Energy-Efficient Activity Recognition Framework using Wearable Accelerometers

Atis Elsts[a,b,*], Niall Twomey[b], Ryan McConville[b], Ian Craddock[b]

[a] *Institute of Electronics and Computer Science, 14 Dzerbenes St., LV-1006, Riga, Latvia*
[b] *Department of Electrical and Electronic Engineering, University of Bristol, 1 Cathedral Square, Bristol, BS15DD, UK*

Abstract

Acceleration data for activity recognition typically are collected on battery-powered devices, leading to a trade-off between high-accuracy recognition and energy-efficient operation. We investigate this trade-off from a feature selection perspective, and propose an energy-efficient activity recognition framework with two key components: a detailed energy consumption model and a number of feature selection algorithms. We evaluate the model and the algorithms using Random Forest classifiers to quantify the recognition accuracy, and find that the multi-objective Particle Swarm Optimization algorithm achieves the best results for the task. The results show that by selecting appropriate groups of features, energy consumption for computation and data transmission is reduced by an order of magnitude compared with the raw-data approach, and that the framework presents a flexible selection of feature groups that allow the designer to choose an appropriate accuracy-energy trade-off for a specific target application.

Key words: feature selection, activity recognition, wearables

1. Introduction

Internet of Things (IoT) networks and applications have gained tremendous popularity in the recent years [1, 2]. This includes applications of wearable devices [3].

*Corresponding author
Email addresses: `atis.elsts@edi.lv` (Atis Elsts)

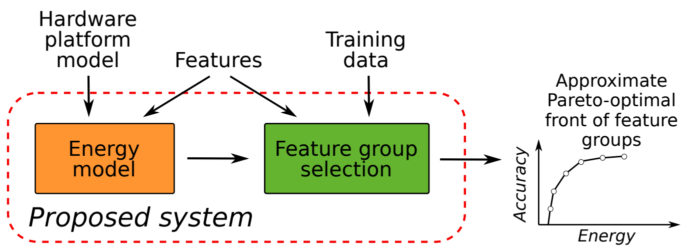


Figure 1: Overview of the proposed system.

Acceleration data from wearable devices are widely used for human activity recognition applications in healthcare [4, 5], fitness [6], long-term behavior monitoring [7] and other areas. Their typical application uses a multistage process: after segmenting and filtering the raw sensor data, a number of statistical features are computed and then used as inputs for a machine learning classifier. Wearable devices are battery powered; they have limited energy budgets, and the balance between high accuracy and energy-efficient operation is important.

Wearable-based behavior monitoring studies often require a prolonged collection of data. Many commercial wearables require frequent recharging, but activity recognition systems for clinical or research purposes may not have the luxury of users that conform to a strict and cumbersome device-charging schedule. For elderly or ill people, the requirement to frequently recharge their devices may even be unethical. It is natural for designers of human activity recognition systems to ask these key questions:

- Given a specific target activity recognition accuracy, for what maximum time wearables can be deployed before they need to be recharged?
- Given a specific target deployment time, what is the maximum accuracy obtainable without recharging wearables during the deployment?

Contributions. This paper proposes a system (Fig. 1) that helps to be answer these questions. It is a framework for finding groups of features that have approximately optimal energy-accuracy trade-offs for a specific target application (i.e., classification of human activities of daily living) on a specific target platform. The framework consists of an energy model that describes the energy costs of feature extractions and transmissions together with a feature selection algorithm that optimizes both for accuracy and energy efficiency. It uses training data collected from a previous study or from pilot experiments, a set of candidate platform, and a hardware platform model as inputs, and produces the approximate Pareto-optimal front of non-dominated feature groups as the output. Our specific contributions are:

- We present a novel feature energy model that accounts for inter-dependencies between features to better estimate the energy consumption in the feature extraction process.
- We evaluate a number of feature group selection algorithms for the application domain.
- We present evidence about the suitability of the Particle Swarm Optimization (PSO) algorithm, which we implement it in two different versions: as a multi-objective and as a single-objective optimization problem.

Prototype system and results. This paper assumes a setup where the sampling, preprocessing and feature extraction are done on the device, and the resulting features are wirelessly transmitted to a central system. We implement a C library for on-board feature extraction, run it on an ARM Cortex-M3 device, and measure the feature extraction time to estimate energy consumption. The energy consumption model as well as three different datasets are used as inputs to the feature group selection algorithms.The evaluation scores the results in two dimensions: first, charge consumption for feature computation and transmission; second, the F_1 score for activity recognition. It compares the Pareto-optimal fronts selected by the PSO algorithms with those selected by methods from our previous work [8]: greedy search and mutual information (MI) based search. We evaluate the proposed system for classification of human activities of daily living with a Random Forest classifier, and compare the accuracy of the PSO algorithms with our previous work [8]. The PSO algorithms produce results that are closer to optimum than the alternatives, and the multi-objective PSO also finds the highest number of points on the front. The feature selection is assumed to be done offline, before the deployment of the data collection and feature extraction code, so that after running the feature group selection algoriths the desired features can be directly encoded in the deployed software.

Compared with our previous work [8] the present research adds selection of feature groups instead of merely evaluating individual features. We extend the feature extraction code from [8] with feature groups, several new features, and generic transforms and filters. Furthermore, we add the complete energy model, and describe how the system can be used to construct a practical feature extraction framework.

Summary of the paper. The paper first surveys the related work (Section 2). Subsequently it presents the energy model (Section 3) and the feature group selection algorithms (Section 4). The evaluation of the framework is given in Section 5, and application examples in Section 6. Finally, the paper ends with conclusions (Section 7).

Nomenclature

F_1	Precision and recall based measure of a test's accuracy
BLE	Bluetooth Low Energy
CBOR	Concise Binary Object Representation
HAR	Human Activity Recognition
IoT	Internet of Things
MI	Mutual Information
PAMAP	Physical Activity Monitoring for Aging People
PSO	Particle Swarm Optimization
RF	Random Forest
SMA	Signal Magnitude Area
SPHERE	Sensor Platform for Healthcare in a Residential Environment
SPW-2	SPHERE Wearable 2
UCI	University California Irvine

2. Related Work

Activity Recognition. Accelerometer is a core sensor for human activity recognition [9, 10]. Even though the recognition accuracy can be improved by using multiple accelerometers at different locations on the body, good results for coarse-grained activities can be obtained just from a single, typically wrist-worn device [11] – a setup that we assume in this paper.

Activity detection using deep learning can achieve state-of-the-art accuracy [12]. However, deep learning is not suitable for the ultra-low energy consumption Class-1 IoT devices [13] our system targets; instead, it typically targets smartphone-class devices [14] and beyond. The work by Lane *et al.* on deep learning for ARM Cortex-M is one exception from this trend; however, they admit that "work remains to make deep models of this scale completely practical" as they cannot be executed in real time [15].

Energy Efficiency in Activity Recognition. Energy efficiency has been a major research goal for the community, as well as a driver for Edge Computing – the trend where computation moves away from the cloud and closer to the data-producing devices [16]. Our work is an instance of the Edge Computing paradigm.

In most of the related work, the accuracy-energy trade-off is not explicitly defined; rather, the strategy is to achieve subjectively "good-enough" accuracy while optimizing the energy usage [17, 18, 19]. As a result the minimal accuracy threshold is hidden in the details in the proposed systems. By being explicit and not forcing a single threshold value, our work achieves better transparency and flexibility.

Yan *et al.* [17] propose to optimize sampling rate and classification features on mobile phones separately for each activity, in a real-time, adaptive fashion. The system proposed in our paper can be applied to select the features for a single, specific activity or a subgroup of activities, serving as a building block in their approach.

Another approach is to decide which sensors can be turned off without losing a lot accuracy. Gordon *et al.* [18] optimize sensor usage based on prediction of future activities. Similarly, in case of multiple sensor devices, some of them can be delegated to "backup" status, thus saving the energy spent by the whole system [20]. Again, these approaches can complement the feature-selection system of this paper. Trivially, a sensor can be turned off if no features use the data produced by this sensor; the energy saved by that would be be captured by the platform's energy model.

Hierarchical activity recognition is another natural extension. For example, Liang *et al.* [19] propose a hierarchical recognition algorithm that only computes the more expensive frequency domain features when the activity cannot be reliably classified by time domain features. Zheng *et al.* [21] show that a hierarchical classifier allows to reduce the sampling frequency several times while maintaining "high accuracacy". Hierarchical classifiers are beyond the scope of the present paper, however, we aim to generalize the results for this in our future work.

Feature Extraction. In terms of feature extraction on low-power embedded devices, we build on our previous work [8]. We extend the work by adding the notion of generalized transforms in the feature extraction stage. We also add a number of new features, and drop those features that showed bad energy-accuracy trade-off in our previous work.

Feature Selection. We build on the extensive existing work in feature selection [22] and experiment with both wrapper and filter methods [23]. The particle swarm optimization method [24] has been previously proposed for feature selection [25]. That includes the multi-objective optimization that relies on nondominated sorting [26]. However, the energy costs of the recognition are typically not quantified in detail; frequently, existing works use the number of features as a proxy for cost (i.e., energy consumption); see [27, 28] for examples. In this paper, we provide a detailed energy model for computing the cost of feature groups.

Accuracy-Energy Trade-Offs. One typical way to investigate the trade-off for the target application is to compare off-node and on-node activity recognition schemes [29]. Our work falls in between these two extreme approaches: while the recognition is done off-node, the software one the node is optimized in an application-specific way to extract only the features that are required by the application.

Chu *et al.* propose a system for multi-objective optimization of mobile sensor classifiers [30]; while the Pareto-optimal offline optimization approach is the same as used in our paper, we operate at the level of feature groups, rather than classifiers. Similarly, Jensen *et al.* propose a method for approaching the accuracy-cost conflict by choosing an appropriate classifier [31]; however, they ignore the feature selection step, as well as abstract away from the target hardware instead of using an empirical energy model.

3. Energy Model

3.1. Features, Transforms, and Filters

Let us denote the vector of the raw samples with $\boldsymbol{s} = (s_1, s_2, \ldots, s_n)$, where $s_i \in \mathbb{R}$. Normally, acceleration data is three dimensional, i.e., there are three vectors $\boldsymbol{s}_x = (x_1, x_2, \ldots, x_n), \boldsymbol{s}_y = (y_1, y_2, \ldots, y_n), \boldsymbol{s}_z = (z_1, z_2, \ldots, z_n)$ corresponding to acceleration in the three spatial dimensions.

In a preprocessing stage, the data is segmented in windows. Assuming window size w and processing interval k, the j-th window of the input data is the vector $W(s)_j = (s_{j \cdot k}, s_{j \cdot k+1}, \ldots, s_{j \cdot k+w-1})$. If $k < w$, the neighboring windows overlap each another.

Features, *transforms* and *filters* are functions that act on the raw data, either on a single dimension separately or the vector of the three spatial dimensions. The difference between a them is that a feature f is calculated once per window ($f : \mathbb{R}^w \to \mathbb{R}$ or $f : \mathbb{R}^{3w} \to \mathbb{R}$), while a transform or a filter t creates an output value for every input value ($t : \mathbb{R} \to \mathbb{R}$ or $t : \mathbb{R}^3 \to \mathbb{R}$). The difference between the transform and a filter is that a transform does not lose information and is reversible. For simplicity, in some occasions in this paper we use the term "transform" to denote any function that conforms to the output value criteria above.

3.2. Feature Preselection

The list of candidate features is given in Table 1. We also introduce a number of *transforms and filters* (Table 2) that preprocess the data before the feature extraction. For example, transforming the data with the *magnitude squared* function makes it more robust to rotations of the wearable compared with computing features of each axis separately. (Note that the list does not include the *magnitude* filter. It was deemed too expensive, since it requires to compute a square root operation for each (x_i, y_i, z_i) sample.) All data is first passed to a median-of-three filter to de-noise it. This filter is assumed to be always enabled, and as such not handled by the group selection process.

Table 1: Features.

Feature	**Definition**
Mean	$\mu_s = \frac{1}{w}\sum_{i=1}^{w} s_i$
Minimum	$min(s)$
Maximum	$max(s)$
First Quartile	$sorted(s)_{w/4}$
Median	$sorted(s)_{w/2}$
Third Quartile	$sorted(s)_{3w/4}$
Inter-quartile range	$sorted(s)_{3w/4} - sorted(s)_{w/4}$
Energy	$E_s = \frac{1}{w}\sum_{i=1}^{w} (s_i)^2$
Standard Deviation	$\sqrt{E_s - (\mu_s)^2}$
Correlation	$C(\boldsymbol{s}_u, \boldsymbol{s}_v) = \frac{\sum_{i=1}^{w}(u_i-\mu_u)(v_i-\mu_v)}{\sqrt{\sum_{i=1}^{w}(u_i-\mu_u)^2 \sum_{i=1}^{w}(v_i-\mu_v)^2}}$
Entropy	$-\sum_{i=1}^{w} P(s_i)\log P(s_i)$

Table 2: Transforms and filters.

Transform/Filter	**Definition**
Median-of-three	$median(s_{i-1}, s_i, s_{i+1})$
Jerk	$s_i - s_{i-1}$
L1 norm	$abs(x_i) + abs(y_i) + abs(z_i)$
Magnitude squared	$x_i^2 + y_i^2 + z_i^2$

The results in [8] show that for recognition of a limited set of coarse-grained activities of daily living (such as walking, standing, sitting, and lying) simple time-domain features have the best energy-accuracy trade-offs. Inspired by those results, we only use time-domain features for this paper, eschewing the need to run the Fourier transform or other similar transforms on the device to obtain frequency-domain features. To make it clear, this

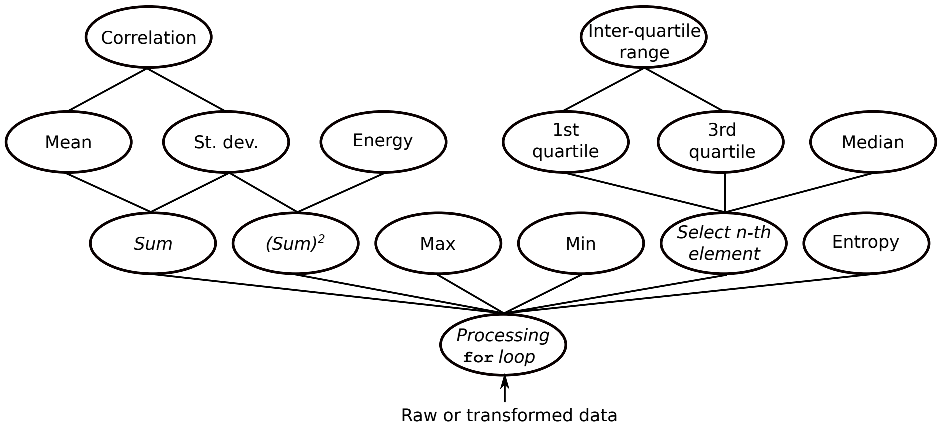


Figure 2: Features under consideration and their inter-dependencies. Labeled in *italic*: intermediate results that are included in the energy model, but not in the feature group selection stage.

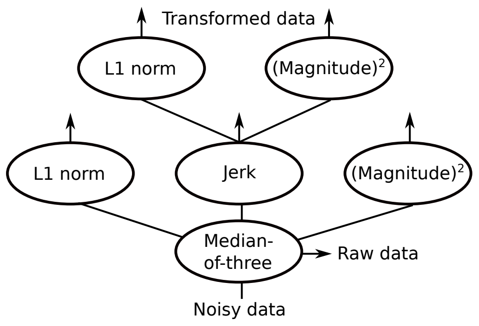


Figure 3: Transforms and filters applied to the raw data.

pre-selection is done because of pragmatic reasons; the approach described further in this paper is not limited to the specific functions we are using.

Floating-point arithmetic is used to compute the *standard deviation*, *correlation* between axis, *energy* and *entropy*. The remaining features, including the *mean*, use only fixed-point arithmetic.

We note that the final list of features includes time domain features typically used in published research in this field, even if occasionally under different names. For example, the "κ feature" defined and used by Wang *et al.* [29] is included implicitly: as mean computed on the *jerk*-transformed data in its normalized version. The *Signal Magnitude Area (SMA)* feature [9] is also included implicitly, as the mean computed on the L1 norm.

In further analysis, we assume that all features are computed on all three axis (x, y, z) of acceleration data, where applicable. The inter-axis correlation feature is computed for all three pairs of axis (xy, xy and yz).

3.3. Energy Costs

Let us define the *cost* of $\mathfrak{f}$, where $\mathfrak{f}$ is a function that is either a feature or a transform, as the energy needed to iteratively compute the function on a single window W of samples ($W \in \mathbb{R}^{3w}$ or $W \in \mathbb{R}^{w}$).

Features and transforms can be combined; for example, one can first transform the data using the *jerk* transform, then transform the result using the *magnitude squared* transform, then segment the data and calculate the standard deviation of each segment. More generally, the combinations of any two different transforms t_i and t_j yields two new transforms $t_i(t_j(s))$ and $t_j(t_i(s))$ in our model. Similarly, any transform t can be combined with any feature f to yield a new feature $f(t(s))$.

Multiple features cannot be combined in this general way; however, one can notice that there are directional dependencies between some of the features. For example, to calculate the standard deviation, one must calculate the mean. Therefore if both the standard deviation and the mean are included in a group of features, then their total calculation cost is equal to the calculation cost of the standard deviation, not the sum of the costs of these two individual features. In Section 3.4 we describe such an optimized implementation, and use it further in the paper.

More generally, if f_1 and f_2 are features that both use an intermediate result $\mathfrak{g}$, where $\mathfrak{g}$ is either a feature or a transform, then the cumulative cost of the feature set $\{f_1, f_2\}$ is:

$$cost(\{f_1, f_2\}) = cost(f_1) + cost(f_2) - cost(\mathfrak{g}) \tag{1}$$

In the special case when the intermediate result $\mathfrak{g}$ is equal to one of the features f_1 or f_2:

$$cost(\{f_1, f_2\}) = max(cost(f_1), cost(f_2)) \tag{2}$$

Let us generalize Eq. 1. First, let us assume that the energy cost of a set $\{f_1, \ldots, f_m\}$ of features and transform is already known and equal to c_m, and that the task is to add a new feature f_{m+1} to this set that uses some intermediate result $\mathfrak{g}$ that is already computed. Then the cost of the combined set is:

$$c_{m+1} = cost(\{f_1, \ldots, f_{m+1}\}) = c_m + cost(f_{m+1}) - cost(\mathfrak{g}). \tag{3}$$

This approach is used to iteratively compute the cost of a set of features using their individual costs (Section 3.5) for the target hardware platform (Section 3.4) using the dependencies shown in Figs. 2 and 3.

3.4. Example Hardware Platform

3.4.1. Platform Description

We evaluate the cost of the on-board feature extraction on SPW-2 [32] (Fig. 4), an embedded hardware platform based on ARM 32-bit Cortex-M3 core. Its limited RAM and program memory size (20 kB and 128 kB, respectively) and CPU speed (48 MHz) do not allow to run high-complexity algorithms. However, the System-on-Chip has a 2.4 GHz ultra-low power wireless radio for data transmission.

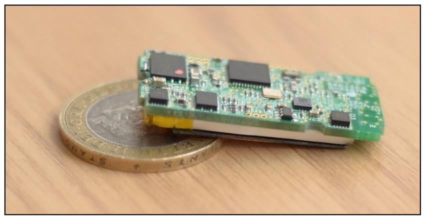


Figure 4: SPW-2: ARM Cortex-M3 based wearable accelerometer platform [32].

3.4.2. Computation

We implement the feature extraction as a stand-alone library[1]. The library is written in C programming language; the code is fully compatible with the C99 language standard and portable, as it does not contain any ARM Cortex specific functionality. To approximate the energy cost of computing each feature, we experimentally evaluate it on the SPW-2. To achieve that, the library is linked with the Contiki-NG operating system[2].

The evaluation of the library consists of performance measurements of 15 000 samples of real 3-axial acceleration data samples, taken from the SPHERE Challenge dataset. For each function, we measure the time it takes to segment the samples in 128-sample windows with 50 % overlap and compute that feature for each window. This window size and overlap has been shown to give good results in previous research [9, 10].

[1]Available at `https://github.com/atiselsts/feature-group-selection`
[2]`http://contiki-ng.org/`

The evaluation results consist of timing measurements that capture the time required to compute each feature. The features are computed on data that is scaled to the range of 8-bit signed integer. As the active-mode current consumption of the SPW-2 platform [32] is constant, the time taken for the computation accurately corresponds to the charge consumption of the microcontroller. We use the electric charge as the main metric, rather than energy (charge times voltage). The CC2650 System-on-Chip has high dynamic range of voltage (from 1.8 to 3.8 V); the exact number is a platform-specific value not relevant to the optimization goals of this paper.

The C library contains both the implementation of individual features and the implementation of feature groups, such as the group {*mean*, *standard deviation*}. The latter is implemented separately, as a group. It is more efficient that way since these features are interdependent. Specifically, both features require the computation of the sum of samples in each window. The inter-dependencies from Fig. 2 are used to decide which feature groups to implement in this combined way.

Note that each feature requires to process the data in a `for` loop. We assume that in an optimized implementation to extract a specific group of N features, there would be just one `for` loop. To accurately evaluate the cumulative charge consumption of this group from our experimental data, we need to sum their individual costs and then subtract the cost of the empty `for` loop multiplied by $N - 1$ (see Eq. 3).

3.4.3. Data Transmission

The CC2650 System-on-Chip supports two radio modes: BLE (Bluetooth Low Energy) and IEEE 802.15.4. As a result, we select IEEE 802.15.4 for our transmission model.

We use a model that assumes a 50 % overhead. That is, the model assumes that in order to transmit one byte of application-layer payload, two bytes need to be transmitted in total. This accounts for packet header overhead, for ACKs, and for occasional retransmissions of complete packets.

To calculate the amount of the application data to transmit, the results of the feature extraction algorithm are encoded in an efficient way. For integers, CBOR [33] encoding is used, while for floating point numbers: their size reduced to 16 bits. Finally, to estimate the charge consumption, we measured the transmission-mode current of the target platform. When the transmission output power is set to 5 dBm, it is approximately 12.0 mA.

Table 3: Charge consumption for feature extraction on the SPW-2 wearable platform.

Feature / transform / filter	**Cost**	**CPU time**	**Avg. current**
	(per 128 sample window)		**(at 50 Hz)**
Mean	0.026 μC	6.8 μs	0.067 μA
Minimum	0.026 μC	6.8 μs	0.067 μA
Maximum	0.026 μC	6.8 μs	0.067 μA
First quartile	0.064 μC	16.8 μs	0.165 μA
Median	0.064 μC	16.8 μs	0.165 μA
Third quartile	0.064 μC	16.8 μs	0.165 μA
Inter-quartile range	0.070 μC	18.2 μs	0.179 μA
Energy	0.032 μC	8.4 μs	0.083 μA
Standard deviation	0.035 μC	9.2 μs	0.090 μA
Correlation	0.067 μC	17.3 μs	0.170 μA
Entropy	0.257 μC	66.9 μs	0.659 μA
Median-of-three	0.033 μC	8.6 μs	0.085 μA
L1 norm	0.034 μC	8.9 μs	0.088 μA
Magnitude squared	0.029 μC	7.6 μs	0.075 μA
Jerk + L1 norm	0.047 μC	12.2 μs	0.120 μA
Jerk + Magnitude sq.	0.048 μC	12.5 μs	0.123 μA
Empty `for` loop	0.010 μC	3.2 μs	0.032 μA

Table 4: Charge consumption for transmission on the SPW-2 wearable platform.

Feature	**Cost per window**	**Avg. current**
	(128 samples)	**(at 50 Hz)**
Mean	0.89 μC	2.29 μA
Minimum	1.02 μC	2.60 μA
Maximum	1.17 μC	3.00 μA
First quartile	1.02 μC	2.60 μA
Median	1.02 μC	2.60 μA
Third quartile	1.02 μC	2.60 μA
Inter-quartile range	0.84 μC	2.16 μA
Energy	1.49 μC	3.81 μA
Standard deviation	1.49 μC	3.81 μA
Correlation	1.49 μC	3.81 μA
Entropy	1.49 μC	3.81 μA
Raw data	31.46 μC	80.54 μA

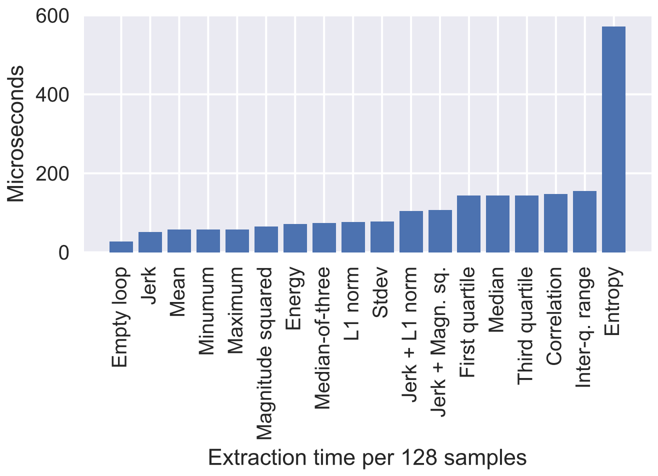


Figure 5: Extraction time for features and transforms.

3.5. Model Instantiation for the Example Hardware Platform

Table 3 and Table 4 show the instantiation of the charge consumption model. The Fig. 5 graphically displays the feature extraction time from the Table 3. The charge consumption costs are given for a single axis of acceleration data. In general, it is more than an order of magnitude cheaper to compute a feature than to transmit the result of the computation. The only exception is the *entropy* feature. Transmission of the raw data unsurprisingly is another order of magnitude more expensive, since it means sending 64 measurements per each window instead of sending just one value.

4. Feature Group Selection Methodology

4.1. Preliminaries

In contrast to single-objective optimization that optimizes over scalars, multi-objective optimizes over vector-valued functions. These optimization problems take the following general form:

$$\begin{aligned} &\min\left(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})\right) \\ &\text{s.t. } \mathbf{x} \in \mathcal{X}, \end{aligned}$$

in which the k functions to be optimized are denoted as f_i (with $1 \leq i \leq k$), and $\mathcal{X}$ is the feasible set of solutions.

A key concept within multi-objective domain is that of dominant solutions. A solution $\mathbf{x}_1 \in \mathcal{X}$ is said to dominate another solution $\mathbf{x}_2 \in \mathcal{X}$ if:

1. $f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2)\ \forall\ i\ (1 \leq i \leq k)$; and
2. $f_i(\mathbf{x}_1) < f_i(\mathbf{x}_2)$ at least once.

This important property means that $\mathbf{x}_1$ is never *worse* than $\mathbf{x}_2$. If a solution $\mathbf{x}_* \in \mathcal{X}$ dominates the set $\mathcal{X}\backslash\{\mathbf{x}_*\}$, then $\mathbf{x}_*$ is said to be *Pareto optimal.* Pareto optimality is noteworthy since the performance of any single objective at a Pareto optimal solution cannot be improved without compromising performance on the other objectives.

The set of Pareto optimal solutions is called the *Pareto front* and it establishes the relationship between a set of Pareto optimal solutions and a set of operating contexts. In this work, the power budget for feature representation calculation defines the operating context. In other words, with access to the Pareto front, feature representations can be adjusted depending on the power budget. Typically, the front will be calculated offline and deployed to the embedded device. The computational expense required to calculate the Pareto front is the primary reason for this, however, the resulting model is trivial to evaluate on embedded devices.

4.2. The Multi-Objective Optimization Problem

The optimization problem in the context of this work is defined as:

$$\begin{aligned} &\textit{minimize}\ (-a(\boldsymbol{f}), e(\boldsymbol{f})) && (4)\\ &\text{subject to}\ ||\boldsymbol{f}|| > 0, && (5) \end{aligned}$$

where $\boldsymbol{f}$ is a set of feature vectors, $a(\boldsymbol{f})$ is the classification accuracy given $\mathbf{f}$, and $e(\boldsymbol{f})$ is the energy cost to compute and transmit $\boldsymbol{f}$. The *solution* of this optimization problem is the Pareto front of k non-dominated sets of feature vectors $\boldsymbol{f}^{(1)}, \boldsymbol{f}^{(2)}, \ldots, \boldsymbol{f}^{(k)}$. The *granularity* of the solution is the number k.

Within this work, we are concerned with two objectives (*i.e.* $k = 2$): high predictive accuracy, and low power consumption for data representation. Taking into account all features and their combinations with the different transforms (Section 3), there are 54 total feature vectors under consideration. Since the number of subsets in a 54-element set is very large, it is not possible to apply a brute force algorithm to find the nondominated subsets of feature vectors. If more features such as frequency domain features are added, the need to reduce the computational complexity of the search becomes even stronger. Note that some of the features are three-dimensional vectors, e.g., *mean*, when computed on a segment of the raw data, results in the triple ($mean_x$, $mean_y$, $mean_z$). If these were separated along the three axis, that would improve the granularity of the results, but also massively increase the number of the features and thus the search space.

4.3. Activity Recognition Classifier

We use the Random Forest classifier to evaluate the accuracy. The general approach described in this paper is not specific to any particular classifier; we selected the Random Forest because it is computationally inexpensive and robust, and has shown good results in a wide range of applications. Furthermore, the features do not need to be normalized when the Random Forest is used; this reduces the computation required for feature extraction. The classifier is implemented using the *scikit-learn* library. The number of trees is set to 100 (the default for version 0.22), and the `class_weight` parameter set to "balanced" to handle skewed class distributions.

4.4. Selection Algorithms

Feature selection methods are categorized in wrapper, filter, and embedded methods [23]. The first treats the problem as a black box, the second uses a pre-processing step independent of the classifier, and the third uses information specific to the classifier. We compare a number of wrapper methods: greedy search and PSO based search, as well as one filter method: mutual information based selection. In terms of embedded methods, the feature importances in the Random Forest is a potential candidate. However, the splits in the decision tree construction process are selected in a way that maximizes information gain. Therefore, the results of selecting by feature importances are going to be the same as when selecting by MI.

4.4.1. Greedy Search

The idea of the greedy search is to start with an empty set of selected features, and then add a single highest-scoring feature in each step. The performance of a candidate group of features $\boldsymbol{f}$ is measured by training a Random Forest classifier on the training data and evaluating its accuracy on the validation data. The measurement score S linearly combines the F_1 score of this evaluation with the energy consumption E of the group $\boldsymbol{f}$:

$$S = W_E E + W_A F_1, \tag{6}$$

The weights W_A and W_E are selected to scale the accuracy and energy metrics to similar amplitude and the same direction: $W_A = -500\,W_E$. Energy is a large number that needs to be minimized, and F_1 score needs to be maximized, subject to $0.0 \leq F_1 \leq 1.0$. Once a feature is selected, it is never removed from the set. See the Algorithm 1 for the details.

Algorithm 1 Greedy Search

```
                                                              ▷ Initialization
max_cost ← energy_cost({raw_data})
selected_features ← ∅
score = $-\infty$
pareto_front = list()
while true do                                                      ▷ Main loop
    best_candidate_score = $-\infty$
    for $f \in$ candidate_features do
        if $f \notin$ selected_features then
            candidate_selection = selected_features ∪ {f}
            new_score ← evaluate_energy_and_f1score(candidate_selection)
            if new_score > best_candidate_score then
                best_candidate_score ← new_score
                best_candidate ← f
            end if
        end if
    end for
    selected_features ← selected_features ∪ {best_candidate}
    if energy_cost(selected_features) ≥ max_cost then
        break
    end if
    improvement ← best_candidate_score − score
    score ← best_candidate_score
    pareto_front.append(selected_features)
end while
return pareto_front
```

4.4.2. Mutual Information Based Selection

Mutual information (MI) is a statistical measure between two random variables X and Y that quantifies the reduction in uncertainty about one random variable given knowledge of another. High MI indicates a large reduction in uncertainty. Hence, MI measures the reduction in uncertainty about the classification target Y given a feature X. More formally, given

Algorithm 2 Mutual Information Based Selection

```
                                                        ▷ Initialization
  $max\_cost \leftarrow energy\_cost(\{raw\_data\})$
  $selected\_features \leftarrow \varnothing$
  $score = -\infty$
  $pareto\_front = list()$
  $MI\_list = list()$
  while $true$ do                                       ▷ Main loop
      for $f \in candidate\_features$ do
          $MI\_list \leftarrow sort(calculate\_MI(f, classes))$
      end for
      for $f \in MI\_list$ do
          $selected\_features = selected\_features \cup \{f\}$
          $new\_score \leftarrow evaluate\_energy\_and\_f1score(selected\_features)$
      end for
      if $energy\_cost(selected\_features) \geq max\_cost$ then
          $break$
      end if
      $pareto\_front.append(selected\_features)$
  end while
  return $pareto\_front$
```

discrete random variables X and Y, the MI between them is:

$$I(X;Y) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p(x,y) \log\left(\frac{p(x,y)}{p(x)\,p(y)}\right) \tag{7}$$

where $p(x, y)$ is the joint probability distribution function of X and Y, and $p(x)$ and $p(y)$ the marginal probability distribution functions of X and Y.

In the MI based selection, all features are initially ranked according to their MI with the classification target classes. Then, the highest ranking features are one-by-one added to the candidate set, until a predetermined number of features have been chosen (Algorithm 2). This is a filter based method; in contrast to the greedy search, it does not use information from classification results to guide the search.

4.4.3. Particle Swarm Optimization Based Search

The Particle Swarm Optimization (PSO) is a global stochastic optimization method. It uses a population of candidate solutions (particles). The *position* of a particle is defined as the n-dimensional vector describing the particles coordinates in the search space. The *velocity* is another n-dimensional vector describing the rate of change of the position. The PSO algorithm is iterative; in each iteration it updates the particles according to simple mathematical rules based on the particles' positions and velocities.

The PSO algorithm is a popular meta-heuristic method for solving non-linear optimization problems, including feature selection [25]. It is suitable for searching in a very large space of candidate solutions, and does not require the optimization function to be differentiable. However, as with other stochastic optimization methods, PSO is not guaranteed to find the global optima. It may also take a long time to converge.

For the purposes of this paper, we define the search space as the power set of the candidate features. Elements of the particle's position vector can take values from 0.0 to 1.0. If the value of an position element x_i is greater than the `THRESHOLD` constant, the i-th feature is defined as *selected* by the particle;`THRESHOLD = 0.9` in our implementation to bias the search towards sparser selections.

Table 5: PSO algorithm parameters (from [25]).

Parameter	**Value**
Maximum Iterations	100
Number of Particles	10 000
Inertia Weight	0.7298
Max Speed	0.6
Acceleration c_1	1.49618
Acceleration c_2	1.49618

We implement two versions of the PSO search:

- Single objective. Here the score S of a particle is a scalar, calculated as in the Eq. 6. The traditional PSO algorithm is used [34].

- Multi-objective. Here the score of a particle is 2-dimensional vector that includes the energy and F_1 score values as its elements. As traditional PSO method cannot handle multi-objective optimization, we utilize the NSPSOFS algorithm by Xue *et al.* [25]. This algorithm relies on nondominated sorting [26] to produce the Pareto-optimal fronts in each iteration, and attempts to move the rest of the particles towards this front. In each iteration it also prunes the Pareto-optimal fronts by sorting its particles by crowding (distance to neighbors) and removing 25 % of the most overcrowded particles.

Algorithm 3 shows the details how the PSO methods are incorporated in the feature group selection process. Table 5 lists configuration parameters of the PSO algorithm; the weight, speed and acceleration parameters are taken from Xue *et al.* [25]. For a detailed explanation of the PSO algorithms, in particular the multi-objective version, we ask the reader to consult [25].

Algorithm 3 PSO Based Search

```
                                                          ▷ Configuration constants
$NUM\_PARTICLES \leftarrow 10\,000$
                                                          ▷ Initialization
$particles \leftarrow \varnothing$
for $f_1 \in candidate\_features$ do
    for $f_2 \in candidate\_features$ do
        if $f_1 \neq f2$ then
            $p \leftarrow Particle()$
            $p.features \leftarrow list(f_1, f_2)$
            $particles \leftarrow particles \cup \{p\}$
        end if
    end for
end for
while $length(particles) < NUM\_PARTICLES$ do
    $p \leftarrow Particle()$
    $p.features \leftarrow random\_subset(candidate\_features)$
    $particles \leftarrow particles \cup \{p\}$
end while
for $p \in particles$ do
    $p.score \leftarrow evaluate\_energy\_and\_f1score(p.features)$
end for
                                                          ▷ Optimization
$run\_particle\_swarm\_optimization(particles)$
                                                          ▷ Result selection
for $p \in particles$ do
    $p.score \leftarrow evaluate\_energy\_and\_f1score(p.features)$
end for
$sorted\_particle\_sets \leftarrow nondominated\_sort(particles)$
$pareto\_front \leftarrow list(particle.features$ for $particle \in sorted\_particle\_sets[0])$
return $pareto\_front$
```

4.5. Datasets

Table 6: Datasets used.

	PAMAP2 Dataset	**HAR Dataset**	**SPHERE Challenge Dataset**
Sampling rate	100 Hz	50 Hz	20 Hz
Number of activities	12	6	3
Number of windows	15 140	10 299	1160
Duration	5.4 h	7.3 h	2.1 h
Wearable position used	wrist	waist	wrist

The *PAMAP2 Dataset* [35] contains data of multiple physical activities performed by 9 subjects wearing 3 inertial measurement units (over the wrist on the dominant arm, on the chest, and on the dominant side's ankle) and a heart rate monitor. In this paper, we use the data of their 12 "protocol" activities: lying, sitting, standing, ironing, vacuum cleaning, ascending stairs, descending stairs, walking, Nordic walking, running, and rope jumping. Data were sampled at 100 Hz in this work and we use only the accelerometer data, although magnetometer and gyroscope data are also available.

The *UCI HAR Dataset* [36] was collected by attaching a smart-phone (with accelerometer and gyroscope) in a waist-mounted holder, with 30 participants conducting 6 activities in a controlled laboratory environment. Six activities were annotated in this dataset: walking, walking up stairs, walking down stairs, sitting, standing, and lying down. The acceleration was sampled at 50 Hz on triaxial accelerometers and gyroscopes. Since gyroscopes can consume several orders of magnitude more power than accelerometers, we only assess the accelerometer data in our treatment of this work.

The *SPHERE Challenge Dataset* [37] contains synchronized accelerometer, environmental and video data that was recorded in a smart home by the SPHERE project [38, 7, 39]. Three sensing modalities were collected in this dataset: 1) environmental sensor data; 2) accelerometer and Received Signal Strength Indication data; and 3) video and depth data. Accompanying these data are annotations on location within the smart home, as well as annotations relating to the Activities of Daily Living that were being performed at the time. In this work we consider only the acceleration data. Twenty activities were annotated in this dataset, and 10 participants participated volunteered for the challenge totaling approximately 9 hours of data. In order to avoid having to deal with missing data in this paper, we use a subset of the dataset: the activities of six participants, each of which has $< 5\,\%$ of samples missing because of lost over-the-air packets, and quantize the readings as 8-bit integers. Only three activities from this subset have sufficient amounts of data (>100 windows each), so we only use those three.

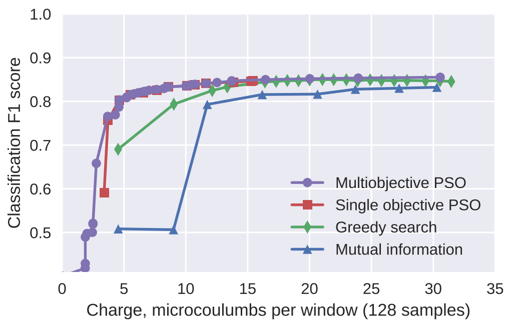
4.6. Feature Group Selection Algorithm

The feature group selection is done for each dataset independently using this process:

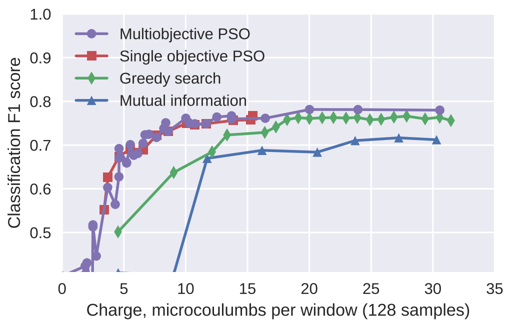
1. The raw data in the dataset is preprocessed: segmented in 128-sample windows (50 % overlap).
2. To each of the segments, one activity value is assigned. If at least 2/3 of entries in that segment have a single activity the value is set to the dominant activity code during that segment; it is set to -1 otherwise.
3. All features are calculated for each window.
4. The features of a randomly selected subject are removed from the dataset.
5. Each feature selection algorithm is run using the features from the main dataset as inputs and F_1 scores from three-fold cross validation as the performance metric.
6. The performance on the subject-left-out is separately measured for each feature group. It is reported to show the generalizability of the results.

5. Results

The results (Figs. 6, 7, 8) show the expected shape of the approximate Pareto-optimal fronts. When the charge consumption is very low, increasing it just slightly leads to massive accuracy gains. Then the curve has an inflection point, and the opposite becomes true: there is just a slight increase in accuracy when new or more costly features are added.

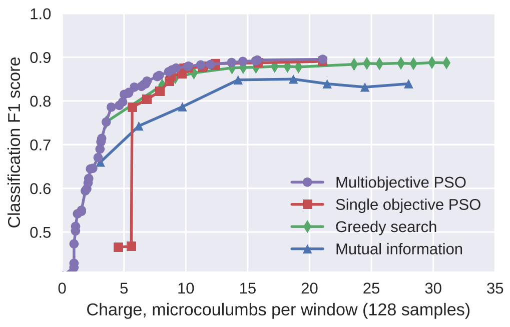


(a) Main dataset

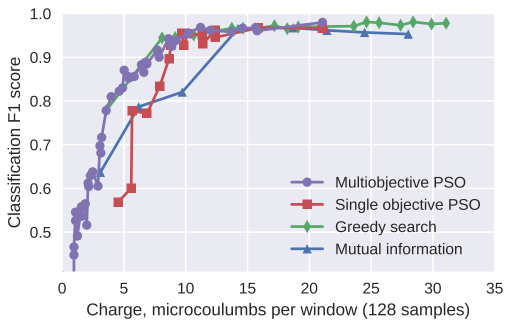


(b) Subject left out

Figure 6: Approximate Pareto-optimal fronts on the PAMAP2 dataset.

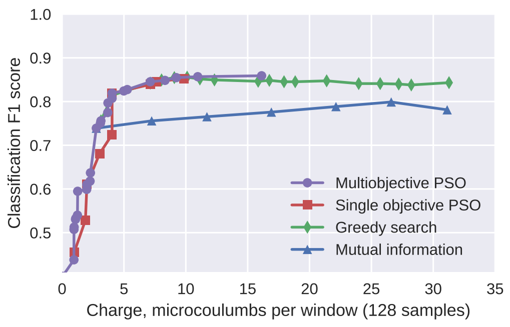


(a) Main dataset

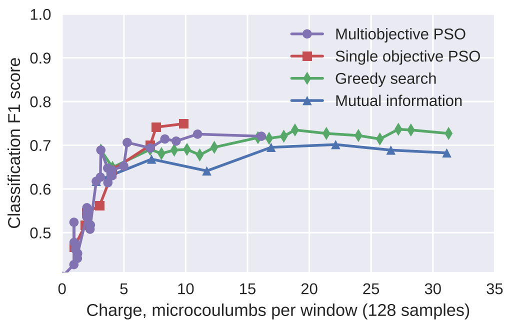


(b) Subject left out

Figure 7: Approximate Pareto-optimal fronts on the HAR dataset.



(a) Main dataset



(b) Subject left out

Figure 8: Approximate Pareto-optimal fronts on the SPHERE dataset.

5.1. PSO Based Search

The PSO methods show the best overall energy-accuracy tradeoff. The multi-objective shows slightly better results. However, its main benefit is that it obtains a higher number of solutions. The multi-objective PSO algorithm avoids crowding of particles, and as a result, it produces a Pareto-optimal front with higher granularity. The number of solutions it is consistently higher compared to the single objective PSO algorithm.

5.2. Greedy Search

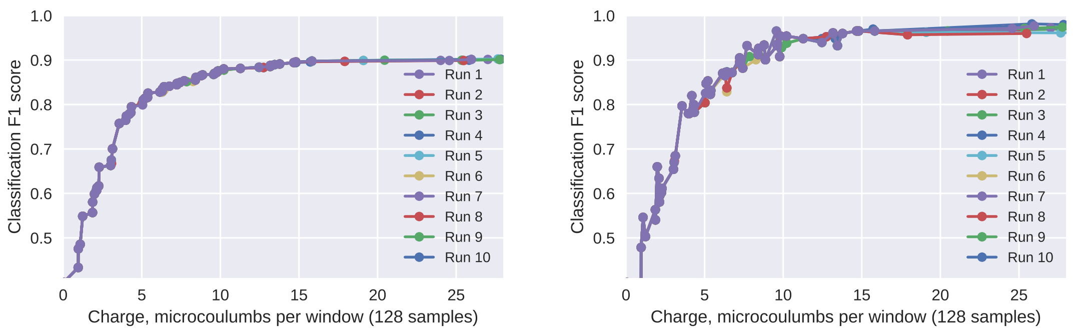
The greedy search finds feature groups that are generally dominated by groups found by the PSO methods. Especially if saving energy is the main concern, the greedy search is not competitive. By its nature, the granularity of the results is low, since each iteration of the algorithm adds a new feature to the candidate set. However, the greedy search is faster to execute than the PSO methods.

5.3. MI Based Selection

This method performs significantly worse than the others. This is explained as it is the only one that does not consider the energy cost in the selection process, and that it ignores the redundancy between different high-ranking features. Untypically, this method performs better on the test data than on validation data, for PAMAP2 and SPHERE datasets: unlike the other methods, this method does not fit the selected features to the validation set.

5.4. Dataset Specifics

The PAMAP2 Dataset shows good match between the main dataset and the subject left out, and is the one that most benefits from the PSO methods. For the other datasets, the shape of the solution graph for the subject left out is slightly more different than the shape of the graph on the main portion of that dataset. The results on the SPHERE Challenge Dataset (Fig. 8) in particular are more affected by randomness, as it has fewer samples: it is an order of magnitude smaller than the other two datasets (Table 6).



(a) Main dataset (b) Subject left out

Figure 9: Results from repeated PSO multi-objective optimizations on the HAR dataset.

5.5. Repeatability

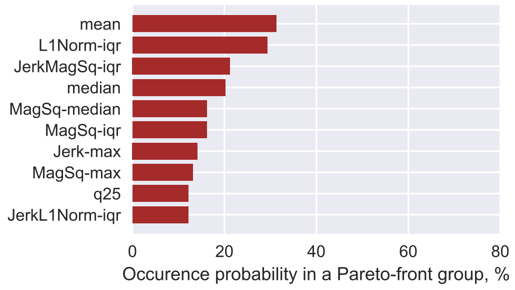
To investigate the repeatability of algorithms we select the best algorithm (PSO, multi-objective version) and run it on the HAR dataset 10 times. The results (Fig. 9) show that the initial selection of energy-efficient feature groups shows perfect repeatability, while high accuracy can be obtained in multiple different ways, so different groups are selected in the different runs. The results on the subject left out set show increased variability compared to the validation set, as the optimization process operates with the latter.

5.6. The Performance of Individual Features

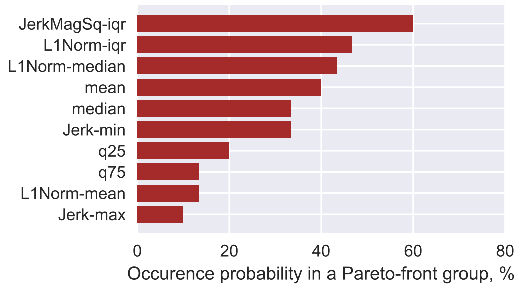
Figures 10 and 11 show the most frequently occurring individual features. These figures exclude the results from the MI based search, as they were generally much worse than the other methods and did not take into account the energy cost.

The results show that there are no universally good features: no single feature shows up in all six different graphs. Each activity recognition application benefits from slightly different features. Furthermore, many of the features have high correlations with other features, therefore can be replaced with the other features at least for some of the applications. (It is worth noting that redundancy or very high correlation between features does not mean that they are always mutually replaceable [23].)

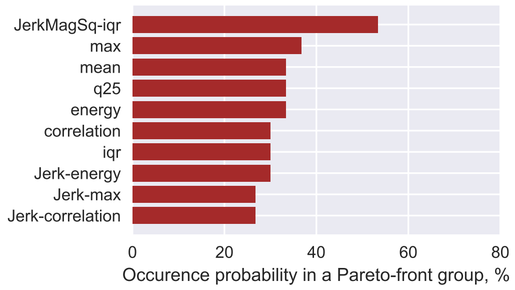
Figure 12 visualizes the frequency and energy consumption of individual features in the results, on all datasets and all algorithms, except the MI based search. *JerkMagSq-iqr* is the only feature that shows up in five out



(a) PSO, multi-objective

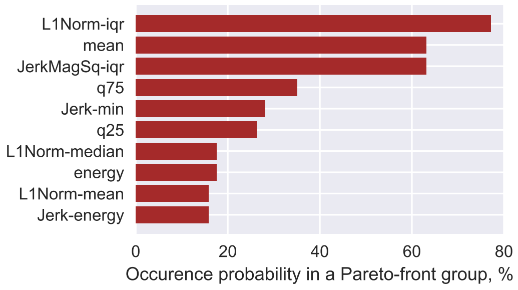


(b) PSO, single objective

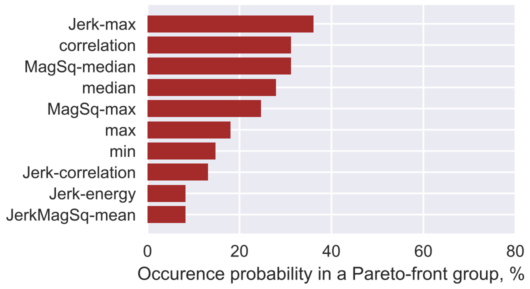


(c) Greedy selection

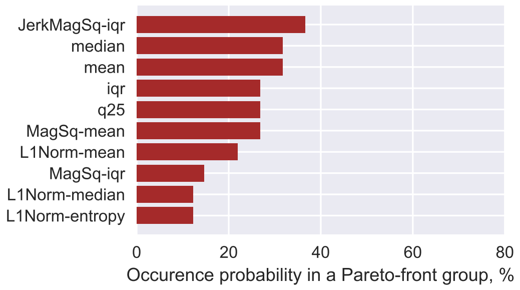
Figure 10: Ten most frequently occurring features, plotted per algorithm.



(a) PAMAP2 dataset



(b) UCI HAR dataset



(c) SPHERE Challenge dataset

Figure 11: Ten most frequently occurring features, plotted per dataset.

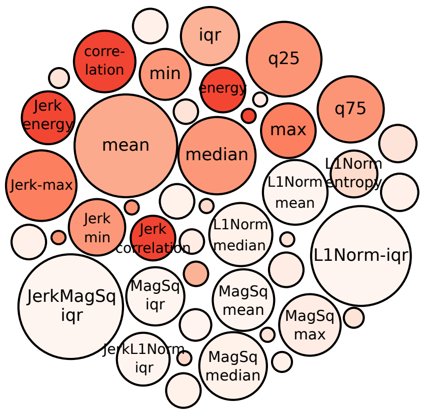


Figure 12: The energy consumption and the selection frequency of individual features. The diameter of the nodes is proportional to their frequencies in the results. The color of a node corresponds to its individual energy consumption (darker color – more energy).

of the six plots. It is likely that the main reason for that is how cheap it is to transmit the results of this feature. However, it would be rather difficult to manually come up with this feature, as it requires two intermediate transforms of the data (first *jerk*, then *magnitude squared*), succeeded by the calculation of both quartiles. We are not aware of any existing research that uses this particular feature. This demonstrates that our generalized approach of combining arbitrary transforms and calculating all candidate features on the result helps to discover novel, useful features.

5.7. Algorithm Runtime Performance

Table 7: Algorithm runtime performance on the SPHERE dataset.

Algorithm	**Runtime, seconds**
Mutual Information	4.6 s
Greedy search	454.2 s
PSO, multi-objective	1924.5 s
PSO, single objective	2413.6 s

The algorithms are envisioned to run offline, on a powerful computer. In Table 7 we provide results on an Lenovo Thinkpad X1 laptop with Intel Core i7-10710U CPU and 16 GB RAM. It can be seen the the mutual information based method is by far the fastest one, while the wrapper search methods incur a significant runtime as they have to train and evaluate RF classifiers on the dataset many times over. The application only uses a single core of the CPU; there is a potential for several-fold improvement if multithreading or GPU were used. The exact performance depends both on the dataset size and the classifier parameters, such as the number of trees in the RF classifier (see Section 4.3).

6. Discussion

6.1. Energy Saved By Using the Feature Extraction

Table 8: F_1 score comparison with and without feature selection.

	PAMAP2 Dataset	**HAR Dataset**	**SPHERE Challenge Dataset**
F_1 score, best feature group	0.855	0.895	0.859
F_1 score, all features	0.854	0.833	0.820
Best F_1 score at $\leq 9.4\,\mu C$	0.833	0.875	0.855

Table 9: Charge consumption comparison with and without feature selection.

	PAMAP2 Dataset	**HAR Dataset**	**SPHERE Challenge Dataset**
Raw data	$94.38\,\mu C$	$94.38\,\mu C$	$94.38\,\mu C$
At 99 % of max F_1 score	$20.02\,\mu C$	$36.04\,\mu C$	$36.24\,\mu C$
At 95 % of max F_1 score	$8.39\,\mu C$	$25.49\,\mu C$	$36.08\,\mu C$
At 90 % of max F_1 score	$6.55\,\mu C$	$6.18\,\mu C$	$7.128\,\mu C$

Wearable applications frequently collect the full acceleration data [39]. Such an approach provides flexibility later on and is especially important if the initial hypothesis is not clear. However, simply adding more features may not improve the accuracy of the prediction (Table 8). When all features are used inputs to the RF classifier, the performance is worse in 5 cases out of 6 compared with selecting and sending over a group of features.

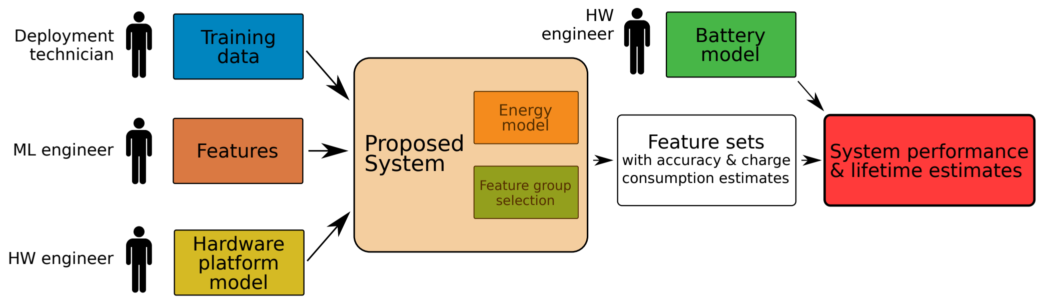


Figure 13: The envisioned application of the proposed system.

Moreover, the raw data transmission has much higher cost compared to extracting and transmitting features. On the target platform, collection raw data for a single window requires $31.46 \times 3 = 94.38\,\mu$C (Table 4). At 10 % of that cost (i.e., at $\leq 9.4\,\mu$C) the accuracy is similar to that obtained from using all features (Table 8). Hence, using the on-board feature extraction reduces the cost tenfold with only a small decrease in accuracy.

6.2. Application Examples

Fig. 13 shows the intended application of this work. The inputs of the proposed system are: labeled training data from a short-term pilot experiment, list of features, and the platform model. The amount of the training data required is not large: in our evaluation it ranges from 2.1 hours for SPHERE to >7 h for HAR (Table 6), although a more detailed activity profile may require more data. The amount of the data has an impact on the result quality (Figs. 6, 7, 8), but even for SPHERE it is acceptable.

The output is the approximate Pareto front of feature groups; it should be used together with a battery model that captures the discharge patterns of the hardware platform's power source (its voltage and capacity dynamics under load). Given both, it is possible to answer questions about the accuracy and longevity of the deployments before actually carrying them out, thus saving time and effort.

Example application 1. *In a smart home project, wearable devices are to be deployed to participants together with recharging instructions. What is the minimum required recharge frequency, given that the system should achieve F_1 score ≥ 0.9 ?* Here, the question can be answered by collecting training data, running the feature group selection, and removing the results with $F_1 < 0.9$. The most efficient remaining feature set can be used, and the charge consumption can be translated to required recharge frequency using a battery model.

Example application 2. *A clinical researcher plans to carry out a 2-week trial with ill elderly people as the wearable users. What is the maximum achievable F_1 score, given that the participants should not be required to recharge the devices?* Here, the charge consumption first must be translated to battery life, and applied as a filter to the results; after that, the highest-scoring feature set provides the answer.

7. Conclusions

This paper proposes a framework for finding groups of features that have approximately optimal energy-accuracy trade-offs for activity recognition from acceleration data. The proposed system helps to answer questions about the expected battery lifetime and recognition accuracy of an activity recognition application without carrying a full-scale labor-intensive deployment. We describe a detailed energy consumption model that takes into account feature inter-dependencies and instantiate this model for an ARM Cortex-M3 based wearable platform. Subsequently, we describe and evaluate a number of feature selection algorithms. Their evaluation using three datasets shows that the multi-objective Particle Swarm Optimization algorithm achieves the best results in terms of the accuracy-energy tradeoff. Extracting and sending the features requires an order of magnitude less energy compared with sending the raw data, while having minimal impact on the F_1 score.

Acknowledgments

This work was supported by the ERDF Activity 1.1.1.2 “Post-doctoral Research Aid” (No. 1.1.1.2/VIAA/2/18/282).

References

[1] J. Sengupta, S. Ruj, S. D. Bit, A Comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT, Journal of Network and Computer Applications 149 (2020) 102481.

[2] W. Kassab, K. A. Darabkh, A–z survey of internet of things: Architectures, protocols, applications, recent advances, future directions and recommendations, Journal of Network and Computer Applications (2020) 102663.

[3] T. McGhin, K.-K. R. Choo, C. Z. Liu, D. He, Blockchain in healthcare applications: Research challenges and opportunities, Journal of Network and Computer Applications (2019).

[4] J. Qi, P. Yang, G. Min, O. Amft, F. Dong, L. Xu, Advanced Internet of Things for personalised healthcare systems: A survey, Pervasive and Mobile Computing 41 (2017) 132–149.

[5] A. Hadjidj, M. Souil, A. Bouabdallah, Y. Challal, H. Owen, Wireless sensor networks for rehabilitation applications: Challenges and opportunities, Journal of Network and Computer Applications 36 (2013) 1–15.

[6] A. Bajpai, V. Jilla, V. N. Tiwari, S. M. Venkatesan, R. Narayanan, Quantifiable fitness tracking using wearable devices, in: Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE, IEEE, pp. 1633–1637.

[7] P. Woznowski, A. Burrows, T. Diethe, et al., SPHERE: A Sensor Platform for Healthcare in a Residential Environment, in: Designing, Developing, and Facilitating Smart Cities: Urban Design to IoT Solutions, Springer International Publishing, 2017, pp. 315–333.

[8] A. Elsts, R. McConville, X. Fafoutis, N. Twomey, R. Piechocki, R. Santos-Rodriguez, I. Craddock, On-board feature extraction from acceleration data for activity recognition, in: Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN).

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

[10] N. Twomey, T. Diethe, X. Fafoutis, A. Elsts, R. McConville, P. Flach, I. Craddock, A comprehensive study of activity recognition using accelerometers, Informatics 5 (2018).

[11] U. Maurer, A. Smailagic, D. P. Siewiorek, M. Deisher, Activity recognition and monitoring using multiple sensors on different body positions, in: International Workshop on Wearable and Implantable Body Sensor Networks (BSN), IEEE.

[12] J. Wang, Y. Chen, S. Hao, X. Peng, L. Hu, Deep learning for sensor-based activity recognition: A survey, Pattern Recognition Letters (2018).

[13] C. Bormann, M. Ersue, A. Keranen, Terminology for Constrained-Node Networks, RFC 7228, IETF, 2014.

[14] R. Possas, S. Pinto Caceres, F. Ramos, Egocentric activity recognition on a budget, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5967–5976.

[15] 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 (2017) 82–88.

[16] M. Satyanarayanan, The emergence of edge computing, Computer 50 (2017) 30–39.

[17] 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, pp. 17–24.

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

[19] Y. Liang, X. Zhou, Z. Yu, B. Guo, Energy-efficient motion related activity recognition on mobile devices for pervasive healthcare, Mobile Networks and Applications 19 (2014) 303–317.

[20] A. Elsts, Source node selection to increase the reliability of sensor networks for building automation, in: EWSN, pp. 125–136.

[21] L. Zheng, D. Wu, X. Ruan, S. Weng, A. Peng, B. Tang, H. Lu, H. Shi, H. Zheng, A novel energy-efficient approach for human activity recognition, Sensors 17 (2017) 2064.

[22] J. Miao, L. Niu, A survey on feature selection, Procedia Computer Science 91 (2016) 919–926.

[23] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of machine learning research 3 (2003) 1157–1182.

[24] J. Kennedy, Particle swarm optimization, in: Encyclopedia of machine learning, Springer, 2011, pp. 760–766.

[25] B. Xue, M. Zhang, W. N. Browne, Particle swarm optimization for feature selection in classification: A multi-objective approach, IEEE transactions on cybernetics 43 (2013) 1656–1671.

[26] N. Srinivas, K. Deb, Muiltiobjective optimization using nondominated sorting in genetic algorithms, Evolutionary computation 2 (1994) 221–248.

[27] 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, pp. 37–42.

[28] C. Emmanouilidis, A. Hunter, J. MacIntyre, A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator, in: Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512), volume 1, IEEE, pp. 309–316.

[29] N. Wang, G. V. Merrett, R. G. Maunder, A. Rogers, Energy and accuracy trade-offs in accelerometry-based activity recognition, in: Computer Communications and Networks (ICCCN), 2013 22nd International Conference on, IEEE, pp. 1–6.

[30] D. Chu, N. D. Lane, T. T.-T. Lai, C. Pang, X. Meng, Q. Guo, F. Li, F. Zhao, Balancing energy, latency and accuracy for mobile sensor data classification, in: Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems, ACM, pp. 54–67.

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

[32] X. Fafoutis, A. Vafeas, B. Janko, R. S. Sherratt, J. Pope, A. Elsts, E. Mellios, G. Hilton, G. Oikonomou, R. Piechocki, I. Craddock, Designing Wearable Sensing Platforms for Healthcare in a Residential En-

vironment, EAI Endorsed Trans. Pervasive Health and Technology 17 (2017).

[33] C. Bormann, P. Hoffman, Concise Binary Object Representation (CBOR), RFC 7049, IETF, 2013.

[34] J. Kennedy, Particle swarm optimization, in: Encyclopedia of machine learning, Springer, 2011, pp. 760–766.

[35] A. Reiss, D. Stricker, Creating and benchmarking a new dataset for physical activity monitoring, in: Proc. of the 5th Int. Conf. on PErvasive Technologies Related to Assistive Environments, ACM, 2012, pp. 40:1–40:8.

[36] D. Anguita, et al., A public domain dataset for human activity recognition using smartphones, in: European Symp. on Artificial Neural Networks, Computational Intell. and Mach. Learning (ESANN).

[37] N. Twomey, T. Diethe, M. Kull, H. Song, M. Camplani, S. Hannuna, X. Fafoutis, et al., The SPHERE challenge: Activity recognition with multimodal sensor data, arXiv preprint arXiv:1603.00797 (2016).

[38] N. Zhu, T. Diethe, M. Camplani, L. Tao, A. Burrows, N. Twomey, D. Kaleshi, M. Mirmehdi, P. Flach, I. Craddock, Bridging e-Health and the Internet of Things: The SPHERE project, IEEE Intelligent Systems 30 (2015) 39–46.

[39] A. Elsts, X. Fafoutis, P. Woznowski, E. Tonkin, G. Oikonomou, R. Piechocki, I. Craddock, Enabling healthcare in smart homes: The sphere iot network infrastructure, IEEE Communications Magazine 56 (2018) 164–170.