than separation, and sees how things fit together.

John Mackey

In recent years, Deep Learning techniques have been successfully applied for 1D signals, exploiting their capability to overcome most of the presented issues, thanks to their properties of local dependency and scale invariance [59]. While Deep Learning methods are powerful and achieve high performance, they rely on very complex models that depend on estimating a large number of parameters, which in turn requires a considerable amount of available data [6, 61], whose quality increases the performance of the classification process. The quality of the dataset used to generate the recognition model can determine the effectiveness of the generated model. Building an effective dataset is a complex task. Several factors undermine its goodness. These include the naturalness with which users perform the tasks, the device's position, the balance of subjects involved, the number of samples recorded, and so on. Even if the dataset design is done rigorously, some factors, unfortunately, are not controllable: the intraclass variability and the interclass similarity. The former means

that different people perform the same activity in different ways, so a bijective association between signal and activity performed does not exist; the latter means that fundamentally different classes show very similar characteristics in terms of sensor data [15, 9, 16, 17].

The Continuous Learning Platform (CLP) aims to make available (i) a large amount of labelled inertial signals related to ADLs and falls; (ii) a catalogue of downloadable activity recognition models, and (iii) a service that, given a set of raw data, identifies the corresponding ADL. Inside the platform, a homogenization procedure allows integrating heterogeneous datasets to obtain a larger dataset for the definition of recognition techniques. Thanks to CLP, it is possible to train an efficient user-independent model thanks to a large amount of data coming from different users and contexts.

The final aim of this work is to define and implement a platform that integrates datasets of inertial signals to make large datasets of homogeneous signals available to the scientific community, enriched, when possible, with context information (e.g., characteristics of the subjects and device position). Thanks to this, it is possible to train an *efficient Deep Learning subject-independent model* in the personalization procedure discussed in Chapter 2.

The main contribution of this thesis is the definition of a *homogenization procedure* that allows integrating existing and heterogeneous datasets to obtain a larger dataset to be used for the definition of recognition techniques, emphasizing data quality, which is essential for efficient training models. The procedure has been implemented and integrated into the Continuous Learning Platform. Moreover, an analysis on the heterogeneity problem of existing datasets is provided, exploiting the differences such as the different units of measurements, the different contexts of data recording. Finally, an evaluation procedure validates the effectiveness of the CLP platform. The experiments showed that the homogenization of existing datasets is beneficial in training efficient Deep Learning models.

The Chapter is organized as follow. First, a review of the state of the art of literature is provided. It follows a description of datasets and their heterogeneity problem. Then, the homogenization procedure and the CLP architecture are explained. Finally, the evaluation procedure adopted is discussed, followed by an analysis of the results experiments.

3.1 State of the Art

The availability of a dataset containing a large number of samples, also obtained from the integration of existing datasets, is a well-known issue both in the field of *ADLs recognition* from inertial sensors and in other domains, such as that related to image processing [53]. Furthermore, Deep Learning methods need massive training sets of labelled data to learn from to reach good predictive performances. Such datasets are expensive to create, especially when quality and domain expertise is required. Some big companies hired large teams to label data in the wild, such as CTD-Pfizer [62] and Google [63]. However, often, those datasets are not public.
Since acquiring labelled data is a costly procedure in terms of resource, time, and people involved, *integrating* existing datasets may be a solution despite the substantial heterogeneity of the data.

In the context of ADLs recognition from inertial signals, Bartlett et al. propose labels aggregation at the semantic level [64]. Labels of six existent datasets, resampled to 200 Hz using linear interpolation, were relabelled manually to reflect their semantic similarity, obtaining 13 different activity labels. However, the proposal does not seem to consider details such as units of measurement or the presence of gravity in accelerations. Furthermore, the proposal allows only one configuration at 200 Hz.

Recently, Siirtola et al. proposed a Matlab tool called Open HAR [57] that aggregates labels at a syntactic level but fails to consider the semantics of the original signals. OpenHAR provides all the datasets in the same format, and an inspection operation was made to find datasets errors, such as sensors in the wrong orientation. So, the framework improves the re-usability of datasets by fixing the errors.

However, datasets with different sampling rates are unified using downsampling to 20Hz. Similarly to Bartlett et al. work, the OpenHAR proposal allows only one configuration.

Obinikpo et al. proposed a system for big data-d-health integration [65], conducting a study of publicly available datasets acquired from heterogeneous sources (i.e. wearable and smart devices). They split the integration into different layers: data acquisition, data processing, analytics, and application. Nevertheless, the proposal is too general concerning the homogenization of different data sources since its major concern is handling missing values while integrating databases.

However, Data Integration is one of the major bottlenecks in Machine Learning and Deep Learning and an active research topic in multiple communities. In the ADLs recognition context, the research is very active and offers many opportunities to researchers.

3.2 Used Datasets and Heterogeneity Problem

This section presents the datasets used in the experimentations and emphasizes their heterogeneity.

3.2.1 Datasets

The datasets used in the experimentation have been selected according to the following criteria: *i)* data should be acquired by smartphones and/or smartwatches in expectation of deploying the ADL recognition model directly to the end user's mobile device; *ii)* datasets have been acquired for HAR purposes, hopefully sharing the same types of activities; *iii)* datasets enriched with additional information related to the subjects' characteristics, such as sex, age, height, weight; and *iv)* datasets acquired with different devices for maximum heterogeneity. The datasets are briefly described below.

Human Activities and Postural Transitions (HAPT) [54]

HAPT contains 3-axial linear acceleration, 3-axial angular velocity, and gyroscope sensor data of 11 ADLs. A Samsung Galaxy S2 acquired data of 30 participants (19-48 years) who placed the smartphone in their belts. Subjects generated around 5 hours of experimental data. HAPT is an updated version of UCI HAR [55].

MobiAct Dataset (MobiAct) [66]

MobiAct includes tri-axial acceleration, gyroscope, and orientation sensor data of 9 ADLs and 4 Falls recorded with a Samsung Galaxy S3 and performed by 57 participants (20-47 years). The smartphone is located with random orientation in a loose pocket chosen by the subject.

Sensors Activity Recognition (SAD) [23]

SAD contains 3-axial acceleration, gyroscope, magnetometer, and linear acceleration sensor data of 7 ADLs. The data were recorded with a Samsung Galaxy S2 smartphone and performed by 10 subjects (male, 25-30 years) who completed each activity for 3-4 minutes. They used five smartphones on five body positions. : one in their right jean's pocket, one in their left jean's pocket, one on belt position towards the right leg using a belt clipper, one on the right upper arm and one on the right wrist.

UMAFall Dataset (UMAFall) [67]

UMAFall includes 3-axial linear acceleration sensor data of 12 ADLs and 3 Falls recorded with a Samsung Galaxy S5 smartphone and an LG G4 smartphone and performed by 19 participants (18-68 years). Subjects repeated a 15-second sequence of movements 18 times. The smartphone has been placed in their right pocket.

Physical Activity Recognition Dataset Using Smartphone Sensors (PARDUSS) [68]

PARDUSS contains 3-axial acceleration, gyroscope, magnetometer, and linear acceleration sensor data of 6 ADLs recorded with Samsung Galaxy SII smartphones and performed by 4 participants (male, 25-30 years). Each activity lasts between 3 and 5 minutes. Four smartphones placed in four different body positions acquired data.

Accelerometer Data Dataset (DA) [69]

DA includes 3-axial linear acceleration sensor data of 5 ADLs and 3 Falls recorded with a Nokia N8 smartphone and performed by 8 participants (25-34 years). The subjects placed the smartphone in their trousers' front pocket.

University of Milano - Bicocca SHAR (UniMiB SHAR) [70]

UniMiB SHAR contains tri-axial acceleration data organized in 3 seconds windows around the peak. Signals of 17 different activities (9 ADLs and 8 Falls) are collected and performed by 30 subjects (18-60 years). The subjects placed the smartphone used for the acquisition (a Samsung Galaxy Nexus I9250) half of the time in the left trouser's pocket and the remaining times in the right one. They repeat each activity several times.

MotionSense Dataset (MS) [71]

MS contains time-series data generated by the accelerometer and the gyroscope sensors of an iPhone 6s worn by 24 participants (18-46 years). Each of the subjects performed 6 ADLs in 15 trials, 9 long trials of around 2 to 3 minutes duration and 6 short trials of around 30 seconds to 1-minute duration, in the same environment and conditions. The smartphone was kept in the participant's front pocket.

3.2.2 Datasets' Comparison

Table 3.1 shows how the datasets differ in terms of frequency and unit of measurement. Moreover, the table shows the smartphones used to acquire the data.

Table 3.1: Datasets information.

Dataset	Sensor	Unit of measure	Frequency	Smartphone
HAPT	accelerometer	g	50 Hz	Samsung Galaxy S2
	gyroscope	rad/s	50 Hz	
MobiAct	accelerometer	m/s^2	87 Hz	Samsung Galaxy S2
	gyroscope	rad/s	87 Hz	
	orientation	$degrees$	87 Hz	
SAD	accelerometer	m/s^2	50 Hz	Samsung Galaxy S2
	gyroscope	rad/s	50 Hz	
	magnetometer	$microTesla$	50 Hz	
UMA	accelerometer	m/s^2	200 Hz	Samsung Galaxy S5, LG G4
PARDUSS	accelerometer	m/s^2	50 Hz	Samsung Galaxy S2
	gyroscope	rad/s	50 Hz	
	magnetometer	$microTesla$	50 Hz	
DA	accelerometer	m/s^2	40 Hz	Nokia N8
UniMiB SHAR	accelerometer	m/s^2	50 Hz	Samsung Galaxy Nexus I9250
MS	accelerometer	g	50 Hz	Iphone 6s
	gyroscope	rad/s	50 Hz	

Figure 3-1 reports an example of a walking activity performed in different datasets. In particular, it is possible to note that units of measurement (UOMs) are different and that the same activity is performed differently.

In order to confirm the necessity of a homogenization procedure, the following Activities of Daily Living have been selected: *jogging, sitting, standing, stairs down, stairs up, walking*. The activities selection is due to several reasons. *i*) inclusion in most datasets; *ii*) most common in HAR; *iii*) distinction from each other.

However, the *iii*) reason is not valid for *sitting* and *standing* activities. After, a data analysis confirmed a high similarity (see Figure 3-2 and Figure 3-3) in their

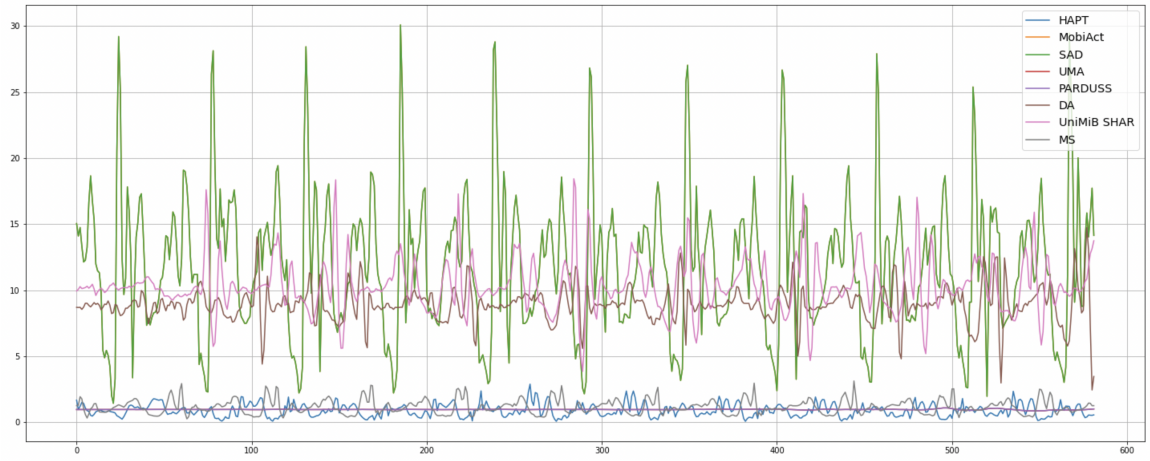

Figure 3-1: Walking activity (magnitude) performed by all original datasets. For example, DA and UniMiB SHAR signals are around the value 9.8, the gravitational acceleration constant present in the UOM of m/s^2. Instead, HAPT and MS are around 0 due to their UOM of g.

signals due to the data acquisition phase. It may be possible that the activities have been carried out similarly but with an opposite sequence. For example, the *sitting* activity starts standing and ends sitting, while the *standing* activity starts sitting and ends standing. For all those reasons, and after some experiments, the *sitting* and *standing* activities are merged into sitting/standing. Moreover, in DA Dataset [69] the *sitting* and *standing* activities are merged into one (*idling*).

Then, the final activities chosen to validate the homogenization procedure are *jogging, sitting/standing, walking downstairs, walking upstairs, walking*.

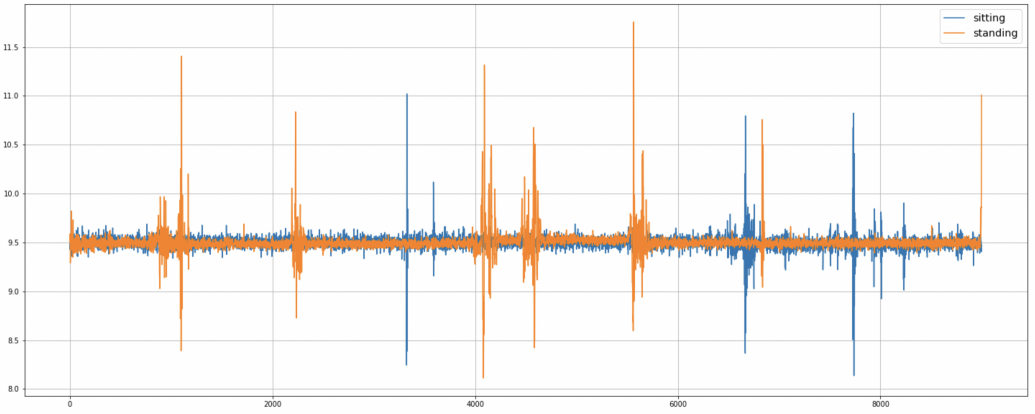

Figure 3-2: Sitting (blue) and Standing (orange) activities (acceleration magnitude) comparison for SAD Dataset.

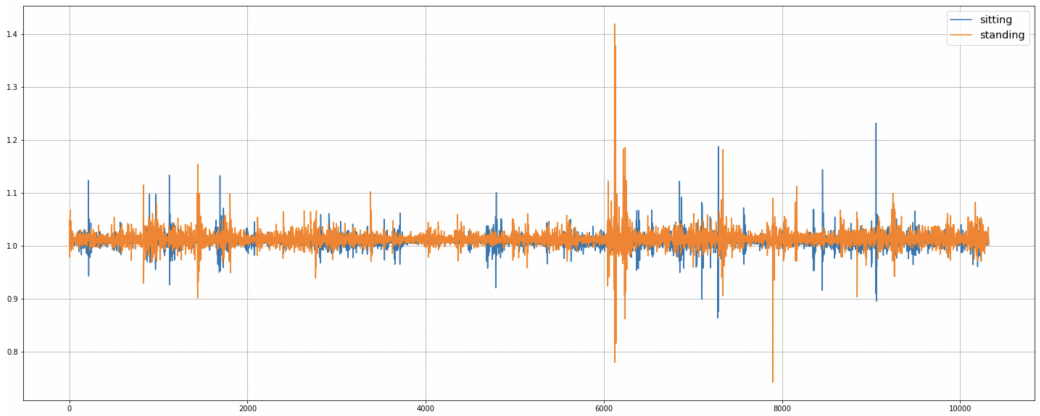

Figure 3-3: Sitting (blue) and Standing (orange) activities (acceleration magnitude) comparison for MS Dataset.

Table 3.2 reports the presence and the corresponding ADL of the five selected activities. It confirms that there is no common standard for labelling inertial signals in all datasets. Therefore, a label homogenization procedure is needed.

Table 3.2: Datasets' labels. ✓indicates that the same ADL name is used, while X that the activity is not present.

	jogging	**sitting/standing**	**walking downstairs**	**walking upstairs**	**walking**
HAPT	X	sitting, standing	✓	✓	✓
MobiAct	✓	standing	stairs down	stairs up	✓
SAD	✓	sitting, standing	✓	✓	✓
UMA	✓	standing down and getting up from a chair	go downstairs	go upstairs	✓
PARDUSS	Running	Sitting, Standing	✓	✓	Walking
DA	running	idling	X	X	✓
UniMiB-SHAR	Running	Standing Up FS, Standing Up FL, Sitting Down	Going Down	Going Up	Walking
MS	jog	sit, std	dws	ups	wlk

3.3 Homogenization Procedure

The homogenization of heterogeneous datasets contributes to creating a unified dataset containing many samples with the same characteristics (i.e. sampling rate, unity of measurement and standard labels) that can significantly improve the quality of observed time series data regarding Activities of Daily Livings.

In this work, the proposed homogenization procedure involves two different phases: *signals homogenization* and *labels homogenization*.

3.3.1 Signals Homogenization

As the name suggests, the signals homogenization procedure focuses on the signals and handles three types of inconsistencies: differences in sampling frequencies, discrepancies in units of measurement, and the presence of gravity for acceleration signals. The process involves three steps:

- *Frequency uniformation.* A resampling operation takes place intending to modify the frequency of the time series. There are two different types of resampling: upsampling, when the sampling rate is increased compared to the original one, and downsampling, when the sampling rate is decreased compared to the original one. The number of samples obtained with the new frequency can be calculated starting from Equation 3.1.

$$numSamples = \frac{numOriginalSamples * newFreq}{originalFreq} \tag{3.1}$$

  Once the new time series with the desired frequency has been obtained, the respective timestamp is calculated for each sample starting from 0. The user provides information regarding the sampling rate in the dataset being imported. If this information is not available, it is calculated using the timestamp provided in addition to the inertial data. When the sampling frequency has been obtained or estimated, the data can be resampled. For all sensors, a frequency of 50Hz has been chosen.

Literature suggests that about 50Hz is a suitable sampling rate that permits to model human activities [72].

- *Unit of measurement uniformation.* The unit of measurement, unlike the sampling frequency, must always be provided. It is not possible to trace this information from inertial data. The unit of measurement of a given sensor is converted to the desired one by a specific formula. For example, to convert an accelerometer from g to m/s^2, it is necessary to add the gravitational acceleration, that is, multiplying by the gravitational acceleration constant, equal to $9.80665 m/s^2$. The following uoms have been adopted: for accelerometer, gyroscope, and orientation m/s^2, rad/s, and $microTesla$ respectively.

- *Gravity uniformation.* To remove gravity and reduce noise or artefacts, a Butterworth filter is commonly applied [73, 74]. A fifth-order 0.5 Hz low pass Butterworth filter with a Nyquist frequency of 25Hz has been considered, assuming the gravity force to have only low-frequency components. Since the information about gravity being included in the signal is often omitted, the Butterworth filter has been applied to all raw inertial signals.

Figure 3-4 shows the result of Signal Homogenization for the same activity in distinct datasets.

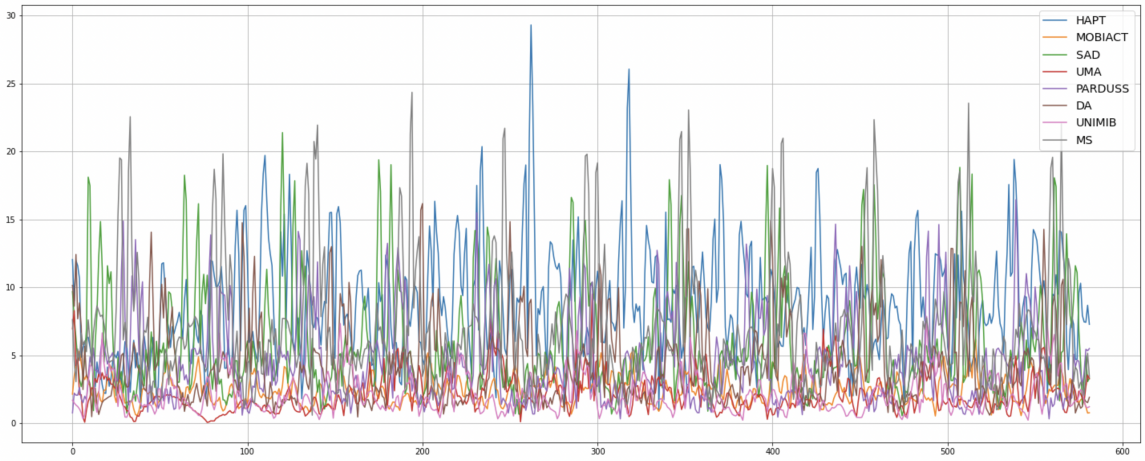

Figure 3-4: Application of Signal Homogenization procedure on Walking activity, the same performed on heterogeneous datasets in Figure 3-1

3.3.2 Label Homogenization

The label homogenization procedure aligns the labels of the ADLs in the dataset being imported with the ones already present in the homogenized dataset. Because there is no shared definition for each ADLs and each dataset can have different and conflicting labels, relying on the syntactic similarity may not be sufficient (e.g., walk and walking describe the same activity although the two labels are syntactically different, on the other hand, sitting, and sitting down, may be syntactically similar, but they refer to different actions: being sit and actively sitting from a standing position).

Three different approaches are proposed to make an accurate mapping:

- *Label Syntax Similarity (LSS).* This index indicates the syntactic similarity of two strings (labels) without considering the signal component. The Levenshtein distance has been used, which is the minimum number of elementary modifications (deleting a character, replacing one character with another, or inserting a character) that allows transforming a string A into another string B. For example, the Levenshtein distance between walk and walking equals 3. This distance can be helpful in some cases, but it cannot provide adequate information to choose the correct mapping automatically;

- *Label Signal Similarity Distance (LSSD).* Considering the signal component from the time series, the 21 hand-crafted features have been extracted from the magnitude component of the signal for each window. These features describe the different properties of the signal both in the time and frequency domain. The average of the features extracted from the windows of a specific activity represents the entire ADL. Each dataset has a feature vector per ADL. The Euclidean distance is then applied to determine possible associations. If the distance of two activities is low, it suggests the associated signals are very similar, and the suggestion is a mapping between the two ADLs;

- *Magnitude comparison graph.* The procedure also includes a manual comparison of the magnitude of different labels' time series for each pair that satisfies the

minimum equality criteria. A random time series is taken for each label to create the graph. This is a visual aid to help a user in deciding which mappings to perform.

The final decision about mappings when importing a dataset is always left to the users.

Table 3.3: Example of Label Syntax Similarity (LSS).

	jogging	**sitting/standing**	**walking downstairs**	**walking upstairs**	**walking**
HAPT	X	- -	-	-	-
MobiAct	-	8	12	12	0
SAD	0	9 8	0	0	0
UMA	0	27	7	7	0
PARDUSS	4	9 8	0	0	0
DA	4	12	X	X	0
UniMiB-SHAR	4	12 12 9	11	11	1
MS	4	13 13	15	13	4

Table 3.4: Example of Label Signal Similarity Distance (LSSD).

	jogging	**sitting/standing**	**walking downstairs**	**walking upstairs**	**walking**
HAPT	X	- -	-	-	-
MobiAct	-	0.085	0.0616	0.0894	0.0936
SAD	0.0535	0.0653 0.0655	0.198	0.1841	0.0644
UMA	0.0877	0.0836	0.1662	0.1451	0.0708
PARDUSS	0.058	0.072 0.072	0.1936	0.1805	0.0734
DA	0.0402	0.0624	X	X	0.0661
UniMiB-SHAR	0.0157	0.0709 0.0609 0.0719	0.1936	0.1805	0.0245
MS	0.0267	0.0548 0.0492	0.14	0.1343	0.0511

However, LSS and the LSSD are useful Decision Support Systems that ease the process and reduce the confusion that may arise from heterogeneous labels.

Table 3.3 and Table 3.4 shows LSS and LSSD criteria, according to datasets and activities summarized in Table 3.2.

LSS Table confirms the heterogeneity problem of labels, but also that it can also not provide adequate information to choose the correct mapping automatically.

Instead, LSSD Table is more meaningful. The values lie in the range [0, 1], with 0 being similar signals and 1 not alike. Observing the values, it is possible to assume that 'walking downstairs' and 'walking upstairs' activities are less similar. Probably, due to the signal acquisition modality (i.e. inside/outside), the physical characteristics of users and the individual activity performances (i.e. a more trained person theoretically has differents moves and speed than a less trained one).

3.4 Continuous Learning Platform Architecture

Continuous Learning Platform (CLP), in Figure 3-5, implements the homogenization procedure described in Section 3.3 and provides the tools to both integrate new datasets and to retrieve homogenized datasets and recognition models.

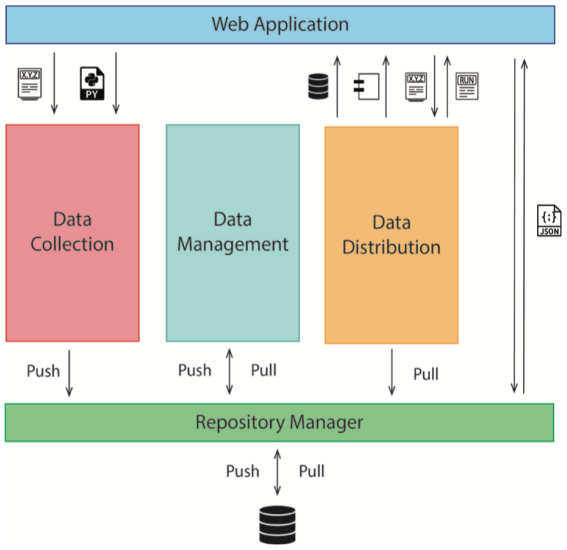

Figure 3-5: CLP architecture.

Three main components constitutes CLP: *i) Data Collection*, that acquires a new dataset; *ii) Data Management* that homogenizes the new dataset and insert it in the incrementally built dataset; and *iii) Data Distribution*, which enables users and applications to query the platform and obtain homogeneous sets of labelled signals, but also ad-hoc trained classifiers.

Two more components complete the platform: the *Repository Manager* that deals with the management and internal storage of all the platform data, and a *Web Application* responsible for making all the services offered by the platform available through an intuitive graphical interface. The interaction is possible by starting the various services through HTTP calls, using REST APIs.

The individual components were developed in Python, while a message broker (RabbitMQ) has been used to permit efficient interaction between the components. The Web Application is built using the Angular framework, while REST APIs have been used for communication. The components are implemented as a Flask server to manage the REST calls coming from the Web Application. The platform uses MongoDB as a database.

The rest of the section will give a more detailed view of the components.

3.4.1 Data Collection

Data Collection, Figure 3-6, includes the following modules. The *Dataset Loader* allows users to physically upload datasets and store them in the local repository, adding them to all the other datasets in the platform. The *Driver Loader* handles the upload of datasets drivers, which are scripts provided by the users allowing the platform to correctly interpreter the data in the dataset. The *Importer* executes the actual import by running a specific driver loading the corresponding dataset, thus standardizing it according to a specific structure shared by all the datasets imported. It is noteworthy that, at this stage, the content of the datasets are still heterogeneous, while they share a common structure.

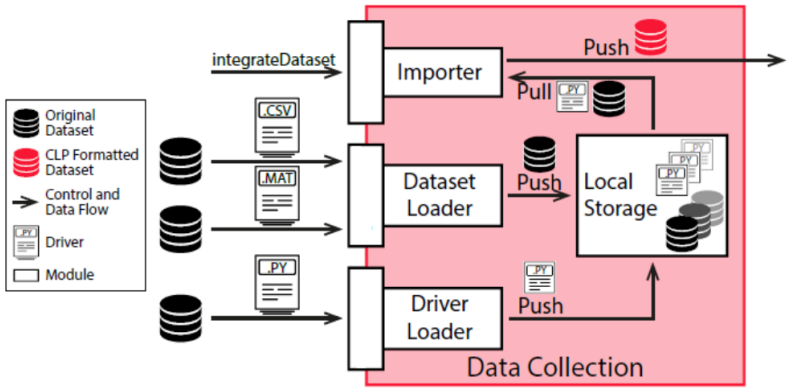

Figure 3-6: Data Collection component.

3.4.2 Data Management

Data Management, Figure 3-7, includes the following modules. The *Data aligner* standardizes each signal in a dataset at a specific frequency and unit of measurement, according to a common configuration. It also applies a Butterworth filter to remove the gravity from accelerations and to reduce noise and other artefacts. The *Feature extractor* computes the magnitude and a set of hand-crafted features [5] from inertial signals to determine signal similarity. Finally, the *Label comparator* uniforms the dataset's labels to include a standard unified set, also considering the signal similarities. This module is semi-automatic: it provides suggestions on the assignment of labels, but ultimately it is down to the end-user to decide whether or not to accept the suggestions.

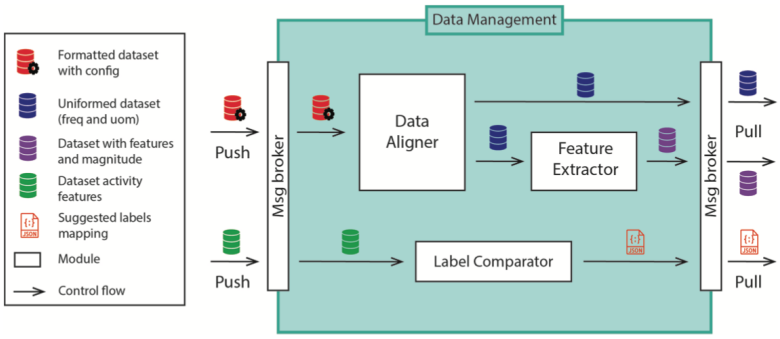

Figure 3-7: Data Management component.

3.4.3 Data Distribution

Data Distribution, Figure 3-8, consists of the following modules. The *Query builder* handles users' queries for homogenized data. The *Classifier builder* trains classifiers according to user queries. Thanks to the homogenization procedure, the models generalize well in terms of inter and intra-subject variability regardless of the fact that they are trained on data coming from different datasets. The *Classifier deployer* distributes according to users' requests; The *Online classifier* provides online services related to classification: given a set of inertial signals, it provides information regarding the subject's activity.

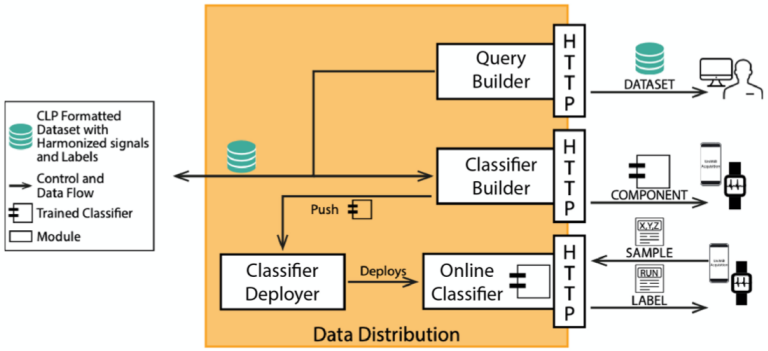

Figure 3-8: Data Distribution component.

3.4.4 Repository Manager

Repository Manager, Figure 3-9, interacts with all the components of the platform and includes the following modules. The *Importer manager* manages all messages from the Data Collection, all those relating to importing a new dataset; the *Config manager* adds a new configuration into the platform. For each, CLP starts a uniformation process according to its values; the *Uniformer manager* handles the communication between the Repository Manager and the Data Management component; the *Query manager* manages the Data Distribution requests to download data and classifiers. It forwards the requests to the database, which returns the requested data and saves them in a JSON file.

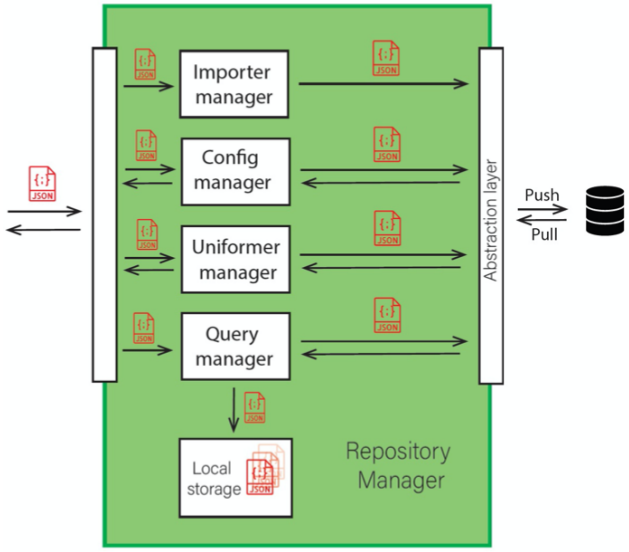

Figure 3-9: Repository Manager component.

Furthermore, the Repository Manager implements an Abstraction layer. It takes care of the separation between the logical part of the component and the database. All messages to and from it must necessarily pass through this layer, which contains all the functions necessary to execute the CRUD functions on the database. It aims to separate everything related to component logic with direct database interaction. With data being one of the most significant research fields of the new millennium, the CLP platform's need to keep pace with technology is an essential requirement. Thanks to this additional layer, it will be possible to replace the database used with a more performing one at any time, simply by replacing the essential interaction functions with it.

3.4.5 Web Application

The Web Application (example of use in Figure 3-10), is responsible for making all the services offered by the platform available through an intuitive graphical interface. The interaction is possible by starting the various services through HTTP calls, using REST APIs. The communication between the Web Application and all the components is synchronous.

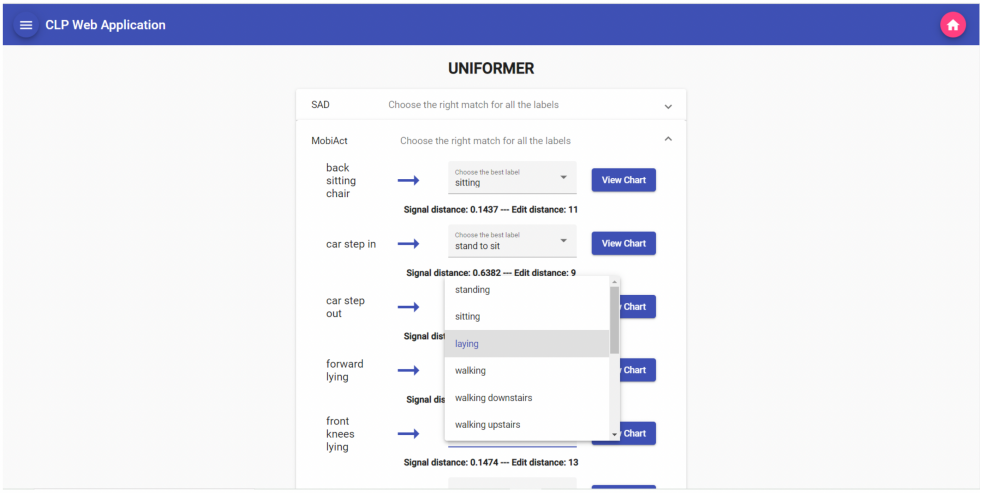

Figure 3-10: Example of Label Homogenization mapping in the Web Application.

The main components of the Continuous Learning Platform have been fully implemented and are available at `https://gitlab.com/Pervasive-Healthcare/CLP`.

3.5 Evaluation Procedure

This experimentation aims to verify whether the joint use of datasets (even if heterogeneous) can generate more effective and efficient human activity recognition models than those generated using a single dataset. To answer this question, the experimentation tries to answer the following *research questions*:

- **RQ1**. *Can a larger dataset built by integrating heterogeneous datasets be an effective base for building an ADL recognition model with a subject-independent approach?*

  Even if datasets have been proven to be different, the proposed homogenization strategy must be verified. Therefore, it is necessary to initially identify the most appropriate strategy for integrating heterogeneous datasets in order to be able to verify performance both in the case of unified datasets and in the case where datasets were merged.

- **RQ2**. *Is fine-tuning on a model generated by merging heterogeneous datasets effective in the case of a new user?* Fine-tuning reduces the required computation time and resources. For this reason, the generation of a recognition model using fine-tuning can also be performed on a mobile device. This research question aims to verify whether fine-tuning is more effective than a classical hybrid approach starting from merged heterogeneous datasets.

RQ1 Experimental Procedure

For each dataset presented in Section 3.2, the following procedure after applying the homogenization procedure is iterated.
All the datasets $D_{i=0,..,n}$, where n is the number of datasets, except D_x, are used to train the subject-independent (SI) model.
D_x is split into three partitions: D_x1 (70%), D_x2 (10% of D_x1) and D_x3 (30%). D_x1 and D_x2 are used to train the baseline model, whereas D_x3 tests it and the SI model. The baseline model allows us to have a comparison of how the homogenization procedure is doing well.
Each of the other datasets D_i is divided into 80% and 20%, respectively, train D_i1 and validation D_i2. One by one, all the datasets are added to the previous, except for D_x. Training and validation data are used to generate a subject-independent model during each iteration. Each iteration provides a different model which considers the new data and the previous added. The SI model is tested on D_x3 and compared to the baseline model.

The adding order of datasets is not essential because of the training from zero. Consequently, the final result is equivalent considering all possible orders.

The RQ1 experimental procedure is also performed on heterogenous datasets in order to have comparable results. In particular, D_x3 is used to test the obtained model after each iteration.

RQ2 Experimental Procedure

The RQ2 experimental procedure compares two different methods: *standard training* and *fine-tuning*. According to the RQ1 experimental procedure, all the datasets are used to train the subject-independent (SI) model, except for D_x.

The standard training approach consists of adding D_x1 and D_x2 to the training and validation data used to generate the SI model to train a new model that considers D_x data. Then, a *Transfer Learning* technique is used to preserve the knowledge acquired during the training phase of the subject-independent model. In particular, a light fine-tuning approach has been applied on D_x1 and D_x2 data. It consists of freezing all the layers up to the classification layer (excluded). The light fine-tuning permits quickly adapting the network to a new user, thus making this approach feasible in real applications.
Finally, both generated models are tested with D_x3, such as in the RQ1 experimental procedure, in order to have comparable results.

3.5.1 Methods

In the following, the methods and the Neural Networks adopted are explained.

Data pre-processing

Two different input data are proposed: *i*) the three axes x,y,z; *ii*) the magnitude extracted from the axes. The former is traditionally used in Human Activity Recognition. While, the latter allows removing information regarding the sensor orientation, giving more similar signals but reducing the input size. Moreover, only the accelerometer has been considered since it is the only sensor common to all datasets.

Moreover, the Data Augmentation method introduced in Section 2.3.1 is used, allowing Deep Learning models to be more robust by producing data variations that the model may recognize in the real world, where the domain continues to diversify and increase rapidly.

Neural Networks

Two neural networks were employed in the experimentation: the *Simplified Convolutional Neural Network (S-CNN)*, summarized in Section 2.4.2, and a *Recurrent Neural Network (RNN)*.

RNNs such as Long-Short Term Memory (LSTM) networks can learn long-term dependencies, making them adapted to temporal dynamics in activity time-series. Table 3.5 shows the overall architecture of the adopted LSTM network in the experiments. The input size is the same as the S-CNN. The network architecture consists of an initial LSTM layer with 100 hidden units, a fully connected layer, and a softmax layer.

Table 3.5: Architecture of adopted LSTM Network.

	Layers
1	LSTM layer with 100 hidden units
2	Fully connected
3	Softmax

Both networks were optimized for all the datasets through the *Adam optimizer*, using a piecewise learning update strategy with a drop factor of 0.1. The batch size was 128, the total number of epochs was 80, and the early stopping has been used to avoid overfitting. The Matlab Deep Learning Toolbox was used for training and testing the Simplified CNN and the LSTM network.

3.5.2 Experiment Results

Table 3.6 reports the baseline, RQ1 (without and with homogenization) and RQ2(standard training and fine-tuning) experimentation results. *HAPT* and *DA* entries are not shown since they do not have all 5 chosen activities. However, they are added to the uniformed dataset and contribute to adding variability. The aim is to validate the homogenization procedure discussed and the Continuous Learning Platform discussed in Sections 3.3 and 3.4. Since it is hard to summarise all experiment results for each approach (i.e. input data and neural network) are reported the average F1-score of

all 5 selected activities for all conducted experiments. According to the following formula, F1-score is used to have a compact metric presentation of precision and recall:

$$F1score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.2}$$

Table 3.6: Macro average F1-score of all experimentations to evaluate the Continuous Learning Platform.

Dataset	Method	Baseline	RQ1 (heterogeneous)	RQ1 (homogenization)	RQ2 (standard training)	RQ2 (finetuning)
MobiAct	3acc-SCNN	91.46	77.93 (-13.53)	85.67 (-5.79)	90.04 (-1.42)	92.36 (+0.90)
	3acc-LSTM	90.72	59.47 (-31.25)	69.91 (-20.81)	85.87 (-4.85)	83.92 (-6.80)
	magn-SCNN	92.59	71.56 (-21.03)	83.28 (-9.31)	91.75 (-0.84)	93.97 (+1.38)
	magn-LSTM	91.33	66.87 (-24.46)	80.10 (-11.23)	89.59 (-1.74)	90.74 (-0.59)
SAD	3acc-SCNN	97.39	78.27 (-19.12)	73.48 (-23.91)	93.98 (-3.41)	96.06 (-1.33)
	3acc-LSTM	94.37	70.00 (-24.37)	74.43 (-19.94)	89.07 (-5.30)	91.53 (-2.84)
	magn-SCNN	89.11	69.35 (-19.76)	85.96 (-3.15)	89.62 (+0.51)	87.86 (-1.25)
	magn-LSTM	82.89	80.12 (-2.77)	84.81 (+1.92)	87.76 (+4.87)	87.98 (+5.09)
UMA	3acc-SCNN	74.31	50.47 (-23.84)	58.85 (-15.46)	71.14 (-3.17)	75.29 (+0.98)
	3acc-LSTM	70.42	38.89 (-31.53)	62.92 (-7.50)	60.33 (-10.09)	70.07 (-0.35)
	magn-SCNN	68.55	56.96 (-11.59)	65.06 (-3.49)	67.50 (-1.05)	69.80 (+1.25)
	magn-LSTM	55.87	55.80 (-0.07)	67.21 (+11.34)	64.77 (+8.90)	66.18 (+10.31)
PARDUSS	3acc-SCNN	60.63	43.45 (-17.18)	74.24 (+13.61)	78.32 (+17.69)	73.48 (+12.85)
	3acc-LSTM	66.10	41.40 (-24.70)	68.47 (+2.37)	67.56 (+1.46)	67.98 (+1.88)
	magn-SCNN	74.82	43.69 (-31.13)	88.34 (+13.52)	90.83 (+16.01)	84.67 (+9.85)
	magn-LSTM	71.06	43.56 (-27.50)	79.52 (+8.46)	82.44 (+11.38)	79.59 (+8.53)
UniMiB SHAR	3acc-SCNN	91.57	50.55 (-41.02)	72.14 (-19.43)	85.69 (-5.88)	91.63 (+0.06)
	3acc-LSTM	81.28	39.56 (-41,72)	60.09 (-21.19)	77.99 (-3.29)	81.86 (+0.58)
	magn-SCNN	87.60	45.91 (-41.69)	67.52 (-20.08)	83.23 (-4.37)	88.96 (+1.36)
	magn-LSTM	76.44	42.93 (-33.51)	57.95 (-18.49)	75.56 (-0.88)	84.97 (+8.53)
MS	3acc-SCNN	95.05	49.46 (-45.59)	88.44 (-6.61)	97.76 (+2.71)	96.94 (+1.89)
	3acc-LSTM	91.85	44.37 (-47.48)	71.45 (-20.40)	92.33 (+0.48)	93.85 (+2.00)
	magn-SCNN	94.68	72.07 (-22.61)	88.44 (-6.24)	94.30 (-0.38)	94.31 (-0.37)
	magn-LSTM	90.98	78.32 (-12.66)	83.74 (-7.24)	91.39 (+0.41)	91.99 (+1.01)
Mean	3acc-SCNN	85.06	58.35 (-26.71)	75.47 (-9.59)	86.14 (+1.08)	87.61 (+2.55)
	3acc-LSTM	82.45	53.09 (-29.36)	67.88 (-14.57)	78.86 (-3.59)	81.96 (-0.49)
	magn-SCNN	84.55	59.92 (-24.63)	79.76 (-4.79)	86.19 (+1.64)	87.00 (+2.45)
	magn-LSTM	78.09	61.27 (-16.82)	75.55(-2.54)	81.91 (+3.82)	86.57 (+5.48)

Concerning *RQ1 experimentation results*, there is a mean decrement in performances of 24.38% for heterogeneous datasets regarding baseline. While applying the homogenization procedure, there is an increment of 16.51% and a difference of -7.87% about the baseline. The numbers demonstrate that uniforming data is beneficial. Moreover, the results are reasonably good considering that it is a *subject-independent approach.*

Instead, considering *RQ2 experimentation results*, there is a good increment of 8.61% in standard training regarding the SI approach, with the disadvantage of re-training all the networks, making it not particularly suitable in mobile device cases. Instead, in the fine-tuning experiment, there is a remarkable increment of 11.12% against the SI method. So, excellent performance can also be obtained by running it on mobile. Regarding the baseline results, there is a mean increment of 0.74% and 3.25%, respectively. Surprisingly, in magnitude and LSTM combination, the increment is 3.82% and 8.48%. The numbers confirm that an improvement could be obtained by adding a small amount of baseline (or a new user) data, confirming the discussion made about Figure 2-12.

In conclusion, the obtained results have demonstrated that: *i*) the increase of data leads to better performances and *ii*) even using datasets from different smartphones, it is possible to obtain good results. In particular, the *ii*) point is to thank the Butterworth filter (by removing gravity and noise) and the magnitude (by removing the device orientation).

Chapter 4

Deep Learning-based Inertial Sensory Clustering

Learning is not attained by chance, it must be sought for with ardor and attended to with diligence.

Abigail Adams

Clustering is a fundamental *Unsupervised Learning* task generally employed in exploratory data mining, image analysis, information retrieval, data compression, pattern recognition and text clustering. It aims to separate data into *clusters* based on similarity, density, intervals or particular statistical distribution measures of the data space. The performances of Clustering algorithms depend on the input data type. Different problems and datasets could require different similarity measures and distinct separation methods, including *Deep Clustering*. Applying Deep Learning neural networks is possible to learn non-linear mappings, transforming the data into more Clustering-friendly representations, where separation is easier for the problem's context [75, 76]. Clustering is beneficial for various tasks. For instance, it can be helpful for automatic data labelling for supervised learning and as a pre-processing step for data visualization and analysis. The former is the task of this labour.

Aim of this work is to experiment with the effectiveness of a DL-based Clustering architecture that combines a Recurrent AutoEncoder and a Clustering Criterion to predict unlabelled HAR signals, considering their effectiveness in solving the problem of data clustering, well-known in Unsupervised Learning. The proposed architecture will be integrated and used in the approaches explained in the previous Chapters. In particular:

- No continuous user interaction will be needed in the personalization of DL-based HAR models (see Chapter 2), which could harm his/her experience and lead to incorrect data annotation. Hence, the suggested approach will annotate unlabelled data silently without interacting with the user, giving a major emphasis on data since the method no longer relies on labels;
- It will be possible to integrate unlabelled datasets or users' recordings to feed the uniformed dataset provided by the Continuous Learning Platform (see Chapter 3). So, it will be possible to increase the inter and intra-variability of data by having the user consense to record his/her data.

This work has been structured as follow.
Firstly, the literature results obtained by Alireza et al. in [77] have been replicated. They defined a DL-based Clustering architecture that learns highly discriminative representations using self-supervision with reconstruction and future prediction tasks informed by feedback from a clustering objective to guide the network towards clustering-friendly representations.
Then, a different architecture was proposed, trying to increase the performances obtained by Alireza et al. Following some analyses and researches, the recurrent encoder has been modified with a Convolutional Gated Recurrent Unit (ConvGRU) [78] based encoder to encode a better deeply embedded representation. First, the convolution operations inside the ConvGRU extract the spatial features, and then the GRU, by its nature, exploit the temporal features. The recurrent decoders are the same because of their ability in forcing the model to look for information deep inside the encoder,

emphasizing extracted spatio-temporal features by the encoder. Finally, an analysis of data has been conducted. In particular, the magnitude extracted from tri-axial sensors has revealed how decreasing the number of sensors in input data is beneficial. However, it also reveals that a small number of channels in input data can deteriorate the performances. The experimentation results expose a comparison between the proposed architecture and some other approaches in the literature [75, 77].

The Chapter is organized as follow. First, a review of the state of the art of literature is provided. It follows an in-depth description of the proposed architecture after the problem statement. Finally, the experiments are discussed and the results analyzed.

4.1 State of the Art

During the last years, several DL-based Clustering approaches have been proposed in the literature [75]. Every method consists of a neural network training procedure for learning deep representative features and, generally, an ML-based Clustering algorithm for data separation [75, 76].
The working principles of a DL-based Clustering method consist of two different stages:

- *Stage 1*: representation learning and parameter initialization with a DNN and training using loss independent of the clustering algorithm (*non-clustering loss*). Then *low-dimensional cluster-friendly deep features* are extracted from the network architecture;

- *Stage 2*: parameter optimization by iterating among computing an auxiliary target distribution and minimizing *clustering loss* and *cluster assignment hardening* where cluster assignments are formulated. Subsequent, the centroid is updated with the backpropagation to optimize the clustering algorithm.

Examples of DNNs, used in Stage 1, are multilayer perceptrons (MLPs) [79], Convolutional Neural Networks (CNNs) [59] and AutoEncoders (AEs) [80, 81]. Instead, in Stage 2, commonly, ML-based clustering algorithms are used to separate data into clusters. The most used are k-Means [82] and Agglomerative Clustering (AC) [83].

Table 4.1: Deep Learning-based Clustering methods.

Method	Architecture	Deep features	Non-clustering loss	Clustering loss	Clustering algorithm
DEC [84]	MLP	Encoder output	Autoencoder reconstruction loss	Cluster assignment hardening	Centroid updates and assignments K-means
IDEC [85]	MLP	Encoder output	Autoencoder reconstruction loss	Cluster assignment hardening	Centroid updates and assignments K-means
DCEC [86]	Convolutional AE (CAE)	Encoder output	Autoencoder reconstruction loss	Cluster assignment hardening	Centroid updates and assignments K-means
DSC [77]	RNN AE (RAE)	Encoder output	Autoencoder reconstruction loss	Cluster assignment hardening	Centroid updates and assignments K-means or AC

DL-based clustering approaches employ different network architectures, structures, loss functions, and training methods to reach their results and enhance the clustering quality. In the following, some methods are briefly reviewed (see Table 4.1).

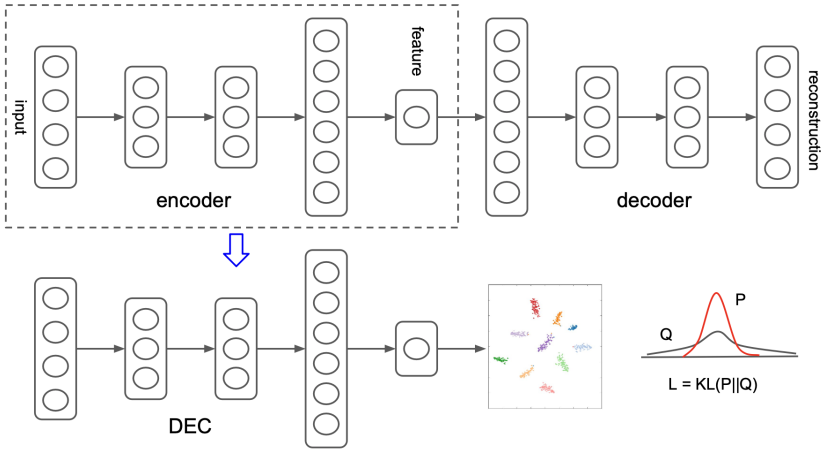

Figure 4-1: DEC network architecture.

Deep Embedded Clustering (DEC) [84] is one of the firsts DL-based Clustering algorithms. It is based on AutoEncoders architecture as an initialization method and uses k-Means for Clustering (see Figure 4-1). Firstly, the method pretrains the

model using an input reconstruction loss function (non-clustering loss). Secondly, a clustering algorithm is adopted to initialize the clustering centers, without considering the decoder. Then, the model is optimized using the CAH loss, and the clustering centers are updated each iteration by minimizing the KL divergence as a loss function.

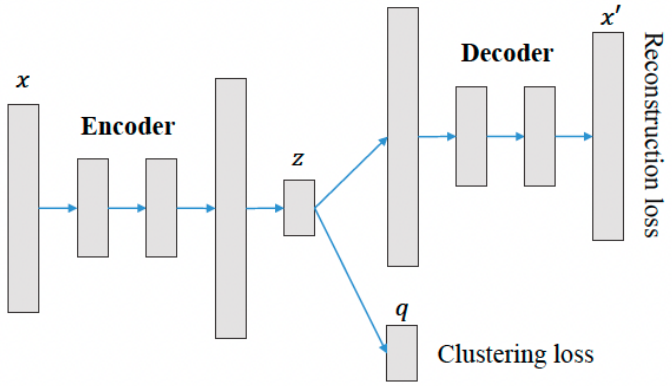

Figure 4-2: IDEC network architecture.

Improved Deep Embedded Clustering (IDEC) [85] is an improvement of DEC, where the decoder layer is not discarded. However, it is used for clustering loss, which is directly attached to the embedding space (see Figure 4-2). Then, the reconstruction loss of AutoEncoders is added to the objective and optimized simultaneously with clustering loss, preserving the local data structure. By saving the local structure, IDEC can prevent the embedded space from being distorted by fine-tuning and ensuring embedded spatial features' representativeness.

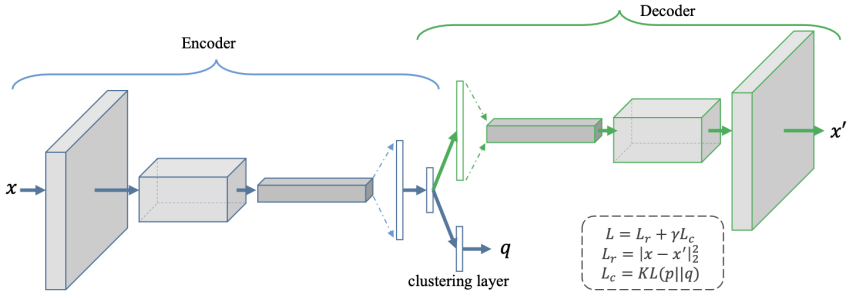

Figure 4-3: DCEC network architecture.

Deep Convolutional Embedded Clustering (DCEC) [86] improves the IDEC algorithm by incorporating a Convolutional AutoEncoder (CAE) (see Figure 4-3), pre-

serving the local data structure. Likewise, the clustering layer is connected to the embedded layer of CAE. Due to its spatial mapping relationships, CAE is used to learn good embedded features, and the Clustering loss divides them into clusters.

The previous DL-based Clustering approaches have been designed for images. Recently, unsupervised learning with Clustering has been proposed in Human Activity Recognition.
Mohammad Abu Alsheikh et al. [87] demonstrated that deep activity recognition models provide better recognition accuracy of human activities using triaxial accelerometers, making them a good approach for weight initialization. Moreover, Harish Haresamudram et al. [88] analyzed the role feature representations play in HAR using wearables. They demonstrated that features extracted from AutoEncoders are better than commonly statistical features, making end-to-end learning architectures particularly suitable for representation learning in Clustering. Lu Bai et al. [89] proposed an unsupervised learning approach. A Deep Learning variational auto-encoder is trained to learn representative features of activities data, grouped regarding the nature of the activity type.

Zhuxi Jiang et al. [90] presented an unsupervised generative Clustering approach within the framework of Variational Auto-Encoder (VAE): Variational Deep Embedding (VaDE). VaDE models the data generative procedure with a Gaussian Mixture Model (GMM), and a Deep Neural Network (DNN) decodes the latent embedding into an observable.

Likewise, Alireza Abedin et al. [91] introduced an unsupervised feature learning process that adopts convolutional auto-encoders to exploit unlabeled data and, some years later, proposed a Deep Clustering architecture,Deep Sensory Clustering (DSC) [77], that learns highly discriminative representations using self-supervision with reconstruction and future prediction tasks informed by feedback from a clustering objective to guide the network towards clustering-friendly representations.

4.2 Deep Inertial Sensory Clustering Architecture

The statement problem is defined as follow. Let us consider the Clustering problem of n samples of inertial signals readings, $X = \{x_1, x_2, ..., x_n\}$ into k clusters, each represented a human activity activity and represented by a centroid $\omega_j, j = 1, ..., k$.

In the following, the proposed architecture of this work is explained. The explanation is divided into two subsections: the *Multi-Task AutoEncoder* employed and the *Clustering Criterion* adopted.

4.2.1 Stage 1: Multi-Task AutoEncoder

An AutoEncoder (AE) [80] is a particular type of feedforward neural network where the input is the same output. In unsupervised learning, AEs are typically used in learning Clustering-friendly representations, which will be used to separate data into clusters. A common AE (see Figure 4-4) is generally composed of two layers: an *encoder* $f_W(.)$ and a *decoder* $g_U(.)$. It aims to find a latent representation for each input sample by minimizing the *mean squared error (MSE)* [92] between its input data and its reconstructed data [93], according to Equation 4.1.

$$min_{W,U} \frac{1}{n} \sum_{i=1}^{n} \|g_U(f_W(x_i)) - x_i\|^2 \tag{4.1}$$

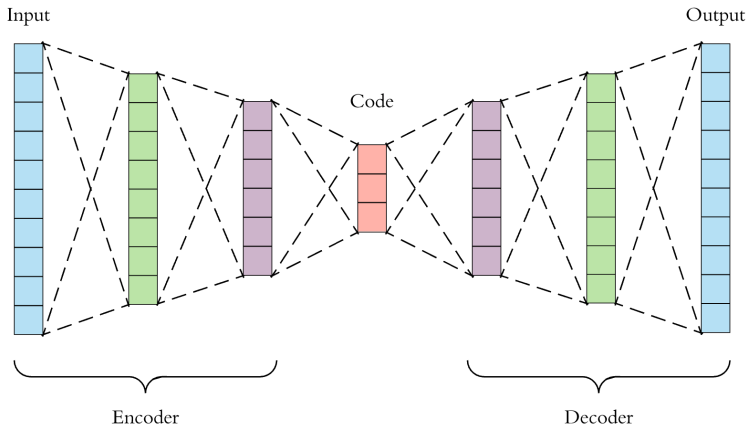

Figure 4-4: Traditional AutoEncoder architecture.

Different AE architectures have been proposed in the literature [94]. A *recurrent neural network (RNN)-based AutoEncoder (RNN AE)* is adopted in this work. RNN architectures have been proved to be powerful for handling sequential data, such as text, sounds, and HAR signals [95, 96]. Compared to multilayer perceptrons (MLPs) and artificial neural networks (ANNs), RNNs have learning parameters called hidden states. These are updated based on the previous and current data information during training, making them well fittable for sequential data.

The *proposed RNN-AE architecture* consists of three RNNs: the encoder ConvGRU [78] and two conditional decoder GRUs [97]. The input to the model is a multi-channel sensor sequence. The encoder ConvGRU reads in this sequence. After the last input has been read, the two decoders GRU takes over and outputs a prediction for the reconstructed sequence and the anticipated sequence.

Recurrent Encoder (Enc_θ)

The recurrent encoder Enc_θ takes as input a raw multi-channel sensor sequence and learns a compacted latent representation to encode the spatio-temporal features of an Activity of Daily Living. In particular, a bi-directional Convolutional Gated Recurrent Unit (ConvGRU) [78], an RNN that combines Gated Recurrent Units (GRUs) [97] with the convolution operation, reads the sensory windowed sequence x in both forward and backward directions. It updates its hidden internal state in each time step according to the received input. The following equations give the update rule for input x_t and the previous output h_{t-1}:

$$z_t = \sigma(W_z \star_n [h_{t-1}; x_t] + b_z) \tag{4.2}$$

$$r_t = \sigma(W_r \star_n [h_{t-1}; x_t] + b_r) \tag{4.3}$$

$$\tilde{h_t} = tanh(W_c \star_n [x_t; z_t \odot h_{t-1}] + b_c) \tag{4.4}$$

$$h_t = (1 - r_t) \odot \tilde{h_t} + r_t \odot h_{t-1} \tag{4.5}$$

where z_t and r_t are the *update gate* and the *forget gate*, σ is the *sigmoid* function,

$\star_n$ represents a *convolution* with a kernel of size n x n, b's are *bias terms* and $\odot$ denotes *element-wise multiplication (or Hadamard product).*

After scanning the entire input sequence x, the final hidden state is reduced in dimensionality through a fully connected layer:

$$z = tanh(\psi(fc(h_t))) \tag{4.6}$$

where ψ is the *Batch Normalization* [98] operation which makes the encoder faster and more stable through normalization of the layers' inputs by re-centering and re-scaling.

The resulting low-dimensional embedded feature $z \in \mathbb{R}^z$ encodes contextual ADL information by representing the spatio-temporal dependencies present in the input sequence x. The operations associated with encoding the input sequence x_i are summarized as:

$$z_i = Enc_\theta(x_i) \tag{4.7}$$

Conditional Recurrent Decoders (Dec_ϕ)

The last hidden state of the encoder ConvGRU, after the dimensionality reduction, is the representation of the input sequence. The decoders GRU are being asked to reconstruct back the input sequence from this representation. A decoder can be of two kinds – *conditional or unconditioned.* A conditional decoder (see Figure 4-5) receives the last generated output hidden state as input. An unconditioned decoder does not receive that input. Using a conditional decoder is beneficial for multiple reasons. It allows the decoder to model multiple modes in the target distribution.

Moreover, if the decoder were given access to the last sequences while generating a particular sequence at training time, it would find it easy to pick up on these correlations. There would only be a minimal gradient that minimises the MSE requiring long-term knowledge about the input sequence. So, the input sequence is removed in a conditioned decoder, and the model is forced to look for information deep inside the encoder.

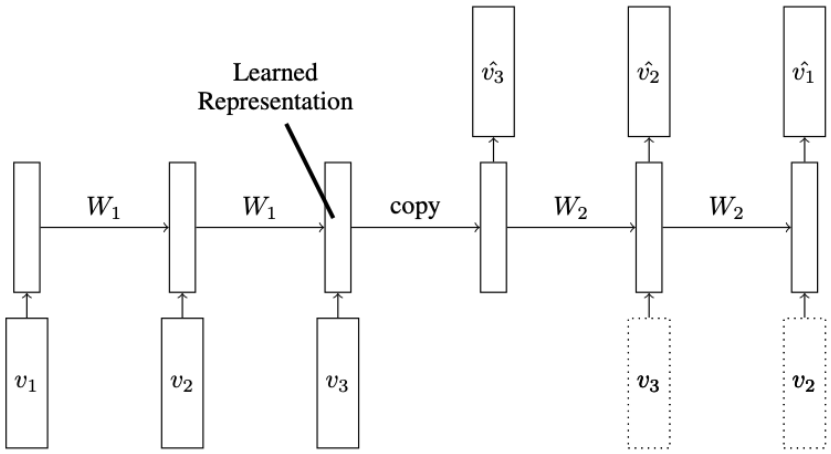

Figure 4-5: Example of Conditional Decoder.

In the proposed architecture, both recurrent decoders Dec_ϕ are conditional, and the following equations define the activation h_t:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{4.8}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1})) \tag{4.9}$$

$$\tilde{h_t} = tanh(W x_t + U(r_t \odot h_{t-1})) \tag{4.10}$$

$$h_t = (1 - z_t) h_{t-1} + z_t \tilde{h_t} \tag{4.11}$$

A context vector is obtained by back projecting the embedded representation z from the encoder into a higher-dimensional space to initialize the decoders' hidden states. Two recurrent decoders GRU simultaneously use the context vector to achieve different self-supervised tasks. As in work [99], the recurrent encoder is shared between decoders with two different expertise. The first decoder is specific to *reconstruct* the temporally inverted input sequence, while the second one learns to *predict* the future sequence that should follow after. So, training the network makes it possible to learn in-depth features that reproduce the input sequence and the information necessary to extrapolate future measurements.

The operations associated with decoding the embedded representation z_i are summarized as:

$$(\bar{y}_i^{rec}, \bar{y}_i^{fut}) = Dec_\phi(z_i) \tag{4.12}$$

where $\bar{y}_i^{rec}$ and $\bar{y}_i^{fut}$ are the reconstructed and the anticipated sequences generated from the input x_i.

Non-Clustering Loss

The *objective* of the Recurrent AutoEncoder is a joint objective function:

$$L_{AE}^{(i)} = L_{rec}^{(i)} + L_{fut}^{(i)} = \|y_i^{rec} - \bar{y}_i^{rec}\|^2 + \left\|y_i^{fut} - \bar{y}_i^{fut}\right\|^2 \tag{4.13}$$

where L_{rec} and L_{fut} indicate the reconstruction loss and the future prediction loss, respectively, and denote the mean square errors between decoder's generated output sequences ($\bar{y}_i^{rec}$ and $\bar{y}_i^{fut}$) and the expected target sequences (y_i^{rec} and y_i^{fut}).

Instead, the *optimal network parameters* of encoder $z_i = Enc_\theta(x_i)$ and decoder $(\bar{y}_i^{rec}, \bar{y}_i^{fut}) = Dec_\phi(z_i)$ are updated by minimizing the reconstruction error:

$$(\theta^*, \phi^*) = min_{\theta,\phi} \frac{1}{n} \sum_{i=1}^{n} L_{AE}^{(i)} \tag{4.14}$$

4.2.2 Stage 2: Clustering Criterion

The reconstruction loss of the AutoEncoder is joined to the objective and optimized along with Clustering loss simultaneously, preserving the local structure of data generating distribution and avoiding the corruption of feature space.

A parametrized Clustering network $f_\mu(.)$ is connected to the AutoEncoder's embedded layer, allowing the estimation of cluster assignment distributions and mapping each embedded point z of input sequence x into a soft label. Then, the Clustering loss L_C is defined as *Kullback-Leibler (KL) divergence* between the distribution of soft labels and the predefined target distribution. Optimizing the Clustering objective makes it possible to refine the feature space and force the network to have Clustering-

friendly representations. In particular, the *Cluster Assignment Hardening (CAH)* is used as a representative centroid-based approach for feature space refinement. The joint optimization criterion, for sample i, is:

$$L^{(i)} = \gamma L_C^{(i)} + L_{AE}^{(i)} \tag{4.15}$$

where the coefficient $\gamma \in [0, 1]$ controls the Clustering objective contribution.
The optimal network parameters are optimized with respect to the global criterion as:

$$(\theta^*, \phi^*, \omega^*) = min_{\ \theta,\phi,\mu} \frac{1}{n} \sum_{i=1}^{n} L^{(i)} \tag{4.16}$$

In the following, the CAH adopted is described.

Cluster Assignment Hardening

The Clustering objective uses the similarities between the data representations and cluster centroids as kernels to compute soft cluster assignments. Then, the CAH loss enforces the soft assignments to have more stringent probabilities.
The Clustering network $f_\mu(.)$ mantains cluster centroids $\mu_j \in \mathbb{R}^{z}{}_{j=1}^{k}$ as trainable weights and maps each embedded point z_i into soft label $Q_i = f_\mu(z_i) = (q_{ij})_{j=1}^{k}$ by following the Student's t-distribution [100]:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2)^{-1}}{\sum_{j'=1}^{k}(1 + \|z_i - \mu_{j'}\|^2)^{-1}} \tag{4.17}$$

where q_{ij} is the j-th entry of q_i, which represents the probability of z_i belonging to cluster j.

By squaring this distribution and then normalizing it, the auxiliary distribution $P_i = (p_{ij})_{j=1}^{k}$ [84] forces assignments to have stricter probabilities (i.e. closer to 0 and 1. P_i helps to improve cluster purity, emphasizing on data points assigned with high confidence, and to prevent large clusters from distorting the hidden feature space.

It is defined as:

$$p_{ij} = \frac{q_{ij}^2 / \sum_i^n q_{ij}}{\sum_{j'=1}^k (q_{ij'}^2 / \sum_{i=1}^n q_{ij'})} \tag{4.18}$$

Lower the KL divergence value, the better we have matched the true distribution with our approximation.

The Clustering loss is defined through minimizing the *Kullback-Leilbler (KL) divergence* [101] between the soft labels and the auxiliary target distribution, via training the layer parameters:

$$L_C^{(i)} = KL(P_i||Q_i) = \sum_i^n \sum_j^k p_{ij}\ log \frac{p_{ij}}{q_{ij}} \tag{4.19}$$

This centroid-based approach needs the cluster centroids to be *initialized only once* at the beginning of the refinement stage. The initial cluster centroids are obtained from classical Clustering algorithms on the embedded representations $z_i = Enc_\theta(x_i)$ after the training of the recurrent AutoEncoder.

The proposed DL-based Clustering architecture is summarized in Figure 4-6.

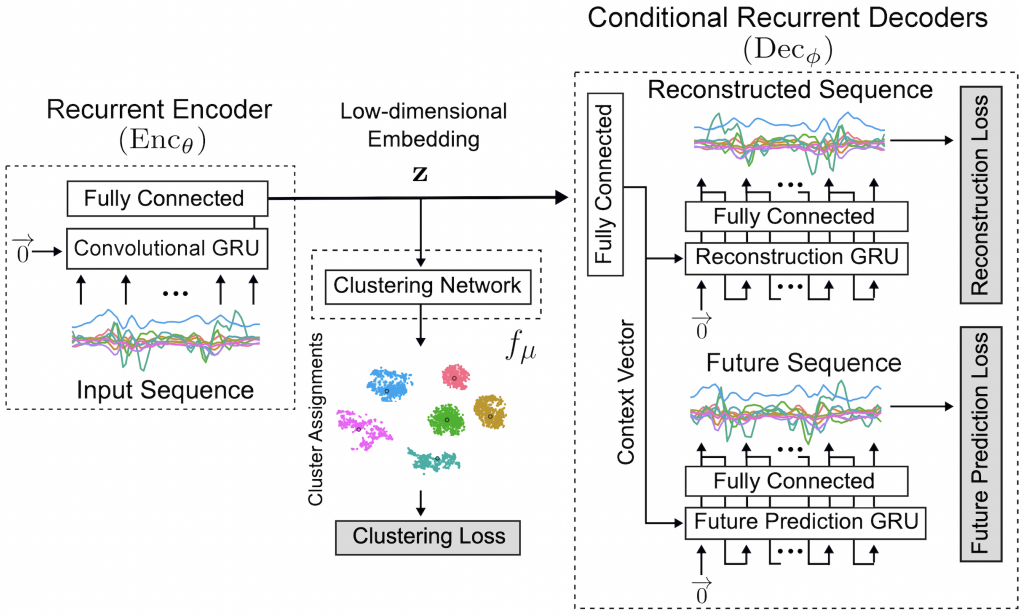

Figure 4-6: Proposed Deep Inertial Sensory Clustering architecture.

4.3 Evaluation Procedure

In this section, the effectiveness of the proposed two-stage Deep Clustering architecture is compared with some main approaches present in the literature (i.e. DEC, DSC).
Firstly, the methods and the datasets are explained. Later, the experiment results are presented and discussed.

The implementation is based on Python and PyTorch.

Network Architecture

The recurrent encoder is a 2-layer bi-directional ConvGRU with 256 hidden units. Instead, the recurrent decoders use uni-directional connections with 256 hidden units. The bottleneck embedded dimension of autoencoders is set to 256. For integrating CAH, only one single layer is used in the clustering network $f_\mu(.)$.
All experiments share the same network architecture.

ML-based Clustering Algorithms

The Machine Learning-based Clustering algorithms adopted are: *k-Means [82]* and *Agglomerative Clustering (AC) [83]*.

k-Means is an iterative Clustering algorithm that aims to discover local maxima in each iteration. The algorithm works in the following steps: *i*) specify the desired number of clusters k; *ii*) randomly assign each data point to a cluster; *iii*) compute cluster centroids; *iv*) re-assign each point to the closest cluster centroid; *v*) re-compute cluster centroids; *vi*)repeat steps 4 and 5 until no improvements are possible.

Hierarchical Clustering is an algorithm that builds a hierarchy of clusters. This algorithm starts with all the data points assigned to a cluster of their own. Then two nearest clusters are merged into the same cluster. In the end, this algorithm terminates when there is only a single cluster left. Hierarchical methods can be either Divisive or Agglomerative. *Agglomerative (AC)* ones begin with n clusters and sequentially combine similar clusters until only one cluster is obtained.

In AC, different *linkage criteria* can be used. The linkage criterion determines which distance to use between sets of observation and can be: *Ward* (minimizes the variance of the clusters being merged), *Average* (uses the Average of the distances of each observation of the two sets), *Complete* (uses the maximum distances between all observations of the two sets).

In k-Means clustering, since one starts with a random choice of clusters, the results produced by running the algorithm many times may differ. For this reason, in the experiments, ten runnings have been done computing the mean and the standard deviation. Instead, results are reproducible in Agglomerative Clustering. k-Means methods usually are less computationally intensive ($O(n)$) and are suited to very large datasets. While, in AC, the computation can be expensive and slow due to its time complexity ($O(n^2)$). In Table 4.2, other differences between the Cluster algorithms are summarized.

All experiments adopted k-Means and Agglomerative Clustering, with three different linkage types (i.e. Ward, Average, Complete).

Table 4.2: Comparison between k-Means and Agglomerative Clustering.

Algorithm	**Parameters**	**Scalability**	**Geometry**
k-Means [82]	Number of clusters	Scalable to a large number of samples	Distances between points
Agglomerative Clustering (AC) [83]	Number of clusters or distance threshold, linkage type and distance	Scalable to a large number of samples across clusters	Any pairwise distance

Optimization Settings

The recurrent AE is pre-trained end-to-end, in mini-batches of size 256, for 100 epochs using *Adam optimizer [102]* with the initial learning rate set to 10 and, after 70 epochs, decayed by a factor of 10. Then, the recurrent AE is refined with the clustering objective till the cluster assignment changes among two consecutive epochs are less than 0.1%. The coefficient γ is set to 0.1. The Cluster centroids are initialized with k-Means and Agglomerative Clustering algorithms.
In all experiments, all those parameters are held constant.

Evaluation Metrics

All clustering methods are evaluated by *Clustering Accuracy (ACC)* [103] and *Normalized Mutual Information (NMI)* [104], widely used in unsupervised learning scenarios. Both metrics lie in the range [0, 1], with 1 being the perfect clustering and 0 being the worst.

The Clustering Accuracy (ACC) takes a cluster assignment (from an algorithm) and a ground truth assignment and then finds the best matching between them. The *Hungarian algorithm* [105] efficiently computes the best mapping, as follow:

$$ACC = max_n \frac{\sum_{i=1}^{n} 1\{l_i = m(c_i)\}}{n} \tag{4.20}$$

where l_i is the true label, c_i the cluster assignment, and m ranges over all possible one-to-one mappings between clusters and labels.

Instead, the Normalized Mutual Information (NMI), between ground-truth labels and the labels obtained by clustering, determines the class labels' entropy reduction, assuming the cluster labels are known. It is used for determining the quality of the clustering and is defined as:

$$NMI(y, c) = \frac{2I(y, c)}{H(y) + H(c)} \tag{4.21}$$

where y is the true label, c is the obtained cluster label, $I(.,.)$ is the mutual information and $H(.)$ is the entropy.

Datasets

The proposed architecture is evaluated and compared on three HAR datasets (as summarized in Table 4.3), according to the work of Alireza et al. in [77]:

- *Human Activity Recognition Using Smartphones Dataset* [55] (referred as *UCI HAR* in the following), which includes 3-axial linear acceleration, 3-axial angular velocity, and gyroscope sensor data of 6 ADLs. The signals were recorded with a Samsung Galaxy S II smartphone at a constant rate of 50Hz, and the activities

were performed by 30 volunteers (19-48 years). Each person performed six activities (walking, walking_upstairs, walking_downstairs, sitting, standing, laying). All the participants were wearing a smartphone on the waist during the experiment execution;

- *Skoda Mini Checkpoint Activity Recognition Dataset* [106] (referred ad *Skoda* in the following), which includes 20 sensors (3-axial acceleration) placed on the left and right upper and lower arm. The signals were recorded at a sampling rate of 98Hz, and the activities were performed by only 1 subject who executed 19 times each activity. The dataset contains 10 activities, which are a subset of 46 activities performed in [107];

- *mHealth Dataset* [108] (referred as *MHEALTH* in the following), which includes 3-axial acceleration from the chest sensor, 2 electrocardiogram signals, 3-axial acceleration from the left-ankle sensor, 3-axial acceleration from the right-lower-arm sensor, 3-axial magnetometer from the left-ankle sensor, 3-axial magnetometer from the right-lower-arm sensor, 3-axial gyroscope from the left-ankle sensor, 3-axial gyroscope from the right-lower-arm sensor. The recorded activities are 12 and are performed by 10 volunteers. The sensors were placed on the subject's chest, right wrist, and left ankle and attached using an elastic strap. All sensors were recorded at a sampling rate of 50Hz.

Table 4.3: A summary of adopted Datasets to evaluate the proposed architecture.

	UCI HAR [55]	**Skoda [106]**	**MHEALTH [108]**
Sensor Sampling Rate	50Hz	33Hz	50Hz
Size of Sliding Window	2.56s	2s	2.56s
# Sensor Channels	9	60	23
# Activities	6	10	12
# Training Windows	7,352	5,653	4,279
# Testing Windows	2,947	705	1,070

Datasets were initially rescaled using per-channel normalization. Then, all the signals have been divided in windows with an overlap between subsequent segments of 50%.

For UCI HAR and MHEALTH, a window of $2.56s$ was used, while for Skoda, a window of $2s$. Furthermore, in MHEALTH, the data relative to electrocardiogram signals have been removed because the proposed method is ad-hoc for accelerometers, gyroscopes and magnetometers.

For all datasets, the *magnitude* of raw 3-axial sensors has been extracted to reduce input data size. However, this does not seem beneficial for UCI HAR and Skoda datasets; therefore, for them, the number of channels has not been modified. This analysis concluded that reducing input size is beneficial for the proposed architecture, but a reasonable number of channels is required.

After the pre-processing phase, the first 50% of sensory measurements in each sample represents the *input sequences* to the proposed architecture. Consequently, the*temporally inverted sequence* of the input is used as the target sequence for the reconstruction task. The *remaining sensory measurements* are considered as the target sequence for future prediction.

4.3.1 Experiments Results

Table 4.4 shows the obtained Clustering results (input space data, autoencoding space data, Deep Clustering) on UCI HAR, Skoda and MHEALTH datasets. The bold font represents the best method for each dataset. The results demonstrate the effectiveness of Deep Clustering approaches against traditional Clustering approaches on input space data and autoencoding space. Moreover, they offer a better performance margin over representative Deep Clustering baselines proposed for image data (DEC, IDEC, DCEC) and sensory readings (DSC).

In the case of the UCI HAR dataset, the proposed architecture overcomes all the baseline approaches, even comparing the DSC (Ward) results declared by Alireza et Al. It is mainly due to a better representation learning of the spatio-temporal features encoded by the ConvGRU layer.

In the Skoda dataset, there are no relevant increases regarding the state of the art approaches. The obtained results are similar to DSC and confirm the dataset's difficulty to be clustered. This is probably due to the dataset sampling rate (33 Hz),

Table 4.4: Comparison of Clustering performances on HAR Datasets. For each Dataset (UCI HAR, Skoda, MHEALTH), the bold font represents the best method. * symbol means that literature results have not been reached.

	UCI HAR				**Skoda**				**MHEALTH**			
	Train		**Test**		**Train**		**Test**		**Train**		**Test**	
	NMI	**ACC**	**NMI**	**ACC**	**NMI**	**ACC**	**NMI**	**ACC**	**NMI**	**ACC**	**NMI**	**ACC**
Traditional Clustering on Input Data Space												
k-Means	49.36 (2.85)	45.04 (2.07)	45.90 (1.38)	40.69 (0.78)	38.54 (1.79)	39.18 (1.11)	36.93 (0.92)	38.59 (1.84)	41.85 (3.03)	43.95 (2.19)	42.14 (3.24)	44.41 (2.29)
AC-Average	1.18	19.16	1.8	18.29	1.51	13.88	18.38	14.37	12.12	10.35	9.63	10.19
AC-Complete	3.45	19.56	19.22	31.69	33.29	31.32	31.86	28.07	23.49	42.10	30.02	13.74
AC-Ward	40.73	42.26	47.71	43.26	41.79	41.35	36.99	33.19	48.07	48.26	48.54	48.60
Traditional Clustering on AutoEncoding Space												
(proposed) k-Means	54.84 (0.41)	55.27 (0.17)	48.27 (0.87)	54.62 (0.40)	50.65 (0.46)	47.83 (0.60)	47.87 (2.19)	43.10 (1.70)	65.14 (0.30)	55.27 (1.02)	64.88 (0.23)	54.04 (1.65)
(proposed) AC-Average	43.82	39.17	41.26	35.12	29.74	28.82	29.63	24.12	30.80	13.69	31.24	13.16
(proposed) AC-Complete	27.77	44.12	28.91	40.62	35.48	34.62	44.13	37.31	46.58	36.22	50.21	30.24
(proposed) AC-Ward	64.21*	63.14*	51.75	60.77	52.85	47.92	54.10	43.45	68.85	58.31	70.98	59.52
End-to-End Deep Clustering												
DEC [84]	55.57	49.93	55.20	49.10	46.86	41.02	46.58	41.08	50.52	43.71	51.68	45.11
IDEC [85]	55.76	51.78	54.43	51.02	49.65	46.70	48.93	40.56	51.28	42.59	52.34	44.76
DCEC [86]	52.77	48.39	53.12	48.88	42.90	39.56	45.51	42.24	44.15	23.51	45.13	24.29
DSC (k-Means) [77]	65.13	63.98	62.81	58.98	**57.56**	51.84	56.57	51.36	58.26	53.02	58.31	53.02
DSC (Ward) [77]	60.28*	60.76*	57.40*	57.96*	52.56	50.78	57.65	51.95	59.42	51.57	60.91	53.33
(proposed) DISC (k-Means)	**85.72**	**92.08**	80.81	**87.99**	52.93	54.45	52.93	**54.45**	**68.15**	**53.96**	**69.38**	**54.83**
(proposed) DISC (Ward)	85.47	81.86	**81.91**	79.40	54.91	**54.57**	**55.93**	53.45	63.23	50.74	61.29	48.82

which is insufficient to register significant information about the desired activity.

Concerning the MHEALTH dataset, the variations with the DSC approach are better considering only the NMI metric. However, by reducing the input size, the computation is faster in training time.

High-dimensional space visualization

The visualization of high-dimensional data using a two-dimensional scatter plot is made by using *Principal Component Analysis (PCA)* [109] and *t-Distributed Stochastic Neighbor Embedding (t-SNE)* [100].

PCA uses the correlation between the dimensions and provides a minimum number of variables that keep the maximum amount of variation or information about how the

original data is distributed. It is based on mathematics and, more specifically, on the eigenvalues and eigenvectors of the data matrix. These eigenvectors of the covariance matrix have the property that they point along with the significant directions of variation in the data, which indicates the maximum variation in a dataset.

Contrary to PCA, t-SNE is not a mathematical technique but a probabilistic one. It minimizes the divergence between two distributions: a distribution that measures pairwise similarities of the input objects and a distribution that estimates pairwise similarities of the corresponding low-dimensional points in the embedding. However, t-SNE is computationally quite heavy and therefore, there are some limitations to using this technique, especially in the case of very high dimensional data. For this reason, another dimensionality reduction technique is commonly applied before using t-SNE.

The dimensionality reduction technique adopted is the following:

- The PCA reduction algorithm reduces the embedded features of each sample to 50 components;
- The t-SNE decreases the obtained 50 components to 2;
- The 2 components are represented in a two-dimensional scatter plot.

This cluster approach uses the combination between PCA and t-SNE because of the high variety of embedded representation, not for the high dimensional dimension of features (256). So, it is possible to use the benefits of both reduction techniques and get a good representation that considers reasonably the maximum amount of variation.

Figures 4-7, 4-8, 4-9 show the progression of the feature space to the final clustering-oriented embedding space performed with the proposed architecture, for each dataset. In particular, the original input data space is shown. Then, the autoencoding space after the first stage of pretraining, and, finally, the feature space of the second stage of the proposed architecture.

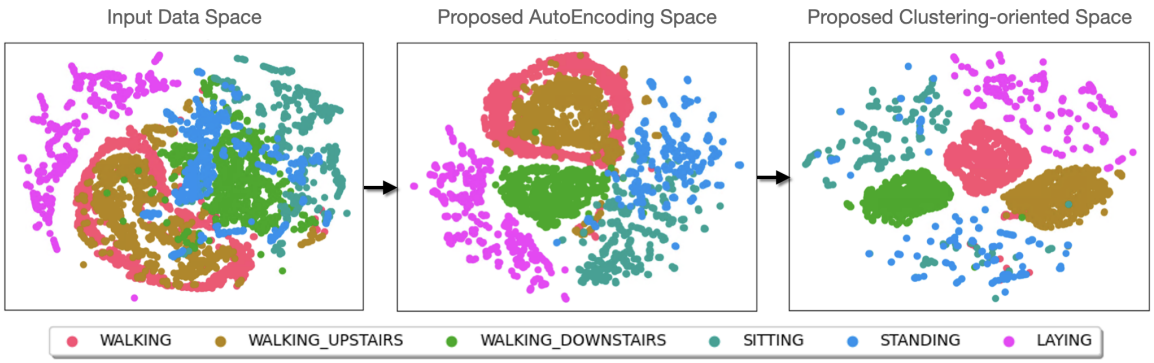

Figure 4-7: Feature space progression of the UCI HAR Dataset with DISC (k-Means initialization).

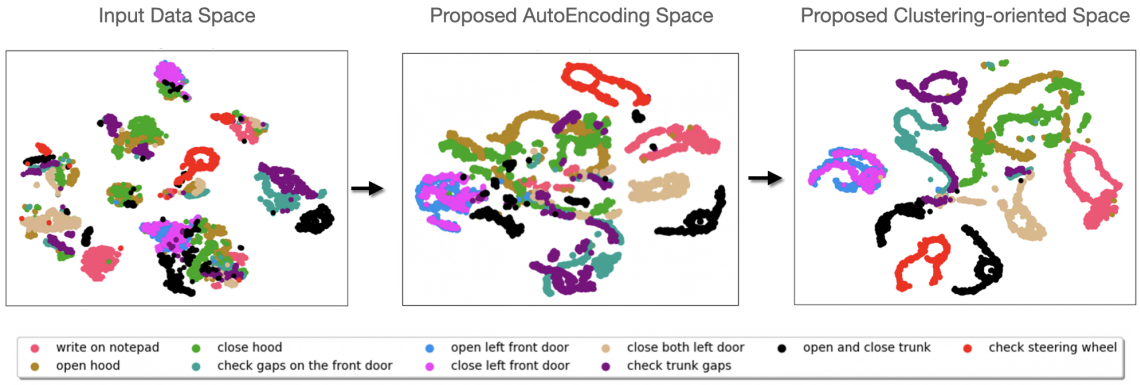

Figure 4-8: Feature space progression of the Skoda Dataset with DISC (Ward initialization).

In the case of the UCI HAR dataset (see Figure 4-7), the proposed architecture discovers well-defined and separated clusters of activity segments with strong correspondence to the ground-truth labels. In particular, three activities (walking, walking upstairs and walking downstairs) are correctly determined, while the others (sitting, standing, lying) are well delimited but more dispersed. Probably due to the high number of participants in the dataset, which increase the inter-variability of activities.

Instead, the Skoda (see Figure 4-8) and MHEALTH (see Figure 4-9) datasets have less clear and distinct clusters, confirming the obtained metrics of NMI and ACC. The reason may be to search into the number of activities, 10 and 12, and the number of training samples, which is low considering the number of classes.

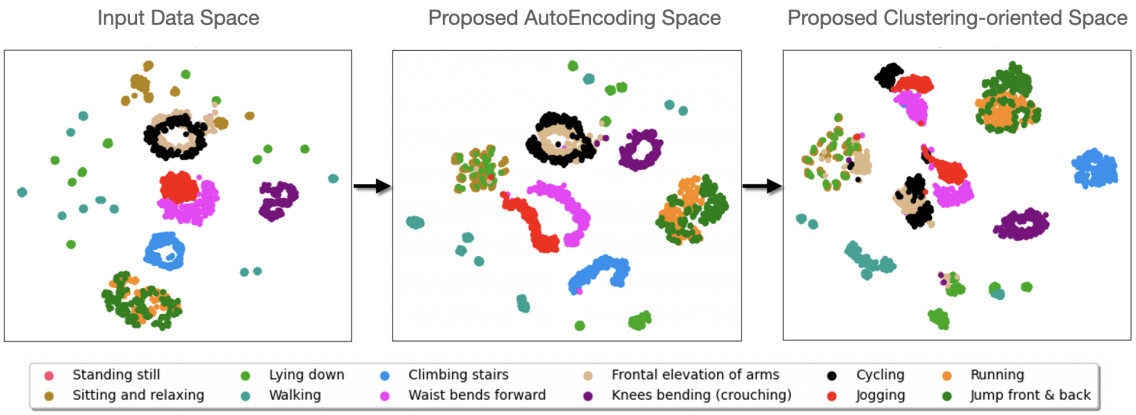

Figure 4-9: Feature space progression of the MHEALTH Dataset with DISC (k-Means initialization).

So, the proposed architecture has difficulty in grouping input data because of the size of the datasets.

Chapter 5

Conclusions

Our imagination is the only limit to what we can hope to have in the future.

Charles F. Kettering

In this thesis, Model-Centric and Data-Centric AI are analyzed and used to propose a procedure to personalize models in Human Activity Recognition. Incremental Learning is a technique that allows obtaining personalized models by learning from streaming data and adapting to new and changing environments. In Human Activity Recognition, Incremental Learning is largely adopted to fit general models to a given new user. This is helpful in real mobile applications. This work employs deep learning approaches in an Incremental Learning procedure compared with a previous method based on Learn++. Experiments carried out on three different datasets showed that, overall, Deep Learning outperforms Learn++. In particular, two different networks have been evaluated: a ResNet and a Simplified CNN. Both CNNs demonstrated to be faster than Learn++ to adapt to a new user, thus demonstrating to require less user interaction than Learn++.

Furthermore, the Continuous Learning Platform is proposed to obtain a larger dataset to define an optimal subject-independent model, essential in the personalization procedures.

CLP enables the integration and the distribution of data coming from heterogeneous sources. In particular, a homogenization procedure has been designed to uniform signals and align labels according to a specific configuration and annotation standard. Eight datasets available in the literature have been integrated and used to train a convolutional neural network and a recurrent neural network. The experimentation results confirm the improvement of performances by combining existing datasets from different smartphones and various contexts.

Finally, an Unsupervised Learning approach is defined to annotate unlabelled data without the user interaction. The true user labels are crucial in personalizing a model due to their inter and intra-variability information. This work defines a DL-based Clustering architecture that learns highly discriminative spatio-temporal representations using self-supervision with reconstruction and future prediction tasks. Then, after the initialization of cluster centroids, a clustering criterion uses the similarities between the obtained in-depth features to generate cluster assignments with strong semantic correspondence to distinct human activities. Two distinct Machine Learning-based Clustering algorithms have been used: k-Means and Agglomerative Clustering with Ward linkage. The experiments demonstrate the effectiveness of the proposed architecture, concerning the state-of-the-art approaches, on the sensory readings clustering task by performing it on three different HAR datasets.

5.1 Future Works

Various tests and experiments have been left for the future due to lack of time (i.e. the experiments with real data are very time consuming, demanding even days to finish a single run). Future work concerns a deeper analysis of particular approaches and new proposals to try different methods.

Relying on this work, designing a software component that interfaces the Continuous Learning Platform and an Android application to develop personalized models is currently on schedule. In particular, the CLP platform provides specific user data by a query to train a model with particular characteristics and makes available an

optimal subject-independent model necessary to the personalization. The overall aim is an application able to personalize models by minimal interaction with the user. So, the proposed Unsupervised Learning approach should be integrated into the personalization pipeline.

Additional prospective directions include polishing the CLP Web application and an intensive test of the overall CLP platform with real-world applications, integrating new existing datasets to improve the uniform dataset's variability.

Since Unsupervised Learning is currently one of the most active research areas, new studies regarding the automatic Clustering of inertial sensory readings are necessary. Literature approaches have reached promising results on image data compared to the Human Activity Recognition field. However, in HAR, more and more researchers are introducing new proposals. Some interesting ideas regarding this work are the following:

- testing with new architectures to obtain a better representation learning of the input data;
- experimenting better ML-based Clustering algorithms to initialize cluster centroids;
- testing new clustering criteria. An idea may be the research of a more fittable distribution to generate soft labels assignment. This could be done by finding the best distribution that fists HAR data by analyzing the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) values.

References

[1] NG Andrew. A chat with andrew on mlops: From model-centric to data-centric ai, 2021.

[2] Anna Ferrari, Daniela Micucci, Marco Mobilio, and Paolo Napoletano. On the personalization of classification models for human activity recognition. *IEEE Access*, 8:32066–32079, 2020.

[3] Anna Ferrari, Daniela Micucci, Marco Mobilio, and Paolo Napoletano. A framework for long-term data collection to support automatic human activity recognition. In *Intelligent Environments 2019: Workshop Proceedings of the 15th International Conference on Intelligent Environments*, volume 26, page 367. IOS Press, 2019.

[4] Daniela Micucci, Marco Mobilio, Paolo Napoletano, and Francesco Tisato. Falls as anomalies? an experimental evaluation using smartphone accelerometer data. *Journal of Ambient Intelligence and Humanized Computing*, 8(1):87–99, 2017.

[5] Anna Ferrari, Daniela Micucci, Marco Mobilio, and Paolo Napoletano. Hand-crafted features vs residual networks for human activities recognition using accelerometer. In *Proceedings of the IEEE International Symposium on Consumer Technologies (ISCT)*, 2019.

[6] Simone Bianco, Remi Cadene, Luigi Celona, and Paolo Napoletano. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277, 2018.

[7] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.

[8] MA Habib, MS Mohktar, SB Kamaruzzaman, KS Lim, TM Pin, and F Ibrahim. Smartphone-based solutions for fall detection and prevention: challenges and open issues. *Sensors*, 14:7181–7208, 2014.

[9] Carlos Medrano, Raul Igual, Inmaculada Plaza, and Manuel Castro. Detecting falls as novelties in acceleration patterns acquired with smartphones. *PloS one*, 9(4):e94811, 2014.

[10] Muhammad Shoaib, Stephan Bosch, Ozlem Incel, Hans Scholten, and Paul Havinga. A survey of online activity recognition using mobile phones. *Sensors*, 15(1):2059–2085, 2015.

[11] DataReportal. Digital 2020 reports.

[12] Akin Avci, Stephan Bosch, Mihai Marin-Perianu, Raluca Marin-Perianu, and Paul Havinga. Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey. In *In Proceedings of the International Conference on Architecture of Computing Systems (ARCS)*, 2010.

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

[14] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 119:3–11, 2019.

[15] Andrea Zunino, Jacopo Cavazza, and Vittorio Murino. Revisiting human action recognition: Personalization vs. generalization. In *Proceedings of the International Conference on Image Analysis and Processing (ICAR)*, 2017.

[16] Nicholas D Lane, Ye Xu, Hong Lu, Shaohan Hu, Tanzeem Choudhury, Andrew T Campbell, and Feng Zhao. Enabling large-scale human activity inference on smartphones using community similarity networks (csn). In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 355–364, 2011.

[17] Christian Krupitzer, Timo Sztyler, Janick Edinger, Martin Breitbach, Heiner Stuckenschmidt, and Christian Becker. Hips do lie! a position-aware mobile fall detection system. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. IEEE, 2018.

[18] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248, 2018.

[19] Sebastian Thrun. Is learning the -th thing any easier than learning the first? 1996.

[20] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[21] Pekka Siirtola and Juha Röning. Incremental learning to personalize human activity recognition models: The importance of human ai collaboration. *Sensors*, 19(23):5151, 2019.

[22] Anna Ferrari, Marco Mobilio, Daniela Micucci, and Paolo Napoletano. On the homogenization of heterogeneous inertial-based databases for human activity recognition. In *2019 IEEE World Congress on Services (SERVICES)*, volume 2642, pages 295–300. IEEE, 2019.

[23] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, and Paul JM Havinga. Fusion of smartphone motion sensors for physical activity recognition. *Sensors*, 14(6):10146–10176, 2014.

[24] Andrew P Hills, Najat Mokhtar, and Nuala M Byrne. Assessment of physical activity and energy expenditure: an overview of objective measures. *Frontiers in nutrition*, 1:5, 2014.

[25] Erika Rovini, Carlo Maremmani, and Filippo Cavallo. How wearable sensors can support parkinson's disease diagnosis and treatment: a systematic review. *Frontiers in neuroscience*, 11:555, 2017.

[26] Prabitha Urwyler, Reto Stucki, Luca Rampa, René Müri, Urs P Mosimann, and Tobias Nef. Cognitive impairment categorized in community-dwelling older adults with and without dementia using in-home sensors that recognise activities of daily living. *Scientific reports*, 7(1):1–9, 2017.

[27] Geoffrey E Hinton, Terrence Joseph Sejnowski, et al. *Unsupervised learning: foundations of neural computation*. MIT press, 1999.

[28] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[29] S. P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28:129–136, 1982.

[30] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[31] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[32] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[34] Enrique Garcia-Ceja and Ramon Brena. Building personalized activity recognition models with scarce labeled data based on class similarities. In *International conference on ubiquitous computing and ambient intelligence*, pages 265–276. Springer, 2015.

[35] Ramin Fallahzadeh and Hassan Ghasemzadeh. Personalization without user interruption: Boosting activity recognition in new subjects using unlabeled data. In *Proceedings of the 8th International Conference on Cyber-Physical Systems*, pages 293–302, 2017.

[36] Ali Akbari and Roozbeh Jafari. Personalizing activity recognition models through quantifying different types of uncertainty using wearable sensors. *IEEE Transactions on Biomedical Engineering*, 67(9):2530–2541, 2020.

[37] HM Sajjad Hossain and Nirmalya Roy. Active deep learning for activity recognition with context aware annotator selection. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2019.

[38] Taesik Gong, Yeonsu Kim, Jinwoo Shin, and Sung-Ju Lee. Metasense: few-shot adaptation to untrained conditions in deep mobile sensing. In *Proceedings of the Conference on Embedded Networked Sensor Systems (SenSys)*, 2019.

[39] Seyed Ali Rokni, Marjan Nourollahi, and Hassan Ghasemzadeh. Personalized human activity recognition using convolutional neural networks. *CoRR*, abs/1801.08252, 2018.

[40] Tong Yu, Yong Zhuang, Ole J Mengshoel, and Osman Yagan. Hybridizing personal and impersonal machine learning models for activity recognition on mobile devices. In *MobiCASE*, pages 117–126, 2016.

[41] Robi Polikar, Lalita Upda, Satish S Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, 31(4):497–508, 2001.

[42] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[43] Yong Luo, Liancheng Yin, Wenchao Bai, and Keming Mao. An appraisal of incremental learning methods. *Entropy*, 22(11), 2020.

[44] Alexander Gepperth and Barbara Hammer. Incremental learning algorithms and applications. In *European symposium on artificial neural networks (ESANN)*, 2016.

[45] Yi-Min Wen and Bao-Liang Lu. Incremental learning of support vector machines by classifier combining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 904–911. Springer, 2007.

[46] Thomas Cederborg, Ming Li, Adrien Baranes, and Pierre-Yves Oudeyer. Incremental local online gaussian mixture regression for imitation learning of multiple tasks. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 267–274. IEEE, 2010.

[47] Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, volume 1, pages 288–293. Morgan Kaufmann, 2000.

[48] Duy Nguyen-Tuong and Jan Peters. Local gaussian process regression for real-time model-based robot control. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 380–385. IEEE, 2008.

[49] Kai Ma and Jezekiel Ben-Arie. Compound exemplar based object detection by incremental random forest. In *2014 22nd International Conference on Pattern Recognition*, pages 2407–2412. IEEE, 2014.

[50] João Roberto Bertini, Maria do Carmo Nicoletti, and Liang Zhao. Ensemble of complete p-partite graph classifiers for non-stationary environments. In *2013 IEEE Congress on Evolutionary Computation*, pages 1802–1809. IEEE, 2013.

[51] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the European conference on computational learning theory (EuroCOLT)*, 1995.

[52] Viktor Losing, Barbara Hammer, and Heiko Wersing. Choosing the best algorithm for an incremental on-line learning task. In *In Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, 2016.

[53] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access, 2017.

[54] Jorge-L Reyes-Ortiz, Luca Oneto, Albert Samà, Xavier Parra, and Davide Anguita. Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171:754–767, 2016.

[55] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *In Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, 2013.

[56] Pekka Siirtola and Juha Röning. Recognizing human activities user-independently on smartphones based on accelerometer data. *IJIMAI*, 1(5):38–45, 2012.

[57] Pekka Siirtola, Heli Koskimäki, and Juha Röning. Openhar: A matlab toolbox for easy access to publicly open human activity data sets. In *Proceedings of the ACM International Joint Conference and International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers (UbiComp)*, 2018.

[58] Daniela Micucci, Marco Mobilio, and Paolo Napoletano. Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences*, 7(10):1101, 2017.

[59] Ming Zeng, Le T Nguyen, Bo Yu, Ole J Mengshoel, Jiang Zhu, Pang Wu, and Joy Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *6th international conference on mobile computing, applications and services*, pages 197–205. IEEE, 2014.

[60] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[61] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[62] Allan Peter Davis, Thomas C Wiegers, Phoebe M Roberts, Benjamin L King, Jean M Lay, Kelley Lennon-Hopkins, Daniela Sciaky, Robin Johnson, Heather Keating, Nigel Greene, et al. A ctd–pfizer collaboration: manual curation of 88 000 scientific articles text mined for drug–disease and drug–phenotype interactions. *Database*, 2013, 2013.

[63] C Metz. Google's hand-fed ai now gives answers, not just search results, 2016. *Wired [Online.*

[64] James Bartlett, Vinay Prabhu, and John Whaley. Acctionnet: A dataset of human activity recognition using on-phone motion sensors. In *Proceedings of the International Conference on Machine Learning (ICML 17)*, 2017.

[65] Alex Adim Obinikpo and Burak Kantarci. Big data aggregation in the case of heterogeneity: a feasibility study for digital health. *International Journal of Machine Learning and Cybernetics*, 10(10):2643–2655, 2019.

[66] George Vavoulas., Charikleia Chatzaki., Thodoris Malliotakis., Matthew Pediaditis., and Manolis Tsiknakis. The mobiact dataset: Recognition of activities of daily living using smartphones. In *Proceedings of the International Conference on Information and Communication Technologies for Ageing Well and e-Health - Volume 1: ICT4AWE, (ICT4AGEINGWELL 2016)*, pages 143–151. INSTICC, SciTePress, 2016.

[67] Casilari Eduardo and A. Jose. Umafall: Fall detection dataset (universidad de malaga). 2017.

[68] M. Shoaib, Johan Scholten, and Paul J.M. Havinga. Towards physical activity recognition using smartphone sensors. In *10th IEEE International Conference on Ubiquitous Intelligence and Computing, UIC 2013*, pages 80–87, United States, December 2013. IEEE Computer Society. 10.1109/UIC-ATC.2013.43 ; null ; Conference date: 18-12-2013 Through 20-12-2013.

[69] Pekka Siirtola and Juha Röning. Recognizing human activities user-independently on smartphones based on accelerometer data. *International Journal of Interactive Multimedia and Artificial Intelligence*, 1(5), 2012.

[70] Daniela Micucci, Marco Mobilio, and Paolo Napoletano. Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences*, 7(10), 2017.

[71] Mohammad Malekzadeh, Richard G. Clegg, Andrea Cavallaro, and Hamed Haddadi. Mobile sensor data anonymization. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, IoTDI '19, pages 49–58, New York, NY, USA, 2019. ACM.

[72] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L Littman. Activity recognition from accelerometer data. In *Aaai*, volume 5, pages 1541–1546. Pittsburgh, PA, 2005.

[73] D Gordon E Robertson, Graham E Caldwell, Joseph Hamill, Gary Kamen, and Saunders Whittlesey. *Research methods in biomechanics*. Human kinetics, 2013.

[74] Ryo Takeda, Giulia Lisco, Tadashi Fujisawa, Laura Gastaldi, Harukazu Tohyama, and Shigeru Tadano. Drift removal for improving the accuracy of gait parameters using wearable sensor systems. *Sensors*, 14(12):23230–23247, 2014.

[75] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

[76] Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data clustering: theory, algorithms, and applications*. SIAM, 2020.

[77] Alireza Abedin, Farbod Motlagh, Qinfeng Shi, Hamid Rezatofighi, and Damith Ranasinghe. Towards deep clustering of human activities from wearables. In *Proceedings of the 2020 International Symposium on Wearable Computers*, pages 1–6, 2020.

[78] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015.

[79] Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5-6):183–197, 1991.

[80] I Goodfellow, Y Bengio, and A Courville Deep Learning. Cambridge, ma, 2016.

[81] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[82] Greg Hamerly and Charles Elkan. Learning the k in k-means. *Advances in neural information processing systems*, 16:281–288, 2003.

[83] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

[84] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR, 2016.

[85] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *Ijcai*, pages 1753–1759, 2017.

[86] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *International conference on neural information processing*, pages 373–382. Springer, 2017.

[87] Mohammad Abu Alsheikh, Ahmed Selim, Dusit Niyato, Linda Doyle, Shaowei Lin, and Hwee-Pink Tan. Deep activity recognition models with triaxial accelerometers. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[88] Harish Haresamudram, David V Anderson, and Thomas Plötz. On the role of features in human activity recognition. In *Proceedings of the 23rd International Symposium on Wearable Computers*, pages 78–88, 2019.

[89] Lu Bai, Chris Yeung, Christos Efstratiou, and Moyra Chikomo. Motion2vector: unsupervised learning in human activity recognition using wrist-sensing data. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, pages 537–542, 2019.

[90] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016.

[91] Alireza Abedin Varamin, Ehsan Abbasnejad, Qinfeng Shi, Damith C Ranasinghe, and Hamid Rezatofighi. Deep auto-set: A deep auto-encoder-set network for activity recognition using wearables. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 246–253, 2018.

[92] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[93] GE Hinton and RR Salakhutdinov. Reducing the dimensionality of data with neural networks. science, 313 (5786), 504-507. *PRABAHARAN POORNACHANDRAN is a professor at Amrita Vishwa Vidyapeetham. He has more than two decades of experience in Computer Science and Security areas. His areas of interests are Malware, Critical Infrastructure security, Complex Binary analysis, AI and Machine Learning*, 2006.

[94] Ganggang Dong, Guisheng Liao, Hongwei Liu, and Gangyao Kuang. A review of the autoencoder and its variants: A comparative perspective from target recognition in synthetic-aperture radar images. *IEEE Geoscience and Remote Sensing Magazine*, 6(3):44–68, 2018.

[95] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 273–278. IEEE, 2013.

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

[97] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[98] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[99] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852. PMLR, 2015.

[100] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[101] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[102] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[103] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[104] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:2837–2854, 2010.

[105] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[106] Thomas Stiefmeier, Daniel Roggen, Georg Ogris, Paul Lukowicz, and Gerhard Tröster. Wearable activity tracking in car manufacturing. *IEEE Pervasive Computing*, 7(2):42–50, 2008.

[107] Thomas Stiefmeier, Daniel Roggen, and Gerhard Troster. Fusion of string-matched templates for continuous activity recognition. In *2007 11th IEEE International Symposium on Wearable Computers*, pages 41–44. IEEE, 2007.

[108] Oresti Banos, Rafael Garcia, Juan A Holgado-Terriza, Miguel Damas, Hector Pomares, Ignacio Rojas, Alejandro Saez, and Claudia Villalonga. mhealthdroid: a novel framework for agile development of mobile health applications. In *International workshop on ambient assisted living*, pages 91–98. Springer, 2014.

[109] Ian Jolliffe. Principal component analysis. *Encyclopedia of statistics in behavioral science*, 2005.