Exploring the Computational Cost of Machine Learning at the Edge for Human-Centric Internet of Things

Oihane Gómez-Carmona[a], Diego Casado-Mansilla[a], Frank Alexander Kraemer[b], Diego López-de-Ipiña[a], Javier García-Zubia[c]

[a]*Deustotech - University of Deusto, Spain*
[b]*Department of Information Security and Communication Technology - Norwegian University of Science and Technology, NTNU*
[c]*Faculty of Engineering - University of Deusto, Spain*

Abstract

In response to users' demand for privacy, trust and control over their data, executing machine learning tasks at the edge of the system has the potential to make the Internet of Things (IoT) applications and services more human-centric. This implies moving complex computation to a local stage, where edge devices must balance the computational cost of the machine learning techniques to meet the available resources. Thus, in this paper, we analyze all the factors affecting the classification process and empirically evaluate their impact in terms of performance and cost. We put the focus on Human Activity Recognition (HAR) systems, which represent a standard type of classification problems in human-centered IoT applications. We present a holistic optimization approach through input data reduction and feature engineering that aims to enhance all the stages of the classification pipeline and integrate both inference and training at the edge. The results of the conducted evaluation show that there is a highly non-linear trade-off to make between the computational cost, in terms of processing time, and the achieved classification accuracy. In the presented case of study, the computational effort can be reduced by 80 % assuming a decline of the classification accuracy of only 3 %. The potential impact of the optimization strategy highlights the importance of understanding the initial data and studying the most relevant characteristics of the signal to meet the cost-accuracy requirements. This would contribute to bringing embedded machine learning to the edge and, hence, creating spaces where human and machine intelligence could collaborate.

Keywords: `Internet of Things, Edge Computing, Machine Learning, Cost-Accuracy, Edge Intelligence, Embedded systems`

1. Introduction

Connecting everyday objects with sensors to create smart environments is transforming human lifestyles in domains like the workplace [1] or the city [2]. In these domains, we can observe how the Internet of Things (IoT) can enhance health [3], wellness [4] or promote sustainable practices [5]. Smart environments sense the physical world, give meaning to the obtained information and trigger suitable reactions. For example, a workplace augmented with IoT can detect and classify unhealthy habits like bad postures or inadequate hydration, and notify those harmful practices to end users [6]. For that, data need to be processed for decision-making, which often involves inference and machine learning techniques. Following the classic IoT architecture, such computation is performed on dedicated cloud services [7]. While this might be the typical architecture to implement, it increases the human adoption gap of IoT due to privacy concerns such as the loss of control over personal data [8] and the feeling that classic IoT-based systems leave the user involvement aside from the equation [9].

To reduce this gap and improve the interweaving between human beings and technology, some scholars [10, 11] demand more human-centric approaches for sensing the physical world that enable confident spaces where human and machine intelligence ally. For Garcia et al. [12], this also means to include humans in the control loop, making them part of the decision process and giving them increased control over the management of the information they generate. Edge computing deserves attention in this human-centric endeavor as it moves the processing power closer to the data source, that is, to a local stage that boosts what is called privacy-by-design [13]. This provides users with local control which is a suitable means to lower the reluctance towards emerging technologies' adoption. On top of these positive aspects, edge architectures have to cope with their own flaws. As an example, while cloud computing platforms can easily scale up if temporary demand for resources requires it, edge computing platforms are typically resource-constrained and cannot just increase their computational power. Thus, the computational need of data processing tasks is a critical factor to consider in such edge designs, and new strategies have to be devised.

To overcome the presented issues, most of the reviewed body of literature adopt hybrid architectures in which demanding tasks are relegated to the cloud or distributed devices (e.g., model (re)training is outsourced to powerful devices and inference is performed in tiny equipment on the edge) [14, 15]. Although we acknowledge that this solves the computing prob-

*Corresponding author
Email address: `oihane.gomezc@deusto.es` (Oihane Gómez-Carmona)

lems, in the vision of this new human-centric IoT, externalizing this personal and sensitive data may compromise the privacy of the user [12]. Therefore, in this article, we continue the work presented in [16] and extend it proposing a new computational model that preserves the data on the edge of the network, both in the training and inference phases. Specifically, we put the focus on Human Activity Recognition (HAR) systems which represent a standard type of classification problems in human-centered IoT applications [17]. The presented case of study refers to a drinking monitor system on the edge where acceleration data, obtained from a wrist-located inertial sensor, is processed to detect the drinking activity of the user in an office environment.

From a methodological point of view, we identified and broke down the different factors that take part in all the stages of the machine learning process (our pipeline includes training and inference). Based on this analysis, we present a holistic optimization strategy through data reduction that combines those factors. Then, we measure the computational cost in terms of execution time, also known as computation latency, and the classification performance, in terms of accuracy, when we apply the proposed strategy. We evaluate these metrics in two well-known resource-constrained devices (Raspberry Pi 3B+ and Raspberry Zero W).

Throughout the manuscript, we show that we can mitigate the limitations of constrained devices if an optimization process is carried out from the early stages of the classification pipeline (i.e., reducing the data size to simplify the input and applying feature engineering techniques). Besides, we analyze how small reductions in the prediction accuracy can potentially enable substantial improvements in the computational cost, particularly for the training stage of the different supervised methods tested. Our results are promising to preserve the personal data on the edge and still provide good inference results. Thus, to find optimal solutions that meet the hardware requirements of the edge. In essence, the aim of this article is to contribute to the community with empirical data to demonstrate the opportunities that the optimization of the pipeline can provide to (i) match the available resources of edge devices for a balanced cost-accuracy trade-off and; (ii) to be able to integrate both inference and training at the edge instead of the cloud, avoiding extra threats on compromising private data.

The rest of the paper is organized as follows. Section 2 outlines the literature on embedded machine learning and edge intelligence. Section 3 describes the different variables and factors that take part in the classification pipeline. Section 4 explains the procedure and the experimental setup followed to measure the performance of the activity recognition system. The obtained results, in terms of classification efficacy, accuracy and their trade-off, are presented in Section 5 and discussed in Section 6. Finally, Section 7 draws some conclusions and future work.

2. Related Work

The idea of combining edge computing and artificial intelligence is an emerging research area. It aims to make intelligent applications less dependent on the centralized cloud and bring them closer to the users [18]. This transition from the cloud to the edge requires to bring complex computation, including machine learning algorithms, to embedded devices [19]. Nonetheless, despite the possibilities offered by modern hardware platforms, the limitations in the computational capabilities of edge devices hinder the execution of machine learning techniques on them [20]. To tackle this open challenge, early research studies mainly focused on verifying the feasibility of edge devices to run machine learning applications [21], and estimating the theoretical computational cost of classification algorithms [22]. In the former case, Lane et al. identified execution bottlenecks and provided some initial insights about how models could be scaled down. In the latter, Jensen et al. referred to this characterization as the accuracy–cost conflict, in which embedded classification systems must not only consider the classification rate but also be able to balance the cost of each classifier.

Similarly, other investigations have approached embedded intelligence evaluating the suitability of the Raspberry Pi platform to implement Machine Learning inference tasks on the edge [23]. In the same line, Desraches et al. [24] considered the simplification of the classification problems to shed light on the feasibility of embedding simple yet accurate probabilistic models on constrained devices. In fact, machine learning frameworks are being adapted to this new edge scenario. For instance, Tensorflow [25] has incorporated a variant that improves on-device inference and provides tools to include pretrained AI models in resource-constrained devices.

However, finding the optimal balance between the cost and the accuracy implies identifying how the available resources could be used more effectively [26]. In this line, what is needed is the development of techniques that optimize the execution of the machine learning tasks. The existing literature follows two main approaches to reduce the gap between the computation complexity and the available resource capacity: (i) implementing more efficient and lightweight representations of the learning methods and (ii) optimizing different parts of the classification pipeline to lower the amount of processed data, while reducing the complexity of the classification systems and the final models.

The first approach corresponds to adapting classifiers to the capabilities of the target hardware or creating lightweight implementations of the machine learning algorithms [27]. The different learning methods differ from each other in terms of computation requirements, memory size, or accuracy properties. For this reason, Alam et al. [28] analyzed the applicability of some of the most common machine learning models for the IoT. They concluded that some traditional machine learning algorithms, such as Naive Bayes, Support Vector Machines, Linear Regression, and Decision Trees, usually generate a relatively low footprint over resources. For this reason, several approaches have been proposed for those classifiers; for instance, optimizing Support Vector Machines [29], k-Nearest Neighbor [30] and implementing new Decision Trees-based algorithms for efficient prediction on IoT devices [31]. Various works have also dealt with the optimization of deep learning models and the ability to deploy of artificial neural networks on tiny de-

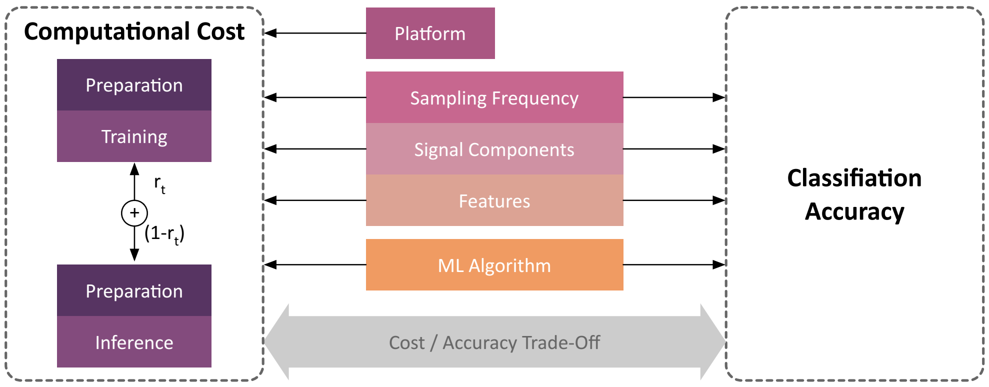

Figure 1: Schematic representation of the different factors affecting the cost-accuracy trade-off.

vices [21, 32]. In this regard, Scheidegger et al. [33] analyzed how deep neural networks can run efficiently on IoT devices and proposed an automatic way to design deep learning models that satisfy user-defined constraints. Moreover, Wang et al. [34] optimized the performance of multilayer perceptron methods on low power embedded devices.

The second approach comprises diverse optimization strategies that, in general terms, seek to save further resources on the different stages of the transformation of data into knowledge [35, 36]. To do that, those strategies address the resource overhead by reducing the data input size or the final model complexity, aiming to achieve a similar classification accuracy. In this respect, feature selection can reduce the cost associated with the extraction of features set [37]. Since smaller number of features could lead to a lower classification accuracy, it is essential to profile this trade-off. This has already been addressed for HAR systems in embedded and wearable devices by Elsts et al. [38] and Zalewski et al. [39]. The sampling frequency is another aspect that deserves attention to improve the final performance of edge intelligent systems [40]. Together with saving resources at the edge, decreasing the sampling frequency can also be potentially interesting to reduce end-device energy consumption as well as communication bandwidth. Kraemer et al. [41] evaluated the effects of the sampling intervals on both energy consumption and accuracy, in order to save energy and costs on the network infrastructure. Similarly, Yan et al. [42] proposed an adaptive approach for mobile devices and Qi et al. [43] presented a novel algorithm to determine the optimal sampling rates according to the requirements of a multi-activity classification and a single activity event detection. In the same way, the number of sensors in a multi-sensor setting can be optimized to improve the trade-off between them and/or signal components, and the classification accuracy. In this line, Kang et al. [44] proposed a dynamic selection of a small subset of sensors. Gordon et al. [45] evaluated the importance of every sensor and compared the trade-off between the number of sensors and the detection accuracy.

Another possibility to simplify the complexity of classification systems is reducing the model size. This effort is geared towards improving the memory impact and computational workload of over-parameterized deep learning models [35]. This optimization is achieved by means of changing the data representation strategy to reduce the number of bits that are used to represent a number, or removing the least essential parameters of the final model [46]. Model compression [47] and pruning techniques [48] have proved to be suitable tools to simplify neural network topologies with a small impact on the accuracy of the classification.

Despite the promising advances reviewed through this section, the research on this topic still focuses on the inference at the edge, relying on centralized servers or distributed approaches for training the models [49]. Model training is often the most time-consuming part of the model development process. Thus, training stages at the edge are very limited by the performance of embedded devices. This limitation makes this specific part of the classification pipeline a key area of focus to bring all the stages of the machine learning process to the local stage. Whilst initial attempts to evaluate the suitability of embedded platforms have been conducted [50, 51], bringing the learning phase closer to the edge and optimizing its performance continues being understudied on the body of the literature. Hence, with this work, we aim to contribute to this open challenge by improving, not only machine learning inference at the edge, but also the training efficiency. In contrast to most of the reviewed works, we present a holistic strategy, i.e., the implemented system is globally analyzed, taking into account the contribution of each part towards the final pipeline.

3. The Trade-off Between Cost and Accuracy

Detecting human behaviour through the classification of sensor data involves an ensemble of computationally intensive tasks. In the following, we want to better understand the computational cost of each of them and how these are connected to the resulting accuracy of the classification results. For that, we have identified five factors that affect the computational cost: (i) the platform on which the classification is performed and the setup, (ii) the sampling frequency in which raw data is captured, (iii) the number of components of the signal (for accelerometer data), or the number of sensors that can be involved, (iv) the number of features extracted from the data, and, finally, (v) the machine learning algorithm used. These factors are depicted in the center of Figure 1.

When calculating the cost, we should differentiate between two main stages in the machine learning process. On the one hand, creating a new model (training stage, also known as model fitting), and, on the other hand, deploying this trained model to make predictions (inference stage). Both stages, as can be observed in Figure 1, involve data pre-processing (preparation), which corresponds to analyzing the raw data and transform them into useful inputs for machine learning algorithms. Such preparation usually relies on features calculated from the processed data that represent the main characteristics of the initial data.

As explained in the introduction section, we foresee to also execute the training stage at the edge and not only inference. In this way, training also contributes to the sum of the total computational cost we will calculate for the whole pipeline. Depending on the machine learning technique, the training cost for a model can be several orders of magnitude higher than its inference associated cost. However, it is true that training new models is usually done much less frequent than inference. For example, a HAR system could infer movements from data several times per minute, while retraining only happens every few days. We capture this by the training ratio r_t; $0 < r_t < 1$, so that the total computational cost c_{total} is calculated as follows:

$$c_{total} = r_t * c_{training} + (1 - r_t) * c_{inference}$$

. We will later study the influence of this ratio on the feasibility of different solutions.

Apart from affecting the computational cost, all factors of Figure 1, except the platform, also influence the accuracy of the classification. Thus, all these factors condition the reliability of machine learning algorithms when it comes to detecting human behavior from the sensor data. This interdependence implies a critical trade-off in the design and operation of the proposed edge system. After the choice of the platform, which is done during deployment and determines the available hardware, the other parameters can be adjusted during run-time. These factors provide the possibility to adjust computational cost and accuracy, which will be studied in the remainder of the article.

4. Procedure and Methodology

To evaluate the performance of several machine learning algorithms, we propose a HAR classification problem where sequences of accelerometer data are used to relate the inertial signals to previously labeled actions. In particular, in this study, we focus on the methodology to design a drinking activity detection system, even though this optimization process can be extrapolated to any classification problem. To this end, a standalone edge architecture is proposed, where the edge devices are in charge of both retraining the initial model as well as inferring new knowledge. In this section, we cover the procedure and methodology followed to understand the impact of the different factors involved in classification problems.

4.1. Initial data

From the existing human activity datasets, we selected one that includes the drinking activity, namely, the ADL Recognition with Wrist-worn Accelerometer Data Set [52]. It is publicly available in the UCI Machine Learning Repository [53]. This dataset contains 14 labeled activities of daily living in 839 recorded trials. These activities were recorded by 16 volunteer participants (11 men and 5 women with ages between 19 and 81 years) wearing a triaxial accelerometer on their right wrists with an output rate of 32 Hz. The accelerometer measurements are provided as a series of data points indexed in time order, so that activities are classified based on the signal behavior over a period of time. As such, this is a time series classification problem.

The selected dataset was binarized to represent two classes: drink activity (100 instances) and the rest of the activities (739 instances). Even though this might create an unbalanced distribution of classes, the available data still provide a suitable framework to perform the comparative analysis of the classification solution. Thus, to maintain the classification pipeline as standard as possible for evaluation purposes, no specific balancing method was applied to the data. Note that performing any balancing method (under-sampling or over-sampling strategies) would have an impact on the computational cost, which should also be considered when balancing the cost-accuracy trade-off. Therefore, we did not introduce such a phase in our proposal.

4.2. Data pre-processing

Within the preparation phase, we applied a 3-point median filter to smooth the signal. This avoids big spikes and anomaly values induced by noise. This is a simple yet effective method to reduce high-frequency noise that usually is combined with low-pass filters [54]. In this case, after checking that there were no significant improvements in classification results, no frequency-domain filter was implemented to reduce the computational complexity. Thereafter, every filtered sequence of data goes through a segmentation process. The entire window of the data sequence is divided into five segments (or sub-windows) of equal length without overlapping.

Algorithms usually rely on features calculated from raw data for classification. This study primarily focuses on time-domain statistical features that exhibit better cost-benefit properties than frequency-domain features [55]. The selected set of features includes mean, min, max, standard deviation, median, kurtosis, skewness, variance and mean absolute deviation. This selection represents the most commonly used features for activity recognition problems according to the literature [56]. These nine statistical features are calculated for every component of the signal (X, Y, and Z), for each of the five segments in which the signal is divided and also for the entire sequence. This sums a total of 162 initial features that are vectorized to characterize every sequence of the dataset. Then the features are scaled into a [0-1] interval using max-min normalization.

4.3. Feature selection

Feature selection is a discriminating process to find important features that have more weight in the model. It consists in reducing the dimension of the feature matrix removing the irrelevant features and obtaining the subset of them that contributes the most on the prediction. Thus, this stage of feature selection is essential when simplification is needed to optimize the classification process and reduce training and inference time. The feature selection strategy can be chosen based on considerations such as simplicity, stability or classification accuracy and it may entail substantial differences on the final classification results [57]. In this work, we apply a Chi2 filtering method that evaluates the correlation between variables (features and target classes) and ranks them according to their contribution to the prediction [58]. This is a computationally light strategy that provides a good balance between its potential results and its simplicity [59].

Table 1: The most representative features and their correlation score values indicating the strongest dependency.

Position	Feature	Score
1	Min X 3	86.121
2	Min X	80.993
3	Mean X 3	66.046
4	Median X 3	62.846
5	Min X 2	44.452
6	Mean X 2	36.978
7	Mean X	36.236
8	Median X 2	34.839
9	Median X	32.655
10	Mean X 4	29.103

The 10 most representative features are listed in Table 1. Every feature is named by its statistical property, its accelerometer component and the segment of the time series sequence to which the feature corresponds. As observed, in this feature selection method, the top-10 features were captured from acceleration data in X, that proved to be the most representative one for the drinking activity. The higher scores of the top-ranked features indicate how relevant this short subset of characteristics is in the classification results. This makes it feasible to reduce the number of features without an entailed substantial loss of accuracy on the detection.

We obtained this reduced subset of characteristics applying the feature selection process to the whole dataset. However, to evaluate the accuracy of algorithms, the most representative features for each model are repeatedly calculated within the validation process using only the training data.

4.4. Experimental setup

According to Dhar et al. [26], understanding how machine learning algorithms can contribute to edge intelligence is a crucial challenge for on-device training. Traditional methods are especially interesting for building edge learning capabilities, particularly when the computation power is low and the memory is limited. Hence, there is a need to explore the traditional machine learning approaches for implementing on-device training. For this reason, this study focused mostly on various supervised machine learning methods that are implemented for the classification problem: Logistic Regression (LG), Random Forest (RF), K-nearest Neighbors (KNN), Naive Bayes (NB) - Gaussian, Linear Support Vector Machine (SVM), Multilayer perceptron (MLP), and Decision Trees (DT).

In terms of hardware, we evaluated three different platforms. A laptop computer works as a baseline device for comparison. Its specifications include an Intel Core i7-975H processor with a base frequency of 2.60 GHz, equipped with 16 GB of RAM. For the IoT devices, two Raspberry Pi Foundation low-cost single board minicomputers were chosen: the Raspberry Pi 3 model B+ and the Raspberry Zero W. The former (that has higher computational capabilities) incorporates a quad-core A53 (ARMv8) 64-bit-based System on Chip (SoC) at 1.4GHz and 1 Gb of RAM. The latter (which is more limited in terms of hardware) includes a single-core ARM11 (ARMv6) 32-bits-based SoC at 1 GHz and 512 Mb of RAM. Even though both devices belong to the same family, judging by their characteristics and performance (in terms of processing power and memory), there are significant differences between them. In fact, taken into account that both devices are compatible with the same software tools and, thus, they are evaluated under the same conditions, the paper provides an equitable comparison between two distinct enough devices.

For the sake of fairness, all experiments are executed solely on CPU with no other application running at the same time. The classification solution software relies on the Python library Scikit-learn [60], which includes efficient versions of the most common algorithms. The experimental process includes the training and the inference phase, as well as the validation of the models for each of the selected machine learning algorithms. The materials used for the evaluation experiments are publicly available [1].

5. Analysis and Results

The results presented in this section can be divided into three different categories: the evaluation of the classification accuracy, the computational cost of machine learning techniques (measured as the necessary time to perform the classification process), and the trade-off between them. In both cases, the impact of the described factors (sampling frequency, number of signal components, number of features and the algorithm) are measured for each of the selected hardware platforms. Finally, in this section we present a selection of cost-accuracy optimal solutions obtained through the process of data analysis and reduction combined with a multi-objective optimization strategy.

5.1. Classification accuracy

This subsection evaluates the accuracy of the machine learning models according to the validation procedure explained in Figure 2. We perform this evaluation through a 5-fold-cross-validation procedure. Such validation was repeated 100 times to obtain more robust results. Four metrics are used to measure the predictive performance of algorithms: *Precision*, which measures the true positives among all positive results; *Recall*, which computes correctly labeled positives based on all the correct positive and negative events; *F1 Score*, the harmonic mean of

[1] `https://github.com/OihaneGomez/Exploring_Computational_Cost_ML_IoT`

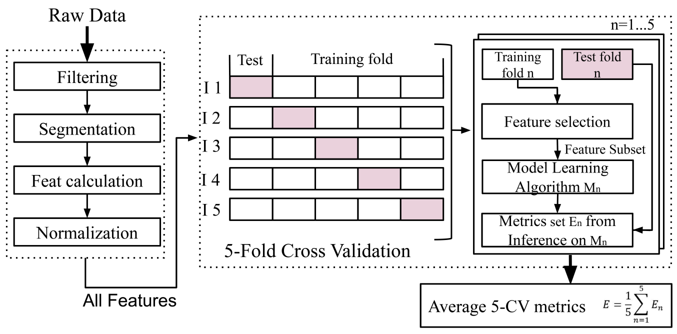

Figure 2: Schematic representation of the model training phase and its evaluation through a 5 cross-validation process.

precision and recall metrics; and *Accuracy*, the ratio of correct prediction over the total number of predictions. Macro average results are provided for the first three metrics.

The first step in the optimization process is to quantify the influence of reducing the data input on the classification capabilities. Thus, we consider the three factors upon which the accuracy depends on and we evaluate them for each of the machine learning algorithms. The influence of the sampling frequency is analyzed by downsampling the original data to lower frequencies. For the number of components, we compare the performance of the models that use all the accelerometer signal components (XYZ) with models that only receive the most representative one (X). This component was selected according to the ranking of features (see Table 1) that revealed X as the most representative component. Finally, we analyze the effects of the number of features on the classification output comparing the results for all the original features set (162), a subset of the top 10, and the 3 top-ranked features.

Table 2: Average F1 results for several sampling frequencies.

	All Features	10 Features	3 Features
32 Hz (Initial)	95,60 (SD=1,86)	94.83 (SD=3,72)	92.55 (SD=4.06)
16 Hz	95.60 (SD = 1.28)	94.07 (SD=2,24)	88.66 (SD=2,90)
8 Hz	94.91 (SD =1.84)	93.52 (SD=2,32)	88,31 (SD=2.54)
4 Hz	94,87 (SD=1,73)	93,52 (SD=1.90)	86,45 (SD=3,48)

With regard to the sampling frequency, Table 2 resumes the average F1 results for the sum of all the learning methods. This tables compared initial results with the ones obtained by reducing the original 32 Hz sampling rate to 16 Hz, 8 Hz and 4 Hz. It can be noted how lower frequencies decrease the classification performance but still maintain acceptable detection rates. In this particular application, this means that it is feasible to lower the frequency up to 4Hz. However, these results could be biased for the stationary nature of the target class (drinking activity) and may not be extrapolated to standard classification problems. For this reason, we have decided to cut off this reduction and keep 16 Hz as the reference downsampled ratio, since this frequency belongs to the 15 Hz–20 Hz range, which is considered as appropriate for HAR systems [61]. Therefore, in Tables 3 and 4 we summarize the performance of all the analyzed supervised machine learning approaches for the original sampling data (32 Hz) and for the downsampled data (16 Hz). In both cases, the results for XYZ and X (the most representative component) are included. As can be observed, the classification rates yield average values over 90 % in almost all cases.

Figure 3 illustrates the influence of the number of features on the accuracy for the regular sampling. In particular, this figure highlights how the detection rates increase quickly at the beginning of the graph and reach values over 92 % with the first 20 features. This indicates that a good balance between the number of characteristics and the accuracy of the classification can be obtained with few features irrespective of the classification method. As can be observed, adding more features to the models can improve the performance in some cases while decreasing it in others, since the feature selection process is independent of the algorithm.

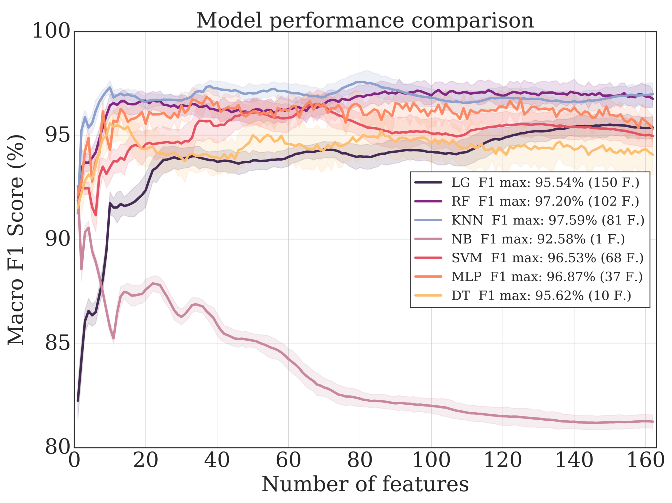

Figure 3: Model performance evolution, measured by the F1-Score, according to the selected number of features.

A thorough analysis shows that K-nearest neighbor performs slightly better than the other algorithms when the classification depends on all the 162 features. Naive Bayes scores are worse for a larger number of features than for a small number of them. This can indicate that the features are not independent enough by themselves, something opposite to the intrinsic modelling assumption of Naive Bayes, which assumes that the presence of a specific feature is completely unrelated to the presence of any other [62]. For this reason, we still include it for the analysis of the cost-accuracy trade-off, but we consider it an outlier for average calculations of Tables 5 and 6.

In Table 5 and Table 6, we show the final impact of every of the evaluated factors compared to the baseline performance (original sampling frequency, XYZ signal components, and 162 features). The results indicate small differences when decreasing the sampling frequency to 16 Hz. Moreover, the classi-

Table 3: Performance of the supervised machine learning algorithms for the original sampling frequency (32Hz).

	All Components (XYZ)											
	All 162 Features				10 Features				3 Features			
	Recall	Precision	F1	Accuracy	Recall	Precision	F1	Accuracy	Recall	Precision	F1	Accuracy
LG	94.73	96.36	95.40	98.10	90.20	93.95	91.75	96.70	82.87	91.03	86.09	94.77
RF	95.66	98.14	96.78	98.70	96.19	96.89	96.46	98.53	93.18	94.54	93.72	97.41
KNN	97.78	96.40	**97.01**	98.72	98.07	96.70	**97.32**	98.85	95.61	96.33	**95.89**	98.29
NB	92.91	76.58	81.26	89.38	94.89	81.00	85.77	92.51	96.22	86.48	90.36	95.33
SVM	95.51	94.67	94.99	97.87	92.99	94.33	93.49	97.30	92.10	93.11	92.47	96.88
MLP	95.40	95.48	95.34	98.04	94.45	94.43	94.31	97.60	94.57	94.17	94.26	97.56
DT	94.62	93.89	94.10	97.48	95.31	96.15	95.62	98.18	93.19	92.82	92.84	96.97
	Most Representative Components (X)											
	All 54 Features				10 Features				3 Features			
	Recall	Precision	F1	Accuracy	Recall	Precision	F1	Accuracy	Recall	Precision	F1	Accuracy
LG	93.66	95.07	94.22	97.62	90.20	93.95	91.75	96.70	82.87	91.03	86.09	94.77
RF	96.05	97.70	96.78	98.68	96.19	96.89	96.46	98.53	93.18	94.54	93.72	97.41
KNN	97.33	96.85	**97.02**	98.74	98.07	96.70	**97.32**	98.85	95.61	96.33	**95.89**	98.29
NB	94.98	83.85	88.05	94.03	94.89	81.00	85.77	92.51	96.22	86.48	90.36	95.33
SVM	95.35	94.73	94.94	97.86	92.99	94.33	93.49	97.30	92.10	93.11	92.47	96.88
MLP	95.43	95.17	95.20	97.97	94.45	94.43	94.31	97.60	94.57	94.17	94.26	97.56
DT	94.74	94.84	94.68	97.76	95.31	96.15	95.62	98.18	93.19	92.82	92.84	96.97

Table 4: Performance of the supervised machine learning algorithms for half the sampling frequency (16 Hz).

	All Components (XYZ)											
	All 162 Features				10 Features				3 Features			
	Recall	Precision	F1	Accuracy	Recall	Precision	F1	Accuracy	Recall	Precision	F1	Accuracy
LG	94.95	96.47	95.57	98.18	88.28	93.47	90.43	96.25	79.90	89.00	83.32	93.84
RF	95.84	97.89	96.76	98.68	95.71	96.24	95.89	98.29	88.75	90.77	89.53	95.72
KNN	97.77	95.99	**96.80**	98.62	97.35	96.48	**96.85**	98.66	90.02	89.91	89.78	95.69
NB	92.98	76.72	81.42	89.50	94.46	79.81	84.63	91.76	94.94	82.63	87.13	93.45
SVM	96.16	95.02	95.48	98.07	92.23	94.87	93.34	97.29	91.42	90.55	90.80	96.08
MLP	95.65	95.97	95.71	98.20	94.23	94.68	94.32	97.62	92.69	89.79	**91.03**	96.08
DT	93.49	93.40	93.29	97.17	93.39	94.03	93.56	97.31	87.48	87.95	87.48	94.76
	Most Representative Components (X)											
	All 54 Features				10 Features				3 Features			
	Recall	Precision	F1	Accuracy	Recall	Precision	F1	Accuracy	Recall	Precision	F1	Accuracy
LG	93.66	95.46	94.42	97.71	88.28	93.47	90.43	96.25	79.90	89.00	83.32	93.84
RF	95.82	97.45	96.54	98.58	95.71	96.24	95.89	98.29	88.75	90.77	89.53	95.72
KNN	97.58	96.45	**96.94**	98.69	97.35	96.48	**96.85**	98.66	90.02	89.91	89.78	95.69
NB	95.11	84.37	88.46	94.27	94.46	79.81	84.63	91.76	94.94	82.63	87.13	93.45
SVM	93.68	93.26	93.33	97.18	92.23	94.87	93.34	97.29	91.42	90.55	90.80	96.08
MLP	94.29	94.68	94.35	97.64	94.23	94.68	94.32	97.62	92.69	89.79	**91.03**	96.08
DT	93.41	94.07	93.58	97.33	93.39	94.03	93.56	97.31	87.48	87.95	87.48	94.76

Table 5: Average F1 results comparison for the original sampling (32 Hz). XYZ and the most representative component (X) are included.

	All Features	10 Features	3 Features	Decrease
XYZ	95,60	94,83	92,55	3,199 %
X	95,47	94,83	92,55	3,058 %
Decrease	0,135 %	0 %	0 %	

Table 6: Average F1 results comparison for half the original frequency (16 Hz). XYZ and the most representative component (X) are included.

	All Features	10 Features	3 Features	Decrease
XYZ	95,60	94,07	88,66	7,259%
X	94,86	94,07	88,66	6,535 %
Decrease	0,774 %	0 %	0 %	

fication results are not drastically penalized when the number of features is reduced to 10 and even 3. However, decreasing the number of features has a more significant impact on lower sampling frequencies. For the original sampling provided by the dataset, using only the top 3 features penalized the results in a 3.19 %. For half of the sampling rate, the decreasing was 7,259 %. Furthermore, considering only the most representative component of the signal (X) does not affect the overall results significantly: 0.135 % for the regular sampling and 0.774 % for the downsampled data. The results are the same when considering only the top 10 and top 3 features in classification. This means that only X-related features are in this top.

5.2. *Computational cost*

In this subsection, we evaluate how the design parameters affect to the model in its two stages: training and inference. Figure 4 represent the experimental sequence in both cases. We measure the computational cost of each process as the necessary time to accomplish all the tasks it encompasses.

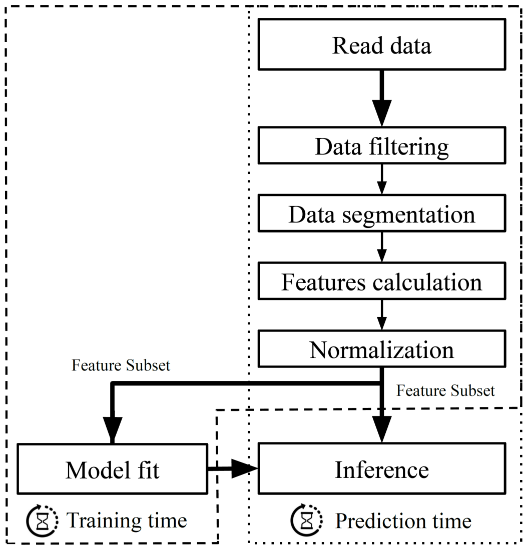

Figure 4: Representation of the training and inference phases.

5.2.1. Training process

Training a machine learning model corresponds to one of the most computationally intensive processes within an activity recognition application. The main reason is that it involves processing large amounts of data. The training process can be divided into two parts, the common pre-processing steps applied to the whole dataset and the algorithm-dependent model fitting task. Table 7 presents the elapsed time for the initial preprocessing stage when all the dataset information is read, and the features are calculated. These results were obtained after repeating the process at least 15 times (3 runs of 5 loops) and calculating the average running times. They are also displayed in Fig. 5, that compares the evolution of the processing time according to the number of features.

Table 7: Computational cost of processing all the dataset for the training process using the original sampling frequency (32 Hz).

	XYZ			X		
	Laptop	Rpi	Rzero	Laptop	Rpi	Rzero
All Feat.	16s	2min 5s	12min 32s	5.3s	52s	5min 5s
10 Feat.	7,1s	43,6s	4min 6s	3.1s	25s	2min 18s
3 Feat.	7s	41,4s	3min 56s	3.0s	22,5s	2min 9s
Decrease	56,32 %	66,94 %	68,61 %	44,05 %	56,78 %	57,73 %

At first sight, it can be noted how the computational time shows a quadratic growth when the data input (the number of features) increases. This is particularly relevant for the scalability of the classification system, and may heavily penalize the performance of less powerful devices. For this reason, a more in-depth analysis of the percentage reduction shows that more significant performance improvements are obtained in those platforms. As expected, the laptop computer outperformed the other devices in absolute terms. In relative terms, the laptop decreased its running times in a 56,32 % for XYZ, while the

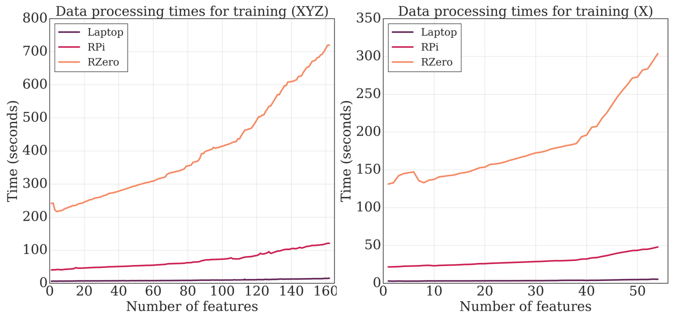

Figure 5: Evolution of the elapsed time for processing all the instances of the dataset for training.

same data reduction achieved a decrease of 68,61 % on the execution times for the Raspberry Zero platform. Moreover, from the baseline parameters (all features and all the signal components) to the final simplified stage (3 features and only the most representative component) the time reduction was 81,24 % for the laptop (from 16 seconds to 3 seconds), 82,05 % for the Raspberry Pi (from 2 min and 5 seconds to 22,5 seconds) and 92,32 % for the Raspberry Zero (from 12 min and 32 seconds to 2 min and 9 seconds).

Table 8: Computational cost of processing all the dataset for the training process and half the original sampling frequency (16 Hz).

	XYZ			X		
	Laptop	Rpi	Rzero	Laptop	Rpi	Rzero
All Feat.	15.98s	2min 5s	12min 19s	5.38s	49.34s	5min 10s
10 Feat.	7.72s	42.92s	4min 5s	3.13s	24.4 4s	2min 18s
3 Feat.	7.43s	40.79s	3min 44s	3.01s	22.1s	2min 8s

The effect of the remaining factor, the sampling frequency, is evaluated in Table 8, which includes a comparison of the preprocessing times for 16 Hz. In this case, contrary to what intuitively could be expected, a lower sampling frequency did not substantially affect the time measurements. A more detailed analysis shows that only little differences are found for the largest number of features compared to Table 7. In some cases, because of the non-deterministic nature of the timing measurements, the performance was even slightly worse. This lack of improvement can be a consequence of the baseline resource overhead associated with data management. Another possible reason is the length of the data window, which is not large enough to have noticeable improvements when it is reduced. Therefore, in our evaluated case of study, this may indicate that the number of computations (e.g, how many characteristics of the signal are obtained) has a greater impact on the pre-processing stage than the effect of the data length. However, this particular conclusion cannot be generalized to similar applications. Thus, it would be necessary to look at the effect of the sampling frequency in applications with larger data sets, longer data streams, more complex filtering stages or frequency-domain features. In the following, we will continue the analysis using only the original sampling frequency (32 Hz), since the

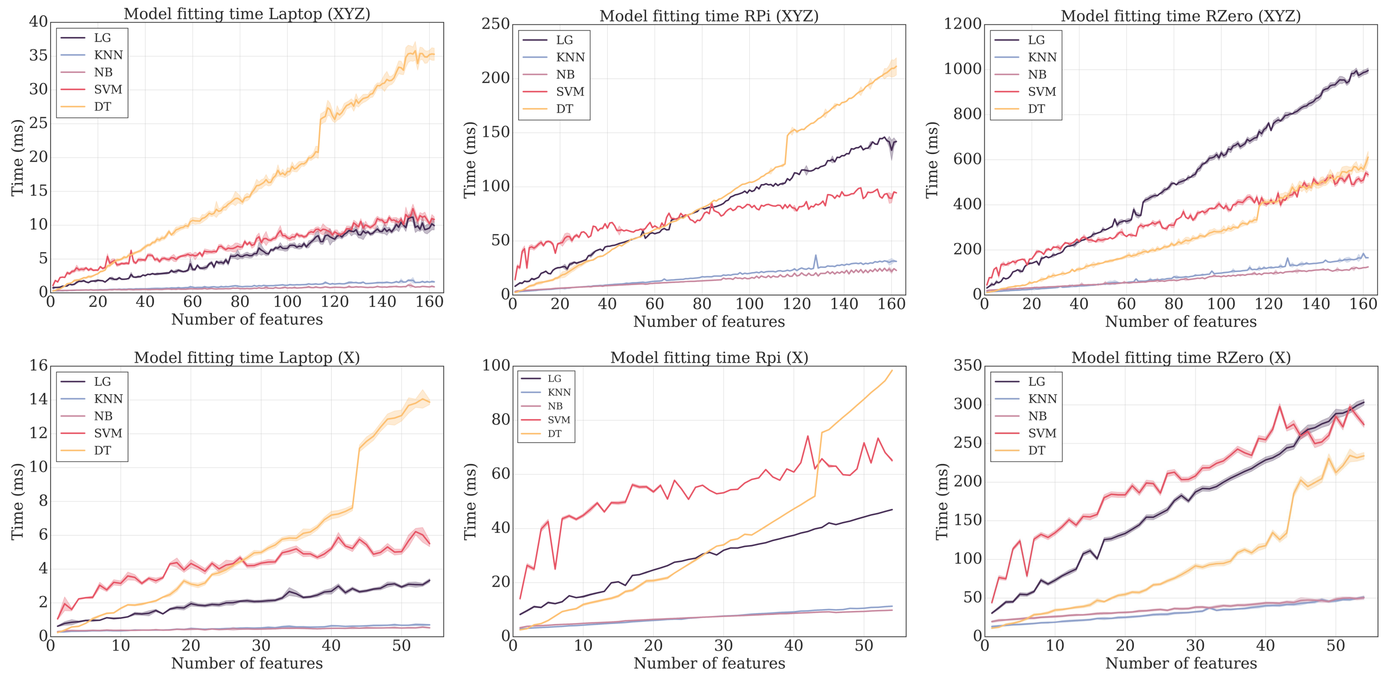

Figure 6: Comparative between the number of features and the elapsed time for fitting the model. MLP and RF are out of this representation since they have a significantly greater duration than those included.

model created with the downsampled data is more sensitive to decreasing the number of features.

The second step of the training stage is to feed the model with the pre-processed data for the fitting task. This task depends on the different classifiers and their parameters, together with the amount of data used to build it. Figure 6 represents the average computational cost of each algorithm for the model fitting task after executing it 1000 times (100 runs of 10 loops). Furthermore, it shows the time impact caused by creating the model with an increasing number of features. Random forest and Multilayer Perceptron techniques are excluded from these plots since their duration, significantly greater than those represented, compromise the visibility of the figure. Instead, their results are presented in Table 9 and Table 10.

Table 9: Elapsed time fitting the model for MLP and RF considering XYZ.

	Laptop			Raspberry Pi			Raspberry Zero		
Features	162	10	3	162	10	3	162	10	3
RF	181.9ms	96.4ms	80.5ms	1,6s	1.1s	970ms	7.8s	6.2s	6.1s
MLP	782.3ms	456.9ms	622.2ms	3,2s	2.3s	2.4s	20.2s	14.9s	17.6s

Table 10: Elapsed time fitting the model for MLP and RF considering X.

	Laptop			Raspberry Pi			Raspberry Zero		
Features	54	10	3	54	10	3	54	10	3
RF	136.0ms	96.3ms	80.6ms	1.3s	1.0s	991.3ms	6.8s	6.5s	6.3s
MLP	719.2ms	452.1ms	624.3ms	2.2s	2.5s	2.4s	13.2s	14.3s	15.7s

For MLP, increasing the number of features can affect positively on the timing results. In this case, from 3 to 54 features, the efficiency of the fitting task was increased. As the MLP is iteratively trained, depending on how fast it converges, the time to fit the data can vary. Consequently, a larger data input size may then improve the fitting process of the model.

The main conclusion that can be extracted from those results is that fitting task times are significantly lower than the time needed to process all the initial data and create the input for the model. Contrary to the pre-processing stage, here the algorithm and its internal parameters play an important role when creating the model. In general terms, the fitting task shows a behavior close to linear when increasing the input size. This involves a baseline computational complexity of $O(nm)$, being m is the number of instances and n the number of features. For this reason, with a small number of features, all the algorithms last similarly, but when increasing the number of features, the computational cost is augmented. On top of that, the increments in time (the slope) of the each of the learning algorithms depend on the complexity of the learning method itself or the different parameters and estimators that can be adjusted on the fitting phase. For this reason, the complexity of the machine learning techniques can be reduced by modifying those fitting parameters to simplify the model, but decreasing its accuracy in return. Still, considerable improvements in the performance can be obtained when adjusting the internal values that affect data model. This is specifically the case for RF and MLP, which are the most resource-demanding algorithms in this comparative.

These slopes are also affected by the platform on which this training process is carried out. In general, under 20 features the times of all the algorithms are within the first segment of the Y-axis for the three platforms, being 200 ms the worst case for the Raspberry Zero W.

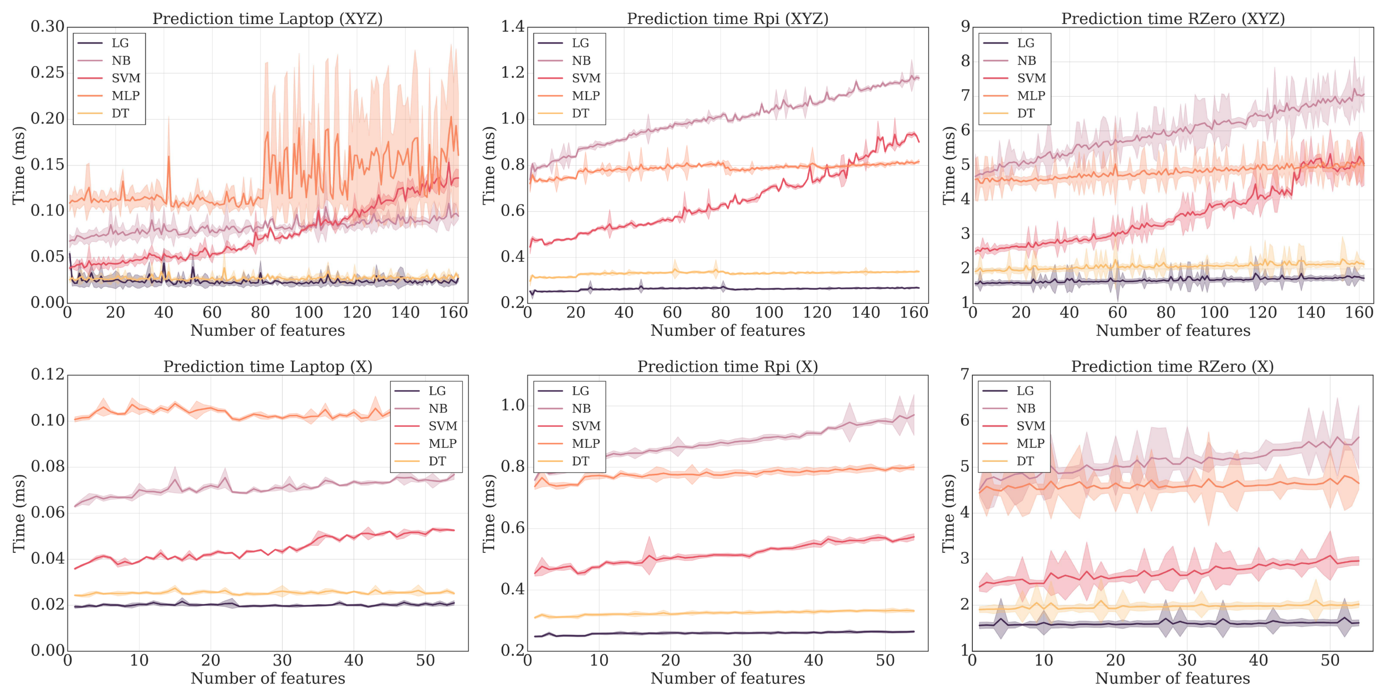

Figure 7: Comparison between the number of features and the elapsed time for predicting a sequence of new data. KNN and RF are out of this representation since they have a significantly greater duration than those included.

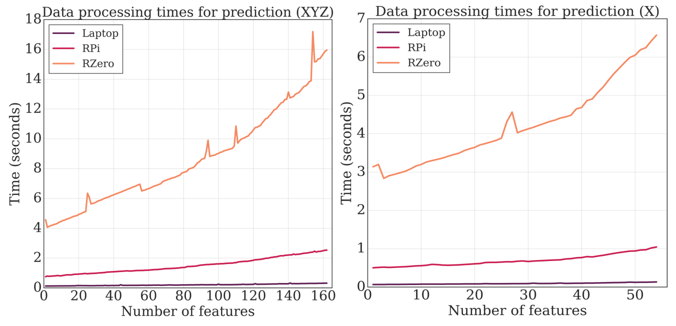

Figure 8: Evolution of the computational cost of processing 14 instances of the dataset for inference.

Table 11: Time needed to process 14 instances of new data for the prediction process.

	XYZ			Most representative component		
	Laptop	Rpi	Rzero	Laptop	Rpi	Rzero
All Features	317 ms	2.62 s	15.9 s	125 ms	1.07 s	6.57 s
10 Features	140 ms	838 ms	5.48 s	71.2 ms	557 ms	3.61 s
3 Features	134 ms	781 ms	5.06 s	62.5 ms	500 ms	3.24 s
Decrease	58,04%	70,20%	68,25%	49,19%	53,14%	50,65%

Table 12: Prediction times for RF and KNN considering XYZ.

	Laptop			Raspberry Pi			Raspberry Zero		
Feat.	162	10	3	54	10	3	54	10	3
RF	5.6ms	5.5ms	5.5ms	46.1ms	39.9ms	39.4ms	261.5ms	260.9 ms	260.7ms
KNN	2.7ms	0.7ms	0.6ms	23.9ms	3.7ms	2.6ms	100.4ms	15.9 ms	13.4ms

Table 13: Prediction times for RF and KNN considering X.

	Laptop			Raspberry Pi			Raspberry Zero		
Feat.	54	10	3	54	10	3	54	10	3
RF	5.6ms	5.6ms	5.5ms	39.6ms	39.0ms	38.7ms	279.8ms	279.0ms	278.8ms
KNN	1.2ms	0.7ms	0.6ms	7.8ms	4.1ms	2.6ms	41.3ms	17.2ms	14.5ms

5.2.2. *Prediction process*

The prediction process is tested to evaluate how long it takes to read and process a sequence of data and compare it with the already trained model. The experiment consists of measuring the elapsed time in the process of classifying a sample of 14 instances of the dataset. Therefore, this test resembles an online classification system where a new sequence of data needs to be predicted without an entailed excessive processing time. Again, the results are divided into two parts, the common process of reading and processing the new data and the inference task.

Table 11 shows the computational cost of reading and processing the 14 instances of new data for a variable number of features, computed after executing the experiment 30 times (10 runs of 3 loops). Again, it illustrates how processing times benefit from the feature reduction step and the selection of the most representative components of the signal. This is especially relevant in the case of the Raspberry Pi and Zero. The total time reduction is 80,28 % for the Laptop PC, 80,93 % for the Raspberry Pi and 97,96 % for the Raspberry Zero. In particular, the Raspberry Zero reduced its total processing times from around 16 seconds to 3 seconds. This improvement is very noticeable and makes a difference when designing online classification systems for low-powered devices. Figure 8 reflects the computational cost tendency that shows again a quadratic growth.

Compared to pre-processing and training, the inference task

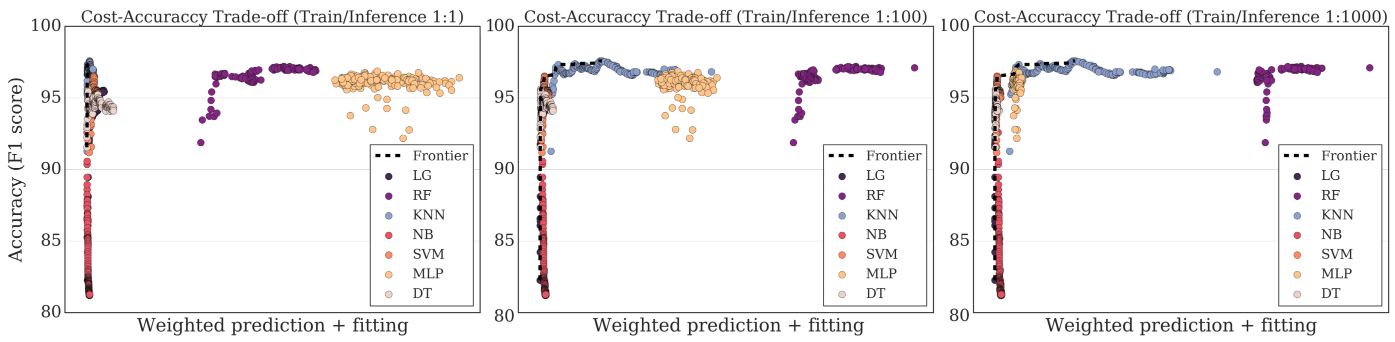

Figure 9: Cost-accuracy trade-off for the balanced computational cost between prediction and fitting times (left) and applying the train inference ratio (center and right). The Pareto frontier represents the most optimal solutions.

is several orders of magnitude faster on all platforms (see Figure 7). Prediction task needs less than 0.3 ms for the laptop device, 1,4 ms for the Raspberry Pi, and less than 8 ms for the Raspberry Zero (mean values of 1000 runs). Random Forest and K-nearest neighbors are out of this representation. Instead, Table 12 and Table 13 include the results of both algorithms.

In this case, reducing the number of features does not significantly affect the time required for inference. Few differences can be found among all algorithms in terms of performance, except the mentioned RF and KNN that took a slightly longer time than the rest of them to compare the new data with the model. Furthermore, Naive and SVM times show a steady increase, while the rest of the algorithms show a sustained response. This is a consequence of the constant computational complexity of the prediction task in some of the evaluated algorithms, which are less sensitive to the data input size. For the prediction phase, the number of features it still is a relevant factor according to the computational cost of pre-processing new data.

5.2.3. *Cost-accuracy trade-off: a Pareto approach*

The correlation between the model size and the accuracy studied along this section lead us to draw an initial conclusion: it is feasible to maintain detection rates over a 90 % when the data input is reduced. In our case, the studied simplification strategy was able to reduce the data input size to a small subset of one representative component and its top-10 features, that outperforms the baseline substantially. Moreover, the time improvements explored in previous subsection produce an enhanced performance in most of the classification tasks when data input size is reduced, regardless of the devices' resources. This allows bringing more complex applications to a local stage that could not be executed in the edge otherwise. Based on the obtained results it is expected that, in some applications, optimizing early stages of the pipeline can improve the performance of the system more than the specific learning method.

From that point, further evaluation can optimize the trade-off between classification accuracy and computational cost. To that end, we conduct a Pareto multi-objective optimization analysis to find the small-sized subset of solutions with the best performance [63]. The Pareto analysis is a statistical tool for decision-making that selects a limited number of Pareto-optimal solution that illustrates the best trade-off between two objectives, cost and accuracy in our case. Since we seek to evaluate the efficiency of the model for both training and inference stages, we include the role of each of these stages through the train/inference ratio (explained in section 3) in the Pareto analysis.

Figure 9 represents the relationship between the total cost (the sum of the prediction and training times without considering data processing) and the accuracy of the system (F1 values). Every point corresponds to a certain number of features for each of the evaluated algorithms running on the laptop device. This figure also illustrates the effect of this weight correction on the total cost when different train/inference ratios are applied. This ratio is a design choice that should be made according to the context of the classification problem. As explained in Section 3, it gives more importance to the predominant task and sets the frequency in which a model may be retrained in comparison to the estimated number of times new data would be classified. For comparative purposes, we have selected a weight correction of r=1 (baseline ratio, no weight correction is applied), r=100 and r=1000 to show the differences. This means that the training stage may occur once every 100 or 1000 times or, in other words, giving 100 or 1000 times more importance to the prediction stage over the training phase. As we explained in initial sections, the weighted computational cost c_{total} is calculated as: $c_{total} = r_t * c_{training} + (1 - r_t)c_{inference}$, with $r_t = 1/r$. This figure also includes the Pareto frontier that represents the most optimal solutions and marks the points with better accuracy (highest y-axis values) and lower computational costs (smaller values on the x-axis).

The train/inference ratio is particularly relevant for those algorithms with an unbalance performance between training and inference tasks, such as KNN and MLP. The former has a great performance on training (See Figure 6) while the inference times are worse than the average (Tables 12 and 13). MLP, on the contrary, performs well for inference (Figure 7), but its complexity demands higher times to create the model (Table 9 and Table 10). As the train/inference ratio grows, we can observe how KNN is penalized, increasing its total weighted computational cost, and MLP moves to the left, closer to the Pareto

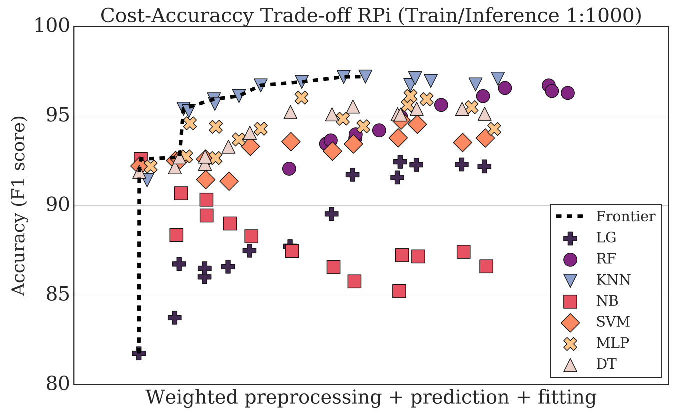

Figure 10: Cost-accuracy trade-off and Pareto optimal points for the reduced data input achieved during the optimization process on the Raspberry Pi 3 device.

Table 14: Final optimal Pareto solutions with the best cost-accuracy trade-off for the Raspberry Pi 3.

Alg	N Features	F1	Total inference time (s)	Total training time (s)
LG	1	81.73	0.503349	21.680870
NB	1	92.58	0.503824	21.676447
SVM	1	92.20	0.503571	21.686039
DT	1	91.88	0.503432	21.675802
KNN	2	95.40	0.514883	21.811192
KNN	3	95.93	0.522636	22.030610
DT	4	92.69	0.513096	22.578788
KNN	6	96.11	0.528342	22.926632
KNN	7	96.70	0.533877	23.088214
KNN	8	96.90	0.544137	23.593250
KNN	9	97.18	0.554919	23.781464
KNN	10	97.19	0.561269	23.153416

frontier. Higher train/inference ratio values would increase this difference in those applications where training times are particularly big compared to the inference time (e.g., when dealing with larger datasets).

Figure 10 shows the Pareto analysis performed only for the final reduced solution (top-10 features and X component) when applying a 1:1000 ratio and for the Raspberry Pi 3 device. In this case, the computational cost included also the pre-processing times for training and inference. Again, the dashed line highlights the Pareto optimal points, where the small set of options that have the most efficient balance between the classification rates and the computational cost are included. Table 14 includes the values associated with each of these points, where total times are the sum of data processing for training and inference together with the fitting or prediction tasks times. The final step in the optimization process is to select the one that better meets the initial design requirements from this selection, either prioritizing accuracy or performance.

5.3. Limitations

Before concluding based on the experimental results, we want to address the possible limitations of this study. The main goal of this work is to analyze the different factors involved in the machine learning pipeline and optimize their influence on the cost-accuracy trade-off. Thus, the significance of this work relies on the comparison of the *relative* results obtained when each of these factors are evaluated. Additional elements affecting the *absolute* results such as the machine learning framework, the programming language, or the hyperparameter optimization are out of the scope of this comparative analysis. For example, the way the different tasks are implemented may have an impact on the execution of the system. For this reason, optimizing general characteristics of the code, such as the way the information is processed (e.g., vectorizing computations over data structures [64]) can improve the general performance of the system. We take this into account in the evaluated solution, but we do not benchmark code optimization mechanisms.

Besides, there are other aspects in the pipeline such as the data filtering method or the feature selection algorithms; the comparison of different strategies for those aspects is out of the scope of this paper. Therefore, we explain the decision criterion to choose them. However, we do not delve into the analysis of the consequences that these decisions may have in the classification results. We note that it would be interesting for future work to evaluate the potential improvements that alternative strategies could entail.

Finally, we evaluated the computational cost in terms of the elapsed time for each of the different stages of the classification problem. Despite running these tasks in isolation, they can be sensitive to internal processes or routines of the device, and hence be non-deterministic. To overcome this issue, we took a large number of measurement samples and provided average results based on a large number of iterations.

6. Discussion

Some intelligent environments, such as smart homes, smart cities or smart workplaces, are especially challenging scenarios where privacy concerns regarding data protection and security are critical issues. In those spaces, the concept of privacy is also related to the ability of the user to determine how, when, and to whom personal information can be disclosed.

In this sense, the edge computing paradigm proposes a concept that brings closer human and machine intelligence and envisages scenarios where positive interactions through privacy-by-design might be created. Data still needs to be captured and processed to create contextual services, but the physical proximity of the users and their data can reduce the critical barriers regarding the acceptance of intelligent environments. From a human being perspective, this ensures that the collected sensitive information remains on personal devices and not on third-party servers, which favors privacy and trust.

Beyond that, creating spaces where human and machine intelligence collaborates is also related to the role of the user, that needs to be carefully considered to reduce their reluctance to be part of this new scenario [65]. More than creating reliable and efficient solutions, such spaces demand an increased control of the users over the system and their data, preventing the threat of intelligent environments taking control over them [66].

For this reason, human-centric approaches should also engage the users to actively collaborate with the machine intelligence [67]. This falls within the concept of human-in-the-loop where approaches like interactive machine learning make possible for end-users to interact with the learning process to tune the system according to their preferences, or even to feed it with more data [68]. In supervised approaches, the user can create personalized human activity recognition systems by providing new annotations [69] while in semi-supervised ones, the model can also be fed with raw time-series data. [70]. Thus, machine intelligence can be adaptively adjusted in terms of human goals. In this scenario, integrating the training stage in the edge can give an answer to the need for privacy of new smart spaces while increasing users' perceived level of trust in the system [71].

The main problem that we have observed throughout this study arises when demanding applications compromise the computational capability of the edge devices. Some scholars have pointed out that pre-trained models are easily deployable in constrained nodes and, hence, inference can run on them. However, processing new data and training new models are computationally intensive tasks that are still insufficiently investigated in edge devices. Thus, if a model needs to be retrained to fit better to the user preferences or to integrate new data, the usual answer to overcome this drawback is to move the training stages to cloud servers, something against the benefits of edge computing architectures in terms of privacy. So, the next stage of an intelligent edge is to find new ways to integrate the whole classification pipeline at the edge to perform on-site training and decrease data exposure.

The presented case of study deserves attention to illustrate the potential impact of the optimization strategy that address this piece or research. In its evaluation, the experimental results suggest that, in some applications, the specific processing times for each algorithm can be practically negligible concerning the time required to capture the data and process it. This emphasizes the relevance of understanding the characteristics of the data and its impact in the performance and cost, particularly for the training phases.

Since the importance of this methodology relies on profiling the different factors affecting the classification pipeline, this holistic approach can be extended to other solutions. By way of illustration, personalized healthcare or smart home domains are some other examples where the adaptation of environments and resources should be in line with individual human needs. In those examples, the different factors of the data collection and processing may produce an optimization opportunity that allows to perform those adjustments and re-train the models in consequence. There, applications like heart-rate ECG monitoring involve high-frequency signals which increase the computational cost [72]. Other solutions such as energy forecasting and smart metering applications for smart homes usually involve complex processing techniques and obtaining frequency domain features for modeling energy consumption patterns [73].

Even considering the specific particularities of the different classification applications, an important step to meet this objective is to find the right balance between the computational cost and the final accuracy of the obtained knowledge. To that end, this analysis seeks to assist in providing a strategy for optimizing similar problems and bringing advanced processing capabilities to the edge of the network.

7. Conclusions and Future Work

In this work we have presented a methodology to meet a satisfactory cost-accuracy trade-off on the execution of the whole pipeline of machine learning techniques. This is motivated by the challenge of improving the feasibility of resource-constrained devices to perform inference and training tasks at the edge, contributing to a more human-centric IoT. To that end, we have embraced a holistic approach based on understanding the different factors that take part in the whole classification process (from data acquisition to the final fitting and prediction stages). Besides, we have empirically analyzed and compared the performance of these training and inference phases concerning HAR algorithms within two resource-constrained platforms.

In the evaluated case of study, the proposed strategy has been able to reduce by more than 80 % the baseline processing time of the machine learning techniques in all the evaluated platforms. Furthermore, acceptable detection rates are maintained, with an accuracy consistently over 90 %, and assuming a decline of classification accuracy of only 3 %. We stress that the selection of the machine learning algorithm is not the only important factor to consider in the classification on the edge. The process of understanding the initial data and optimizing the process from the early stages of the classification pipeline is vital to meet the cost-accuracy requirements.

The obtained results provide us with an outlook for future work on the area. Firstly, having applied our proposed strategy to one specific classification problem we cannot ensure that the results can be generalized to any dataset or application. However, we argue that the presented methodology can be reused in similar problems and may lead to similar conclusions. Therefore, we should explore where and how can be extrapolated. Besides, we aim at integrating the solution in a real-world edge architecture for validating the proposal in a online classification scenario.

Finally, the approach presented in this article opens the door to adopt similar mechanisms to tackle the optimization of other applications for IoT edge devices and provides some guidelines suitable for any other sensor data available within the Internet of Things paradigm.

Acknowledgments

We gratefully acknowledge the support of the Basque Governmentś Department of Education for the predoctoral funding of one of the authors and the Deustek Research Group. We also acknowledge the support of the the Ministry of Economy, Industry and Competitiveness of Spain for SentientThings under Grant No.: TIN2017-90042-R.

References

[1] C. A. Simmers, M. Anandarajan, The Internet of People, Things and Services: Workplace Transformations, Routledge, 2018.

[2] R. Sánchez-Corcuera, A. Nuñez-Marcos, J. Sesma-Solance, A. Bilbao-Jayo, R. Mulero, U. Zulaika, G. Azkune, A. Almeida, Smart cities survey: Technologies, application domains and challenges for the cities of the future, International Journal of Distributed Sensor Networks 15 (6) (2019) 1550147719853984.

[3] H. H. Nguyen, F. Mirza, M. A. Naeem, M. Nguyen, A review on iot healthcare monitoring applications and a vision for transforming sensor data into real-time clinical feedback, in: 2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design (CSCWD), IEEE, 2017, pp. 257–262.

[4] J. Cahill, R. Portales, S. McLoughin, N. Nagan, B. Henrichs, S. Wetherall, Iot/sensor-based infrastructures promoting a sense of home, independent living, comfort and wellness, Sensors 19 (3) (2019) 485.

[5] A. Irizar-Arrieta, O. Gómez-Carmona, A. Bilbao-Jayo, D. Casado-Mansilla, D. López-De-Ipiña, A. Almeida, Addressing behavioural technologies through the human factor: A review, IEEE Access 8 (2020) 52306–52322.

[6] O. Gomez-Carmonaa, D. Casado-Mansillaa, J. Garcıa-Zubiab, Opportunities and challenges of technology-ased interventions to increase healthwareness in the workplace, Transforming Ergonomics with Personalized Health and Intelligent Workplaces 25 (2019) 33.

[7] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, W. Zhao, A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications, IEEE Internet of Things Journal 4 (5) (2017) 1125–1142.

[8] S. Pearson, A. Benameur, Privacy, security and trust issues arising from cloud computing, in: 2010 IEEE Second International Conference on Cloud Computing Technology and Science, IEEE, 2010, pp. 693–702.

[9] D. Casado-Mansilla, P. Garaizar, D. López-de Ipiña, User involvement matters: The side-effects of automated smart objects in pro-environmental behaviour, in: Proceedings of the 9th International Conference on the Internet of Things, 2019, pp. 1–4.

[10] M. Conti, A. Passarella, S. K. Das, The internet of people (iop): A new wave in pervasive mobile computing, Pervasive and Mobile Computing 41 (2017) 1–27.

[11] S. Chen, T. Liu, F. Gao, J. Ji, Z. Xu, B. Qian, H. Wu, X. Guan, Butler, not servant: A human-centric smart home energy management system, IEEE Communications Magazine 55 (2) (2017) 27–33.

[12] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, E. Riviere, Edge-centric computing: Vision and challenges, ACM SIGCOMM Computer Communication Review 45 (5) (2015) 37–42.

[13] S. Pape, K. Rannenberg, Applying privacy patterns to the internet of things'(iot) architecture, Mobile Networks and Applications 24 (3) (2019) 925–933.

[14] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, K. Chan, When edge meets learning: Adaptive control for resource-constrained distributed machine learning, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 63–71.

[15] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, J. Zhang, Edge intelligence: Paving the last mile of artificial intelligence with edge computing, arXiv preprint arXiv:1905.10083.

[16] O. Gómez-Carmona, D. Casado-Mansilla, D. López-de Ipiña, J. García-Zubia, Simplicity is best: Addressing the computational cost of machine learning classifiers in constrained edge devices, in: Proceedings of the 9th International Conference on the Internet of Things, IoT 2019, Association for Computing Machinery, New York, NY, USA, 2019. `doi:10.1145/3365871.3365889`.
URL `https://doi.org/10.1145/3365871.3365889`

[17] Z. Hussain, M. Sheng, W. E. Zhang, Different approaches for human activity recognition: A survey, arXiv preprint arXiv:1906.05074.

[18] D. Xu, T. Li, Y. Li, X. Su, S. Tarkoma, P. Hui, A survey on edge intelligence, arXiv preprint arXiv:2003.12172.

[19] F. Samie, L. Bauer, J. Henkel, From cloud down to things: An overview of machine learning in internet of things, IEEE Internet of Things Journal.

[20] T. Adegbija, A. Rogacs, C. Patel, A. Gordon-Ross, Microprocessor optimizations for the internet of things: A survey, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 37 (1) (2017) 7–20.

[21] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, F. Kawsar, An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices, in: Proceedings of the 2015 international workshop on internet of things towards applications, ACM, 2015, pp. 7–12.

[22] U. Jensen, P. Kugler, M. Ring, B. M. Eskofier, Approaching the accuracy–cost conflict in embedded classification system design, Pattern Analysis and Applications 19 (3) (2016) 839–855.

[23] M. Yazici, S. Basurra, M. Gaber, Edge machine learning: Enabling smart internet of things applications, Big Data and Cognitive Computing 2 (3) (2018) 26.

[24] F. Desraches, O. Kamara-Esteban, D. Casado-Mansilla, C. E. Borges, Forecasting the usage of appliances of shared use: an analysis of simplicity over complexity, in: 2018 Energy and Sustainability for Small Developing Economies (ES2DE), IEEE, 2018, pp. 1–6.

[25] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning, in: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), 2016, pp. 265–283.

[26] S. Dhar, J. Guo, J. Liu, S. Tripathi, U. Kurup, M. Shah, On-device machine learning: An algorithms and learning theory perspective, arXiv preprint arXiv:1911.00623.

[27] A. R. Neto, B. Soares, F. Barbalho, L. Santos, T. Batista, F. C. Delicato, P. F. Pires, Classifying smart iot devices for running machine learning algorithms, in: 45^{o} Seminário Integrado de Software e Hardware 2018 (SEMISH 2018), Vol. 45, SBC, 2018, pp. 1–12.

[28] F. Alam, R. Mehmood, I. Katib, A. Albeshri, Analysis of eight data mining algorithms for smarter internet of things (iot), Procedia Computer Science 98 (2016) 437–442.

[29] K. Z. Haigh, A. M. Mackay, M. R. Cook, L. G. Lin, Machine learning for embedded systems: A case study, BBN Technologies: Cambridge, MA, USA.

[30] C. Gupta, A. S. Suggala, A. Goyal, H. V. Simhadri, B. Paranjape, A. Kumar, S. Goyal, R. Udupa, M. Varma, P. Jain, Protonn: Compressed and accurate knn for resource-scarce devices, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 1331–1340.

[31] A. Kumar, S. Goyal, M. Varma, Resource-efficient machine learning in 2 kb ram for the internet of things, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 1935–1944.

[32] L. Andrade, A. Prost-Boucle, F. Pétrot, Overview of the state of the art in embedded machine learning, in: 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2018, pp. 1033–1038.

[33] F. Scheidegger, L. Benini, C. Bekas, A. C. I. Malossi, Constrained deep neural network architecture search for iot devices accounting for hardware calibration, in: Advances in Neural Information Processing Systems, 2019, pp. 6054–6064.

[34] X. Wang, M. Magno, L. Cavigelli, L. Benini, Fann-on-mcu: An open-source toolkit for energy-efficient neural network inference at the edge of the internet of things, arXiv preprint arXiv:1911.03314.

[35] M. Shafique, T. Theocharides, C.-S. Bouganis, M. A. Hanif, F. Khalid, R. Hafız, S. Rehman, An overview of next-generation architectures for machine learning: Roadmap, opportunities and challenges in the iot era, in: 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2018, pp. 827–832.

[36] X. Fafoutis, L. Marchegiani, A. Elsts, J. Pope, R. Piechocki, I. Craddock, Extending the battery lifetime of wearable sensors with embedded machine learning, in: 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), IEEE, 2018, pp. 269–274.

[37] R. Cilla, M. A. Patricio, A. Berlanga, J. M. Molina, Creating human activity recognition systems using pareto-based multiobjective optimization, in: 2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, IEEE, 2009, pp. 37–42.

[38] A. Elsts, R. McConville, X. Fafoutis, N. Twomey, R. J. Piechocki, R. Santos-Rodriguez, I. Craddock, On-board feature extraction from acceleration data for activity recognition., in: EWSN, 2018, pp. 163–168.

[39] P. Zalewski, L. Marchegiani, A. Elsts, R. Piechocki, I. Craddock, X. Fafoutis, From bits of data to bits of knowledge—an on-board clas-

sification framework for wearable sensing systems, Sensors 20 (6) (2020) 1655.

[40] A. Khan, N. Hammerla, S. Mellor, T. Plötz, Optimising sampling rates for accelerometer-based human activity recognition, Pattern Recognition Letters 73 (2016) 33–40.

[41] F. A. Kraemer, F. Alawad, I. M. V. Bosch, Energy-accuracy tradeoff for efficient noise monitoring and prediction in working environments, in: Proceedings of the 9th International Conference on the Internet of Things, IoT 2019, Association for Computing Machinery, New York, NY, USA, 2019. `doi:10.1145/3365871.3365885`.
URL `https://doi.org/10.1145/3365871.3365885`

[42] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, K. Aberer, Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach, in: 2012 16th international symposium on wearable computers, Ieee, 2012, pp. 17–24.

[43] X. Qi, M. Keally, G. Zhou, Y. Li, Z. Ren, Adasense: Adapting sampling rates for activity recognition in body sensor networks, in: 2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), IEEE, 2013, pp. 163–172.

[44] S. Kang, J. Lee, H. Jang, Y. Lee, S. Park, J. Song, A scalable and energy-efficient context monitoring framework for mobile personal sensor networks, IEEE Transactions on Mobile Computing 9 (5) (2009) 686–702.

[45] D. Gordon, J. Czerny, T. Miyaki, M. Beigl, Energy-efficient activity recognition using prediction, in: 2012 16th International Symposium on Wearable Computers, IEEE, 2012, pp. 29–36.

[46] N. D. Lane, S. Bhattacharya, A. Mathur, P. Georgiev, C. Forlivesi, F. Kawsar, Squeezing deep learning into mobile and embedded devices, IEEE Pervasive Computing 16 (3) (2017) 82–88.

[47] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, D. Shin, Compression of deep convolutional neural networks for fast and low power mobile applications, arXiv preprint arXiv:1511.06530.

[48] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, arXiv preprint arXiv:1611.06440.

[49] A. E. Eshratifar, M. S. Abrishami, M. Pedram, Jointdnn: an efficient training and inference engine for intelligent mobile cloud computing services, IEEE Transactions on Mobile Computing.

[50] J. Liu, J. Liu, W. Du, D. Li, Performance analysis and characterization of training deep learning models on nvidia tx2, arXiv preprint arXiv:1906.04278.

[51] P. S. Chandakkar, Y. Li, P. L. K. Ding, B. Li, Strategies for re-training a pruned neural network in an edge computing paradigm, in: 2017 IEEE International Conference on Edge Computing (EDGE), IEEE, 2017, pp. 244–247.

[52] B. Bruno, F. Mastrogiovanni, A. Sgorbissa, A public domain dataset for adl recognition using wrist-placed accelerometers, in: the 23rd IEEE International Symposium on Robot and Human Interactive Communication, IEEE, 2014, pp. 738–743.

[53] D. Dua, C. Graff, UCI machine learning repository (2017).
URL `http://archive.ics.uci.edu/ml`

[54] E. Arias-Castro, D. L. Donoho, et al., Does median filtering truly preserve edges better than linear filtering?, The Annals of Statistics 37 (3) (2009) 1172–1206.

[55] W. Dargie, Analysis of time and frequency domain features of accelerometer measurements, in: 2009 Proceedings of 18th International Conference on Computer Communications and Networks, IEEE, 2009, pp. 1–6.

[56] M. Janidarmian, A. Roshan Fekr, K. Radecka, Z. Zilic, A comprehensive analysis on wearable acceleration sensors in human activity recognition, Sensors 17 (3) (2017) 529.

[57] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Computers & Electrical Engineering 40 (1) (2014) 16–28.

[58] H. Liu, R. Setiono, Chi2: Feature selection and discretization of numeric attributes, in: Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence, IEEE, 1995, pp. 388–391.

[59] J. Suto, S. Oniga, P. P. Sitar, Feature analysis to human activity recognition, International Journal of Computers Communications & Control 12 (1) (2017) 116–130.

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, Journal of machine learning research 12 (Oct) (2011) 2825–2830.

[61] N. Twomey, T. Diethe, X. Fafoutis, A. Elsts, R. McConville, P. Flach, I. Craddock, A comprehensive study of activity recognition using accelerometers, in: Informatics, Vol. 5, Multidisciplinary Digital Publishing Institute, 2018, p. 27.

[62] P. Domingos, M. Pazzani, On the optimality of the simple bayesian classifier under zero-one loss, Machine learning 29 (2-3) (1997) 103–130.

[63] C. Qian, Y. Yu, Z.-H. Zhou, Subset selection by pareto optimization, in: Advances in Neural Information Processing Systems, 2015, pp. 1774–1782.

[64] S. v. d. Walt, S. C. Colbert, G. Varoquaux, The numpy array: a structure for efficient numerical computation, Computing in Science & Engineering 13 (2) (2011) 22–30.

[65] D. Casado-Mansilla, P. Garaizar, A. Irizar-Arrieta, D. López-de Ipiña, On the side effects of automation in iot: Complacency and comfort vs. relapse and distrust, arXiv preprint arXiv:1911.08657.

[66] E. Kaasinen, T. Kymäläinen, M. Niemelä, T. Olsson, M. Kanerva, V. Ikonen, A user-centric view of intelligent environments: User expectations, user experience and user role in building intelligent environments, Computers 2 (1) (2013) 1–33.

[67] S. Amershi, M. Cakmak, W. B. Knox, T. Kulesza, Power to the people: The role of humans in interactive machine learning, Ai Magazine 35 (4) (2014) 105–120.

[68] F. Bernardo, M. Zbyszynski, R. Fiebrink, M. Grierson, Interactive machine learning for end-user innovation, in: 2017 AAAI Spring Symposium Series, 2017, pp. 1–7.

[69] T. Miu, P. Missier, T. Plötz, Bootstrapping personalised human activity recognition models using online active learning, in: 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, IEEE, 2015, pp. 1138–1147.

[70] H. L. Cardoso, J. M. Moreira, Human activity recognition by means of online semi-supervised learning, in: 2016 17th IEEE International Conference on Mobile Data Management (MDM), Vol. 2, IEEE, 2016, pp. 75–77.

[71] K. Yu, S. Berkovsky, D. Conway, R. Taib, J. Zhou, F. Chen, Do i trust a machine? differences in user trust based on system performance, in: Human and Machine Learning, Springer, 2018, pp. 245–264.

[72] O. Kwon, J. Jeong, H. B. Kim, I. H. Kwon, S. Y. Park, J. E. Kim, Y. Choi, Electrocardiogram sampling frequency range acceptable for heart rate variability analysis, Healthcare informatics research 24 (3) (2018) 198–206.

[73] Y. Wang, Q. Chen, T. Hong, C. Kang, Review of smart meter data analytics: Applications, methodologies, and challenges, IEEE Transactions on Smart Grid 10 (3) (2018) 3125–3148.