An Embedded System for NILM Using Machine Learning

Giovanni Braglia
UNIMORE/UTFPR
Modena, Italy
Email: giovanni.braglia97@gmail.com

André Eugenio Lazzaretti
UTFPR
Curitiba, Brazil
Email: lazzaretti@utfpr.edu.br

***Abstract*—The interest in power managing systems has been growing in recent years since every industrial or domestic plant moves towards techniques to efficiently reduce energy demand and costs related to it. An attractive solution is represented by Non-Intrusive Load Monitoring (NILM) systems, whose primary purpose is to find a more appropriate way of keeping track of the power consumption caused by each of the loads that are connected to the monitored plant. A possible real-life implementation of a NILM system is addressed in this work, discussing all the fundamental blocks in its structure, including detecting events, feature extraction, and load classification, using publicly available datasets. Additionally, we provide a solution for an embedded system, able to analyze aggregated waveforms and to recognize each appliance's contribution in it. The main algorithm, its features, drawbacks, and implementation are thus explained, showing current and future challenges for the final application.**

I. Introduction

In the next decades, increasing energy demand will claim not just for new sources but also for intelligent and efficient ways of managing consumption. Nowadays, solutions are moving towards smart grids techniques that, due to the mutual exchange of information among their users, allow a better distribution and management of electric power.

Non-Intrusive Load Monitoring (NILM) [1] is a power managing solution that aims at analyzing a system's current and voltage signals, allowing to recognize which appliances are used and their respective power consumption. From a practical point of view, this means developing a plug-and-play device that can recognize loads connected to an electrical system and display the amount of individual energy used [1]. In particular, non-intrusive means that loads' data are gathered from a single device instead of multiple sensors placed on individual appliances, which might intrude on the monitored system.

According to [2], [3], NILM approaches usually consider the following steps: (i) Data Acquisition; (ii) Feature Extraction; and (iii) Load Identification. Data acquisition has a critical role in the next steps. A high sampling frequency system, for example, might directly impact the signatures of the signals that have to be analyzed, such as their resolution or harmonic components. On the other hand, a low sampling frequency leads to more affordable costs, thus a cheaper device. During the process of feature extraction, the disaggregated signal referring to a single appliance is analyzed. The purpose of this step is to calculate a set of features coherent with data acquisition that will be exploited in the classification process needed for the load identification.

Several strategies could be applied to identify the loads that are consuming energy [4]. In particular, in this work, supervised machine learning algorithms are used to recognize loads through classifiers, which are models trained on our specific data. Many related works [5]–[7] use machine learning algorithms because of their robustness and generalization performance over new data. On the other hand, machine learning algorithms require high computational capabilities. Therefore some architectures might be preferable due to their intrinsic parallelism [8].

In this sense, this work aims to develop an embedded system to perform NILM with high-sampled voltage and current signals. To do so, a detection algorithm was implemented to identify and disaggregate each appliance; next, a specific set of features were extrapolated from the target signals and suddenly sent to a classifier for their recognition, where four different classifiers were analyzed and compared. Moreover, the whole algorithm was implemented over an embedded system, for which execution times were recorded. Thus, we believe that the results presented in this work form a step towards a possible real-life implementation of a NILM device.

This paper is organized as follows. Section II focuses on the main techniques used for this work, including a small overview of the state-of-the-art solutions. Section III resumes the theoretical aspects worthy of mention. Section IV explains the proposed method elaborated for the development of the algorithm behind a NILM system. Finally, Section V exposes and discusses the results obtained, and Section VI concludes this work.

II. Related Work

In this work, NILM was mainly divided into three procedures, i.e., data acquisition, feature extraction, and load identification, leading to three fundamental blocks: load detection, features extraction and classification.

Detect a load properly meant, in this project, recognize that a turn-on event occurs and then use a disaggregation algorithm to isolate the selected load for its further analysis. In [9], for instance, an algorithm called High Accuracy NILM Detector (HAND) is proposed. This algorithm tracks the envelope of a signal and computes its variance. Then, the use of an empirical threshold exploits the higher variance values to find out if an event occurred. Similarly, the Half-Cycle Apparent Power (HCApP) algorithm [10] analyzes the apparent power signal, moving to a binary representation of it, and determines a transient window for both the turn-on and turn-off events based on power variations . Still in this context, Discrete Wavelet Transform (DWT) performs signal segmentation using wavelet decomposition [11], allowing the calculation of an universal threshold, instead of an empirical one. Even a Kalman Filter could be used for detection, as presented in [12]. In this case, from a comparison between the real and the estimated signal, the higher number of errors in the latter is usually related to the signal regions where more changes occur, which are usually transients. One last detection approach uses vectorization [13], [14] to characterize the samples of a signal into ”valid” and ”non-valid” states. Non-valid states are associated with events such as abrupt transitions, high derivatives values or high-frequency oscillations, which resemble the turn-on and off conditions that we are looking for.

After detection, the isolated signal must be processed to extract the relevant features for classification; in particular, features are based on load characteristics, whose information content might be influenced by the respective sampling frequency. Steady-state methods, for example, identify devices based on variations in their steady-state signatures. Different steady-state features are already present in the literature and enable low-cost hardware because of low-sampling rate requirements. On the other hand, some limitations occur when many loads show similar steady-state shape [15]. Moreover, those methods slow the monitoring process since they require waiting until transient states settle down. To provide a further mean of discrimination, transient state methods for feature extraction can be introduced [16], since transient behavior is typically related to the physical task that the load performs. It is also advantageous to identify variable drive-connected loads since the drive startup is generally repeatable. However, one of the major drawbacks is that a high sampling rate is required which translates in a more expensive hardware [17].

The last step in the sequence is classification. This process aims to analyze the features collected and recognize the load that has been detected. The most widely used approach is supervised learning, using pattern recognition or optimization-based algorithms. Supervised learning requires labeled datasets for training classifier model, to identify and generalize over data of the same nature (voltage and current signals in this case) [18]. Optimization methods perform load recognition using a heuristic approach, thus comparing the extracted feature vector of an unknown load with the ones present in the appliance database and minimizing an error function to find the closest possible match [5]. On the other hand, pattern recognition methods exploit database features to cluster appliances based on their characteristics [19]. Even though supervised learning can be considered more reliable, one limitation is its strong dependency on the datasets used for training the model, which implies a considerable effort to build datasets with new appliances, requiring years of data collection. This is why recently, researchers have shown an increased interest in unsupervised learning techniques [20]. Those methods try to isolate the aggregated load measurements without evaluating previously collected information, but clustering data based on repetitive patterns discovered by the model itself. Good results (accuracies $> 90\%$) have been reached by the research, suggesting that unsupervised learning might be a good solution in NILM [21]. However, some very non-trivial problems need to be solved. For example, many clusters are formed in a multi-state appliance due to multiple states, therefore confusing the classifier model. One solution might be use deep learning techniques [22]. For example, several neural networks can be trained in cascade and used as pattern classifiers to identify the various loads. Each network classifies the load’s family at a specific level. Those algorithms are compelling, but their computational complexity makes them prone to fit some specific hardware (such as GPU) rather than others.

As a matter of fact, the embedded system that will host the algorithm for a NILM device should be appropriately chosen according to the tasks required, cost of the device and power consumption constraints. Ideally, a NILM device is a multitasking system, able to analyze multiple loads simultaneously, exploiting parallel computing to boost performances in terms of speed and optimization of power consumption. This might be achieved using GPUs or FPGAs, inevitably affecting the costs. The work presented in this paper tries to focus on the performances rather than costs, to increase calculus accuracy and system capacity, therefore using a powerful embedded system to host the algorithm for a NILM module. This is small part of the project presented in [5], [23]–[25], where the intentions are to use several load monitoring modules units controlled by an Operation Center, whose task is managing each unit and analyze power-consuming from the data collected.

III. Theoretical Aspects

In this section, we present the theory behind the most important building blocks that were included in the realization of a NILM device. As mentioned in the section before, loads’ signals were analyzed employing a detection algorithm, features extrapolation, and classifier identification processes.

A. Detection Method

Detection alerts the system that a load switched on, isolating the appliance from an aggregate power signal and manipulating it later on. In particular, every time a new transient occurs, this will be reported to the system through the start and the end instants of the transient itself.

Among the algorithms mentioned in the section above, in this work, the HAND algorithm [9] was applied as detection strategy. The HAND’s working principle is simple and intuitive: the variation of the standard deviation $\sigma(t)$ of the signal current’s envelope $e(t)$ is tracked using a moving window. A threshold separates the events characterized by high $\sigma(t)$ amplitude variations and the one with low variations, thus

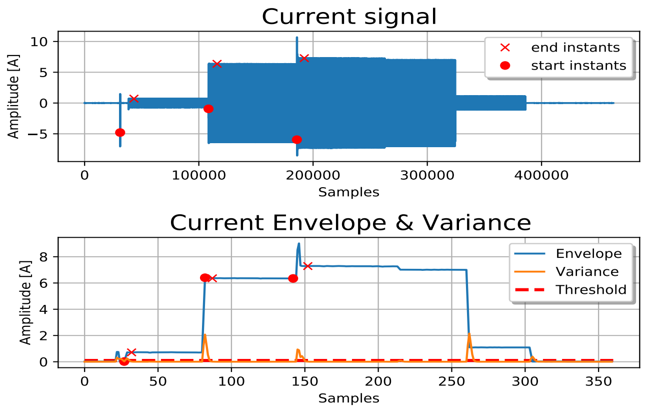


Fig. 1: t_s and t_e detection for a composed signal (Led Pannel, Oil Heater Power 1 and Incandescent Lamp) in LIT Dataset.

allowing to classify among transient and steady-state intervals. The algorithm is summarized in Algorithm 1.

Algorithm 1 HAND Algorithm

```
Input: signal, moving window size (L), threshold in
Output: start ($t_s$) and end ($t_e$) instants of transients out
 1: Compute signal envelope $e(t)$
 2: Initialize the first L instants of the standard deviation $\sigma(t)$ of $e(t)$:
    $\sigma(t_i) = std[e(t_0), e(t_1), ..., e(t_L)]$, i = 0, 1, ... , L
 3: Initialize the first L instants of the mean $\mu(t)$ of $e(t)$:
    $\mu(t_i) = avg[e(t_0), e(t_1), ..., e(t_L)]$, i = 0, 1, ... , L
 4: $j \leftarrow L$
 5: while $j \leq length(signal)$ do
 6:     Compute mean as $\mu(t_j) = \mu(t_{j-1}) + \frac{1}{L}[e(t_j) - e(t_{j-L})]$
 7:     Compute variance as $\sigma^2(t_j) = \frac{1}{L^2}\sigma^2(t_{j-1}) + \frac{1}{L-1}[e(t_j) - \mu(t_j)]^2$
 8: end while
 9: $k \leftarrow 0$
10: while $k \leq length(signal)$ do
11:     if ( $\sigma(t_k) > threshold$ and $\frac{d\sigma(t_k)}{dt_k} > 0$ and $\frac{de(t_k)}{dt_k} > 0$) then
12:         $t_k$ is a start instant $t_s$
13:     end if
14:     if ($\sigma(t_k) < threshold/3$ and $\frac{d\sigma(t_k)}{dt_k} < 0$) then
15:         $t_k$ is an end instant $t_e$
16:     end if
17: end while
18: return $t_s$,$t_e$
```

In this work, it was assumed that each appliance is turned on just when the previous one's transient is already finished. This assumption is coherent with the signals of the used datasets [2], [23]. Moreover, to avoid false detection due to the possible presence of noise, $e(t)$ has been smoothed with a moving average filter when HAND was implemented.

Fig.1 shows some results from the detection algorithm. Note that it is just needed to detect turn-on events for our purposes; thus, in the HAND algorithm, a condition over the envelope's gradient was introduced.

B. Feature Extraction

Both COOLL [2] and LIT [23] datasets provide current and voltage signals of appliances that are mainly household devices. For the two datasets, the same features were extrapolated from the signals and collected in two separate databases. As a consequence of the detection algorithm, it was possible to calculate the features for both the steady and the transient state of a load signal in two different time windows, respectively.

Active, reactive, and apparent power were first extrapolated since the signals analyzed are voltages and currents signals. It was also possible to provide further information to the classifiers using other more standard features like average value, covariance, and the harmonics' amplitude. Moreover, the nature of the processed signals makes possible the use of V-I trajectories features for steady-state intervals, where one cycle of both current and voltage signals were extrapolated [6]. For those features, an exhaustive review was presented in [5] where, among the commonly used ones, a new set of features was proposed to increase the number of signatures for the classification process. In this work, the used features were: peak value, active/reactive/apparent power, median, covariance of voltage (V) and current (I) signals, variance, steady state average value, harmonics up to the 12^{th} order, current span, V-I trajectory area, and area with loop direction. All of them are listed and explained at the public available repository located at https://github.com/giobraglia/Signal-Processing-Features.

C. Classifiers

Machine learning techniques were implemented in this project, to make the system autonomous and able to generalize recognition over every kind of appliance connected to the monitored system. Two are the macro-categories when we refer to machine learning [26]. *Supervised learning* aims at learning a model from labeled training data that allows us to make predictions about unseen future data. On the other hand, *unsupervised learning* usually tries to cluster data, approaching the classification problem without class labels.

In this project, we used a supervised learning approach since all the loads were already provided with an identification label. In particular, four different machine learning algorithms were used and evaluated: Logistic Regression (LR), Support Vector Machine (SVM), K-nearest-neighbors (KNN), and Random Forest classifier (RF).

IV. Proposed Method

Fig. 2 explains the process of our NILM procedure. The lower branch represents the steps required to properly select a classifier model, starting from storing all the features in specific databases used for the training process of classifiers. In this last passage, performances can be increased using model evaluation techniques [26]. This method is fundamental to tune the classifiers properly and select one of them for a later implementation on the embedded system. On the other hand, the upper branch shows the fundamental blocks for online load identification. Here, the system operates over a fixed time window, where it observes if some turn-on events occur. Thus, the loads detected are disaggregated to reduce the influence from any other appliance components [5], [15]. Once the signal is isolated, the system calculates the respective features and sends them to the classifier in order to receive, as output, the load identification label. In the following paragraphs, this method is detailed.

A. Datasets

The two datasets used for this work are the COOLL [2] and LIT Dataset [23]. Both datasets are considered high sampled ones, as the signals are provided with a sampling frequency of 100 kHz and 15 kHz, respectively. COOLL waveforms all refer to individual appliances. Hence, the disaggregation process was not needed. On the other hand, LIT dataset provides aggregated waveforms, which recorded multiple loads that were turned on in sequence. This characteristic inevitably requires the use of an isolation (disaggregation) algorithm. Moreover, LIT dataset includes in its waveforms the instants of every on/off events, a useful feature for evaluating detector performances. In particular, the LIT dataset is splitted into three groups: synthetic, simulated, and natural. For this project, the synthetic dataset was used, where all the appliances were measured in a more controlled way, with a precise load shaping. Table I and II detail the appliances used in the COOLL and LIT datasets, respectively.

TABLE I: COOLL Dataset.

Appliances	**Num. of Appliances**	**Num. of Signals**
Drill	6	120
Fan	2	40
Grinder	2	40
Hair dryer	4	80
Hedge trimmer	3	60
Lamp	4	80
Paint stripper	1	20
Planer	1	20
Router	1	20
Sander	3	60
Saw	8	160
Vacuum cleaner	7	140
Total	42	840

B. Detection Method

This work implemented the HAND as a detection algorithm. In particular, in [9], the detector uses a threshold settled empirically, i.e., a static threshold. In our case, a dynamical adaptive threshold was used employing two different approaches. The first one calculates the threshold averaging the value of the variance signal's local peaks. Alternatively, the second one considers the average of the Fourier transform of the variance signal. After detection is performed successfully, the appliance needs to be adequately isolated with a disaggregation algorithm [15], [27]. As shown in Fig. 3, this algorithm supposes that aggregated waveforms can be seen as the linear combination of the loads that are on. Consequently, the load is isolated by subtracting to it a previous interval of the aggregated waveform, in order filter out previous loads components.

C. Training Procedure

Supervised learning processes expects a dataset for the training of classifiers [26], therefore all the waveforms in [2] and [23] were processed and analyzed.

TABLE II: Appliances in LIT Synthetic Subset (LIT-SYN).

ID	**Device**	**Power (W)**
1	Microwave Standby	4.5
2	LED Lamp	6
3	CRT Monitor	10
4	LED Panel	13
5	Fume Extractor	23
6	LED Monitor	26
7	Cell phone charger Asus	38
8	Soldering station	40
9	Cell phone charger Motorola	-
10	Laptop Lenovo	70
11	Fan	80
12	Resistor	80
13	Laptop Vaio	90
14	Incandescent Lamp	100
15	Drill Speed. 1	165
16	Drill Speed. 2	350
17	Oil heater power 1	520
18	Oil heater power 2	750
19	Microwave On	950
20	Air heater Nilko	1 120
21	Hair dryer Eleganza - Speed 1	365
22	Hair dryer Eleganza - Speed 2	500
23	Hair dryer Super 4.0 - Speed 1 - Heater 1	660
24	Hair dryer Super 4.0 - Speed 1 - Heater 2	1 120
25	Hair dryer Parlux - Speed 1 - Heater 1	660
26	Hair dryer Parlux - Speed. 2 - Heater 1	885

In this work, accuracy, learning curves and ROC curves were used as classification metrics. Moreover, the final classifier has undergone few evaluation steps before being selected. First, the Grid Search technique for parameter tuning was applied. In the sequence, Sequential Backward Selection (SBS) helped in finding all those features that lower classifier's performance, which translates in a dimensional reduction of the features dataset. Then, Grid Search was applied again after the models were trained over a smaller dataset. Finally, Principal Component Analysis (PCA) was used to see if we can provide any additional information to the classifier through feature extraction.

D. Feature Selection and Extraction

As mentioned above, since a predefined set of features was exploited, we should find a way to extrapolate all the helpful information they contain. However, there might even be some unwanted information that we must cleanout. SBS is an algorithm for feature selection that iteratively selects a subset of features and evaluates a classifier's performance over it. The result is that we can easily observe if some features are affecting the quality of identification. Moreover, a further benefit is that by eliminating features, the dataset size is reduced as well as the amount of data that has to be processed.

Another possible technique is PCA. This algorithm provides a way to find the principal components in terms of variance in the original dataset and, therefore, it is a transformation of the features. Consequently, the computational cost is increased if PCA is applied, but this enables the analysis on both dataset and classifiers, possibly adding new information to them.

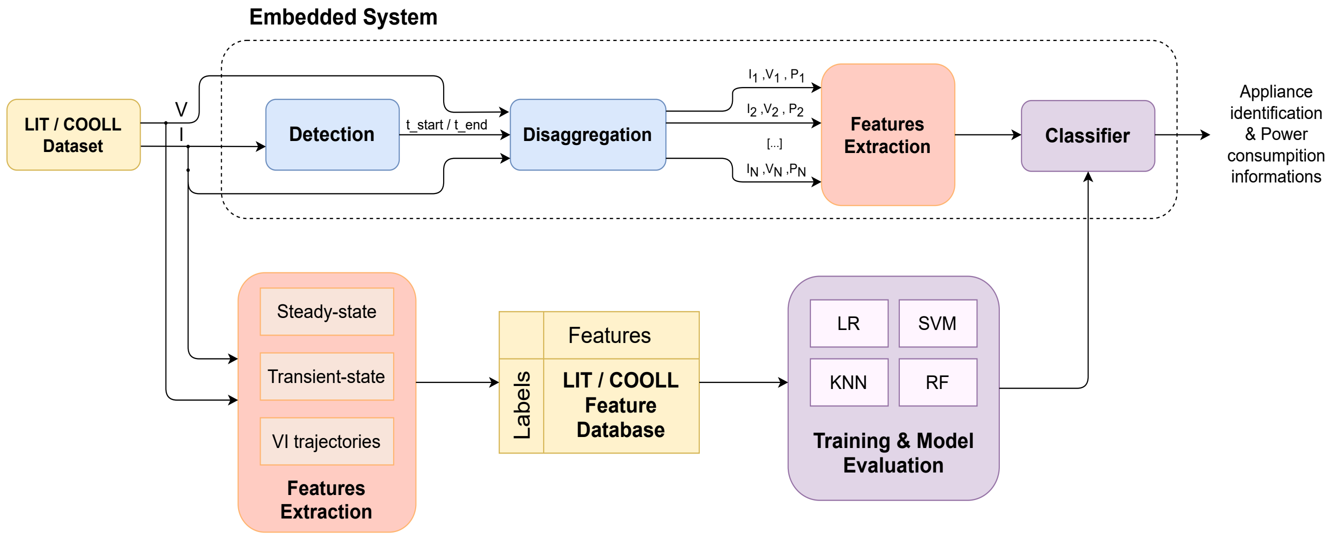


Fig. 2: Project's workflow. Starting from COOLL [2] and LIT [23] Dataset, all the signals features have been extracted and collected in separates databases. Then, four classifiers are trained and selected after an evaluation process. The classifier's model with the best performances was chosen and included in an embedded system. The latter first detect the transients of loads and then isolates, through disaggregation, each appliance for the signal processing. In the last step, the features are fed into the classifier.

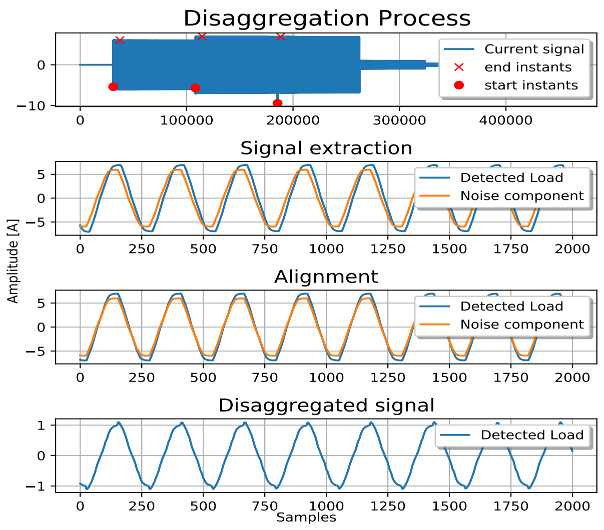


Fig. 3: Details of the disaggregation algorithm. After detecting the selected load, it stores an interval of the same length of the load signal right before the detected part. This component is then subtracted to the signal of interest in order to get rid of other loads' components.

E. Classification metrics

The classifiers' prediction were evaluated in terms of accuracy, which basically tells how many true positives and true negatives events occurred among all the classifications. However, accuracy value might be secretly affected by some unbalances in the data. If, for instance, an appliance on the dataset is always misclassified while the others no, the result is that an high accuracy is achieved although one device is never recognized. That's why Receiver Operating Characteristics (ROC [28]) curves supported the performances analysis in this work, as they investigate over true positive and false positive rate.

Another issue, is that the performances are directly affected even by the dimension of the training dataset: a low number of samples provides just few information to classifiers, while an high one risks that we end up over-fitting the data. For this purpose, learning curves track classifiers' accuracy over training dataset with different dimensions, so it can be chose correctly.

F. Embedded System

The project was conceived to be implemented on an embedded system. Therefore, the NVIDIA Jetson TX1 was chosen as an embedded system for hosting all the codes. NVIDIA Jetson TX1 is a full-featured development platform for visual computing; it is ideal for applications requiring high computational performance in a low power envelope (the whole board consumes 10 W).

Jetson TX1 comes with a Linux environment, includes support for many common APIs, such as python's API, used throughout this work. The board integrates 256-core Maxwell GPU, where Maxwell is the codename for a GPU microarchitecture developed by NVIDIA. The GPU is the main reason why the board has been chosen. Despite CPU always runs codes in sequence, GPU's main feature is the ability to parallelize codes, optimizing computation costs and increasing speed. Therefore, the use of GPU could enhance our system performance a lot, allowing us to analyze more signals simultaneously, thus increasing the capacity of our system, being able to monitor larger environments.

V. Results and Discussion

In this section the results are presented. During the real application of the whole algorithm, some problems related to the detection method appeared and will be discussed in the next paragraphs. Next, the final performances of the classifiers are reviewed before providing some informations about the whole algorithm implementation.

A. Detection

In COOLL dataset, since just single appliance waveforms are available, we tried to extrapolate a reduced set of features without the distinction between steady and transient states. Therefore, the signals were always analyzed in their entire duration instead of separate intervals of them. In this first analysis, COOLL dataset was useful to test and improve the whole algorithm chain, before including the detection and disaggregation blocks (as shown in Fig. 2). This was not possible for the LIT dataset, where a detection algorithm for aggregated waveforms is needed. For instance, in the case of waveforms related to eight simultaneous appliances, empirical results show that the HAND algorithm correctly detected just the 61% of them. As a matter of fact, during simulations three main problems were highlighted:

1) the setting of an adaptive threshold could avoid the detector to sense appliances operating at very low amplitudes;
2) the appliances whose transient swings many times before reaching the steady-state will be detected multiple times. This can result in different features values, leading to wrong predictions;
3) since variance is sensitive to signal variations, the presence of noise can cause false positive events detection.

While to avoid 3) we can simply smooth the signals before their analysis, the other two problems were difficult to be solved since they derived directly from the HAND operation criteria. Therefore, to build the LIT features dataset correctly, the turn on and off instants recorded in [23] were used and applied to the detector. This ensures that every load in a waveform was recognized.

B. Classification

The classifiers' performance, displayed in terms of accuracy, are shown in Table III, IV and V.

Particularly, Table III refers to the analysis over COOLL dataset waveforms. In this case, since all the signals referred to individual appliances, no disaggregation was needed, thus no error due to that process was introduced. This significantly influenced all the classifiers' performances, leading to promising results (accuracy $> 97\%$). Moreover, note that after SBS, the feature dimension was reduced from 30 to 17 features. This result indicates that there were redundant information that could be removed among the features calculated for the COOLL dataset.

TABLE III: COOLL Dataset Test Accuracies.

Classifier	**Preliminary test**	**Grid Search**	**SBS**	**Grid Search**
LR	99.4 (26)	99.4	**99.4 (17)**	99.4
SVM	99.4 (26)	99.4	**99.4 (17)**	99.4
KNN	98.2 (26)	98.2	**98.8 (17)**	98.8
RF	99.4 (26)	99.4	**99.4 (17)**	99.4

Table IV and V refer, instead, to the results obtained for the LIT dataset. Table IV displays classifiers' results when the group of waveforms, based on the number of simultaneous appliances recorded (APP in the Table, from 1 to 8) were separated. This was done to investigate the errors introduced by the detection and disaggregation processes. The classifiers' performances decrease with a higher number of loads, which clearly lower the detection accuracy. This is probably due to the fact that simultaneous loads can increase the amount of noise in the line leading to wrong detection. Moreover, devices with high power requirements could end up fading signals coming from appliances consuming less, which will be distorted or not recognized at all.

To conclude, the same evaluation process of the COOLL dataset was conducted in Table V. Besides, even the PCA technique was used, while for the COOLL dataset this was not useful because, after the result obtained, PCA would only increase the computational complexity. Notice that even in this case, the SBS algorithm was able to spot and remove the features that were actually decreasing the performances. Even though not all the classifiers share the same subset of features after SBS, as suggested in [27] we observed that a lower number of harmonics (up to the 5^{th} order) was enough, since higher order ones usually displayed similar amplitudes values. On the other hand, since we didn't want to get rid of those features completely, it was decided to group higher harmonics information to calculate total odd and even harmonic distortion.

TABLE IV: LIT Dataset Test Accuracies for Different Number of Appliances.

Classifier	**1 APP**	**2 APP**	**3 APP**	**8 APP**
LR	97.6	98.5	94.3	90.3
SVM	97.6	98.9	98.4	90.9
KNN	95.2	98.9	96.5	91.6
RF	98.8	99.3	97.5	96.8

TABLE V: LIT Dataset Test Accuracies with Dimensionality Reduction.

Classifier	**Preliminary test**	**Grid Search**	**SBS**	**Grid Search**	**PCA**
LR	88.2 (30)	88.7	87.4 (23)	87.4	**94.3**
SVM	96.2 (30)	96.2	97.0 (28)	95.4	**97.3**
KNN	94.5 (30)	94.5	95.6 (21)	95.6	**96.8**
RF	97.6 (30)	97.9	98.1 (24)	**98.1**	97.3

Despite the case of COOLL dataset, where all the classifiers achieved similar performances, the SVM and RF classifiers turned out to be the best ones for the LIT dataset.

Both SVM and RF exploit an heuristic approach which tries, through the optimization of some parameters, to produce the best possible solution for the data separation [26]. SVM intrinsically tries to maximize the margin between hyperplanes defined by the so-called support vectors. Therefore, if normalization is used, SVM performs well because pattern recognition is performed over data with the same scale. On the other hand, RF works properly even when data are on various scales, not requiring previous normalization process. Moreover, RF is a committee method, meaning that potentially better accuracy can be achieved by increasing the committee's complexity. Also, RF works very well even when the feature space is non-linear, which is our case. This is why, in the end, RF was chosen as the best model for classification. In Fig. 4 the learning and ROC curves of the Random Forest classifier are shown. The learning curve shows that, with more than 2000 training samples, there is no substantial increase on the accuracy value of a validation dataset. Thus, a higher number of training samples is not necessary. Also we can notice that the variance and the error in the case of the validation curve are very low, suggesting that we do not observe overfitting or underfitting conditions in our classifier.

The ROC curves show, instead, that the results achieved for the RF classifier are really close to the ones of an ideal classifier, meaning that the classifier is capable of discriminating among different classes and identify them correctly. Since we are in a multi-class problem and ROC curve is an instrument for binary classification, for every class, a one-versus-rest approach was used every time a ROC curve was compiled.

C. Embedded System

The whole algorithms were embedded on the NVIDIA Jetson TX1 board, as shown in Fig. 2, and directly ran from the terminal.

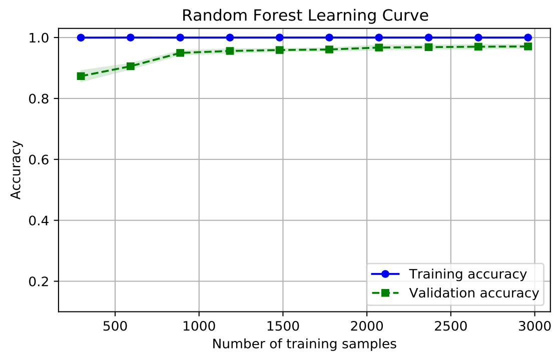
Once a classifier has been trained, there is the possibility of saving the respective model for further classification tasks. Therefore, after the model evaluation process, the classifier was included in the whole algorithm that, in sequence, applies detection, disaggregation, features' calculation, and classification processes. This algorithm was used in simulations over the test dataset to emulate a device capable of collecting external data and recognize the respective loads. In the simulation, the execution time of all the classifier models, when analyzing one single appliance, were recorded:

- LR 1.47550 ± 0.0054 μs;
- SVM 2.71 ± 0.6 μs;
- KNN 4.140 ± 0.09 μs;
- RF 117.74 ± 1.4 μs.

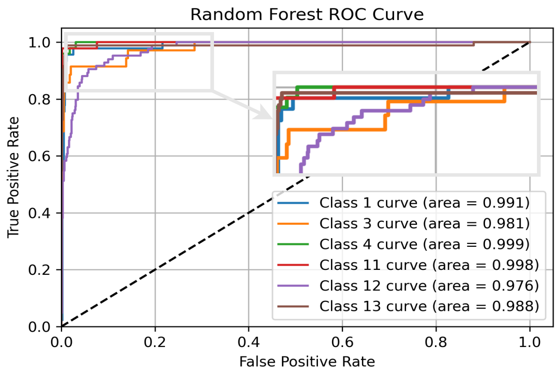
It is interesting to see how the complexity of a classifier influences the execution time. The tests evidence that, in general, the RF classifier takes almost 100 times more than other classifiers to be executed. That is because multiple decision trees (18 in our case) are evaluated for the final classification. Moreover, to see how much the whole process takes, also the execution times related to aggregated waveforms were analyzed, i.e.:

- 1APP 0.13094 ± 0.0020 ms;
- 2APP 0.426 ± 0.07 ms;
- 3APP 0.688 ± 0.08 ms;
- 8APP 2.987 ± 0.08 ms.

Note that, in this case, the RF classifier was implemented. In particular, for 1APP waveforms, the execution time takes a little bit longer than the RF result previously registered due to the detection and disaggregation processes. As expected, we can see that higher the number of appliances, the longer it takes to analyze them. This problem may be solved if all the signals are processed in parallel.



(a) RF Learning curve



(b) RF ROC curve

Fig. 4: Random Forest classifier results for the LIT dataset. The former figure shows that for a training dataset dimension higher than 2000, the validation set's performance is almost constant, with very low variance and error. Instead, the latter figure shows the mean ROC curve of 6 different classes in LIT Dataset (please see Table II for the class ID-device correspondence)

VI. Conclusions and Future Work

In this work, a possible real implementation of a NILM system has been presented. By exploiting two publicly available datasets, we showed that the proposed approach allows to build a device that recognizes the appliances connected to the system that we want to monitor. In particular, from the tests and results, the random forest classifier appeared to be the most robust classifier for our purposes, whose model misclassified only 36 appliances over more than 4000 simulations.

The problem that limited the project was the lack of a robust and reliable detector. Although the core part of the project is the classifier, as long as a proper algorithm for detection is not found, some important error components will prevent features from being extracted correctly, therefore yielding errors in classifiers' predictions. A possible solution is provided in [23], where the selectivity of multiple detectors is combined in an ensemble detector, able to choose the best interval that has been targeted from its sub-units. Its implementation might be one of the future improvements in this project.

We also demonstrate that the algorithm fits the purposes of building a device that collects current and voltage waveforms and displays loads information. In this project, the NVIDIA TX1 board turned out to be reliable and fast, requiring no such high implementation and computational effort as expected. In this sense, another further improvement is making the board able to analyzed multiples signals in parallel. This could be only achieved, for example, if the whole algorithm is mapped to the GPU to exploit its power capability and parallel computation, thus increasing code optimization and reducing power consumption.

Acknowledgments

The authors would like to acknowledge the scholarship of the Electronics Engineering course of the University of Modena and Reggio Emilia.

References

[1] G. W. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, pp. 1870 – 1891, 04 1992.

[2] T. Picon, M. Nait Meziane, P. Ravier, G. Lamarque, C. Novello, J.-C. Le Bunetel, and Y. Raingeaud, "COOLL: Controlled on/off loads library, a public dataset of high-sampled electrical signals for appliance identification," *arXiv preprint arXiv:1611.05803 [cs.OH]*, 2016.

[3] M. Figueiredo, A. Almeida, and B. Ribeiro, "Home electrical signal disaggregation for non-intrusive load monitoring systems," *Neurocomputing*, vol. 96, pp. 66–73, 2012.

[4] A. Zoha, A. Gluhak, M. A. Imran, and S. Rajasegarar, "Non-intrusive Load Monitoring approaches for disaggregated energy sensing: A survey," *Sensors (Switzerland)*, vol. 12, no. 12, pp. 16 838–16 866, 2012.

[5] B. Machado Mulinari, R. Linhares, D. Campos, C. Costa, H. Ancelmo, A. Lazzaretti, E. Oroski, C. Erig Lima, D. Renaux, and F. Pottker, "A new set of steady-state and transient features for power signature analysis based on v-i trajectory," in *IEEE PES Innovative Smart Grid Technologies Conference - Latin America*, 09 2019, pp. 1–6.

[6] A. L. Wang, B. X. Chen, C. G. Wang, and D. D. Hua, "Non-intrusive load monitoring algorithm based on features of V–I trajectory," *Electric Power Systems Research*, vol. 157, pp. 134–144, 2018.

[7] A. Ruano, A. Hernandez, J. Ureña, M. Ruano, and J. García, "Nilm techniques for intelligent home energy management and ambient assisted living: A review," *Energies*, vol. 12, p. 2203, 06 2019.

[8] Y. Zuo, B. Li, Y. Zhao, Y. Jiang, Y.-C. Chen, P. Chen, G.-B. Jo, J. Liu, and S. Du, "All-optical neural network with nonlinear activation functions," *Optica*, vol. 6, no. 9, pp. 1132–1137, Sep 2019. [Online]. Available: http://www.osapublishing.org/optica/abstract.cfm?URI=optica-6-9-1132

[9] M. Nait Meziane, P. Ravier, G. Lamarque, J. Le Bunetel, and Y. Raingeaud, "High accuracy event detection for non-intrusive load monitoring," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 2452–2456.

[10] D. Renaux, R. Linhares, F. Pottker, A. Lazzaretti, C. Lima, A. Coelho Neto, and M. Campaner, "Designing a novel dataset for non-intrusive load monitoring," in *2018 VIII Brazilian Symposium on Computing Systems Engineering (SBESC)*, 2018, pp. 243–249.

[11] A. Ukil and R. Živanović, "Adjusted haar wavelet for application in the power systems disturbance analysis," *Digital Signal Processing*, vol. 18, no. 2, pp. 103 – 115, 2008.

[12] M. Bollen and I. Gu, *Signal Processing of Power Quality Disturbances*. John Wiley & Sons, 2006.

[13] P. Dantas, W. Sabino, and M. Batalha, "Energy disaggregation via data mining," in *Proceedings of the 4th Brazilian Technology Symposium*, 05 2019, pp. 541–546.

[14] D. P. B. Renaux, C. R. E. Lima, F. Pottker, E. Oroski, A. E. Lazzaretti, R. R. Linhares, A. R. Almeida, A. O. Coelho, and M. C. Hercules, "Non-Intrusive Load Monitoring: an Architecture and its evaluation for Power Electronics loads," in *IEEE International Power Electronics and Application Conference and Exposition (PEAC)*, nov 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8590472/

[15] H. Ancelmo, B. Machado Mulinari, F. Pottker, A. Lazzaretti, T. de Paula Machado Bazzo, E. Oroski, D. Renaux, C. Erig Lima, and R. Linhares, "A new simulated database for classification comparison in power signature analysis," in *20th International Conference on Intelligent Systems Applications to Power Systems*, 12 2019, pp. 1 – 7.

[16] Y. Du, L. Du, B. Lu, R. Harley, and T. Habetler, "A review of identification and monitoring methods for electric loads in commercial and residential buildings," 10 2010, pp. 4527 – 4533.

[17] J. Liang, S. K. K. Ng, and J. W. M. Kendall, G.and Cheng, "Load signature study –part i: Basic concept, structure and methodology." *IEEE Transactions on Power Delivery*, vol. 25, no. 2, pp. 870–878, 2010.

[18] A. Zoha, A. Gluhak, M. Imran, and S. Rajasegarar, "Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey," *Sensors (Basel, Switzerland)*, vol. 12, pp. 16 838–16 866, 12 2012.

[19] M. Nait-Meziane, P. Ravier, K. Abed-Meraim, G. Lamarque, J.-C. Bunetel, and Y. Raingeaud, "Electrical transient modeling for appliance characterization," *EURASIP Journal on Advances in Signal Processing*, vol. 55, pp. 1–19, 12 2019.

[20] G. Jacobs and P. Henneaux, "Unsupervised learning procedure for nilm applications," in *2020 IEEE 20th Mediterranean Electrotechnical Conference (MELECON)*. IEEE, 2020, pp. 559–564.

[21] M. Nait-Meziane, P. Ravier, G. Lamarque, J.-C. Le Bunetel, and Y. Raingeaud, "High accuracy event detection for non-intrusive load monitoring," in *International Conference on Acoustics, Speech, and Signal Processing*, 03 2017.

[22] J. Kelly and W. Knottenbelt, "Neural nilm: Deep neural networks applied to energy disaggregation," in *Proceedings of the 2nd ACM international conference on embedded systems for energy-efficient built environments*, 2015, pp. 55–64.

[23] D. P. B. Renaux, R. R. Linhares, F. Pottker, A. E. Lazzaretti, C. E. E. de Lima, A. Coelho-Neto, and M. H. Campaner, "Designing a novel dataset for non-intrusive load monitoring," in *VIII Brazilian Symposium on Computing Systems Engineering*, 2018.

[24] ——, "Designing a novel dataset for non-intrusive load monitoring," in *VIII Brazilian Symposium on Computing Systems Engineering*, 2018.

[25] R. R. Linhares, C. R. E. Lima, D. P. B. Renaux, F. Pottker, E. Oroski, A. E. Lazzaretti, B. M. Mulinari, H. C. Ancelmo, A. Gamba, L. A. Bernardi, and L. T. Lima, "One-millisecond low-cost synchronization of wireless sensor network," in *VIII Brazilian Symposium on Computing Systems Engineering*, 2018.

[26] S. Raschka and V. Mirjalili, *Python Machine Learning, 3rd Ed.*, 3rd ed. Birmingham, UK: Packt Publishing, 2019.

[27] A. S. Bouhouras, P. A. Gkaidatzis, K. C. Chatzisavvas, E. Panagiotou, N. Poulakis, and G. C. Christoforidis, "Load Signature Formulation for Non-Intrusive Load Monitoring Based on Current Measurements," *Energies*, vol. 10, no. 4, pp. 1–21, April 2017.

[28] T. Fawcett, "Introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, pp. 861–874, 06 2006.