Recognizing Appliances and their Events with the same NIALM Algorithm

Marcel Mathis, Patrick Huber, Alexander Klapproth

CC-iHomeLab, Lucerne University of Applied Sciences And Arts

Abstract

People are willing to save energy yet most have little to no understanding about the energy consumption of their household devices. Non-intrusive appliance load monitoring (NIALM) is an approach to automatically measure a household's energy consumption and identify switched-on appliances. This paper presents our ongoing research on improving electric load recognition thus helping people to better understand their energy consumption.

A BFTree detects events and classifies devices using features extracted with a Fourier Transformation. Combining detection and classification into one algorithm is achieved by introducing the concept of "noise". Our novel approach detects events with 93% accuracy and correctly classifies 88% of the switched on devices.

Keywords—NIALM; NILM; load disaggregation; event detection; device categorization; device labelling; device recognition; machine learning algorithm; BFTree; intelligent environment; Fourier Transform

Introduction

While low power consumption of electrical devices is of growing importance for users and manufactures resulting in more power efficient devices, the overall consumption is growing. In Switzerland, one reason for the increasing consumption is the growing population. Another reason is the raising number of electrical household appliances. The Swiss government published different strategies describing how the total energy consumption can be decreased in homes [1], e.g.:

a) Substitution of devices with more efficient replacements
b) Automatic switch off of unused devices
c) Visualization of energy consumption to give users a better understanding of their energy consumption

It is getting more and more difficult to identify household devices that significantly contribute to the energy consumption. New devices have many device states (e.g. a blender can have different spin levels influencing its energy consumption) and it is not straight forward to know the associated energy consumption. In addition, most users have little to no understanding about their energy consumption although they are willing to save energy.

Non-intrusive appliance load monitoring (NIALM) is one approach to face these challenges: Based on low cost measurement systems or smart meter values, NIALM automatically detects and classifies switched-on appliances. Thus, NIALM automatically assess how much energy each appliance consumes and opens new possibilities to save energy by better informing the user: Inefficient devices can be easily identified, even automatically switched off, and real time information on consumption can be provided to users to increase their energy awareness at home.

Current NIALM approaches typically use a two-step approach: first, device state changes such as switch-on or switch-off events are detected without considering the individual device characteristics. If an event is detected, a second algorithm recognizes the device class that caused the change based on previously trained characteristics. These two steps are performed with independent algorithms. In some cases, the classification algorithm will classify incorrectly detected events to known classes. A common source of such events is strong noise as typically generated by switched mode power supplies. The second algorithm must classify these incorrect events to known device classes, although no similar reference devices are trained. This NIALM approach therefore suffers from error propagation through the two steps. In this paper we present an approach to combine the two algorithms while still using an event-based approach. Without setting a single threshold, events are detected correctly in 93% of all cases and in 88%, the true class has been chosen by the classification algorithm.

Related work

The first NIALM system was proposed by Hart [2] more than 20 years ago. A good overview on NIALM research was compiled by Liang [3], [4]. Liang explains that the main challenge in NIALM is to reliably detect different working states of electrical devices in order to maximize the device recognition performance. Our approach builds on Mathis et al.'s work [5] on labelling device states to categories to maximize recognition accuracy.

Recently, Parson presented a new NIALM approach [6], which also uses a single algorithm for device recognition. For his algorithm evaluation, he compared different private and public datasets and looked at different disaggregation algorithms to detect individual devices in a household. He disaggregated the devices in 10s intervals with a Hidden Markov Model. One of the main differences to our work is the sampling rate of the measurements. We use a micro level approach while Parson uses a macro level one. Micro level means that the sampling rate is higher than the fundamental waveform of the AC signal, whereas the macro level approach's sampling rate is lower. While smart meters measure internally in micro level, they often only provide macro level data. Another difference is that Parson does not implement an event based approach.

An often used approach for system validation is using dedicated sub-meters at device level to provide the true switching event of each device. The NIALM algorithm can then be validated on this data. Kolter published such data [7]. As some of these sub-meters still contain more than one device, an evaluation tool has been developed for our application, on which we validate our system. In further work our system will get validated on the public data such as from Kolter.

Methods

The developed NIALM framework is described in the following paragraphs (see Figure 1). Our method works with a single algorithm for event detection and classification and is integrated into the NIALM framework of [5]. This section describes data acquisition, data preparation and subsequent event detection and classification of devices.

Data acquisition

Voltage and current measurements are used as input for NIALM. Current sensors are most suitable to recognize plugged in devices as they register all the electrical device's state changes and therefore are able to disaggregate the individual devices. Adding voltage information enables to trigger the current sensor to the zero-crossing of the voltage. These further increases the accuracy of the total power consumption measurement and phase shifting can get observed.

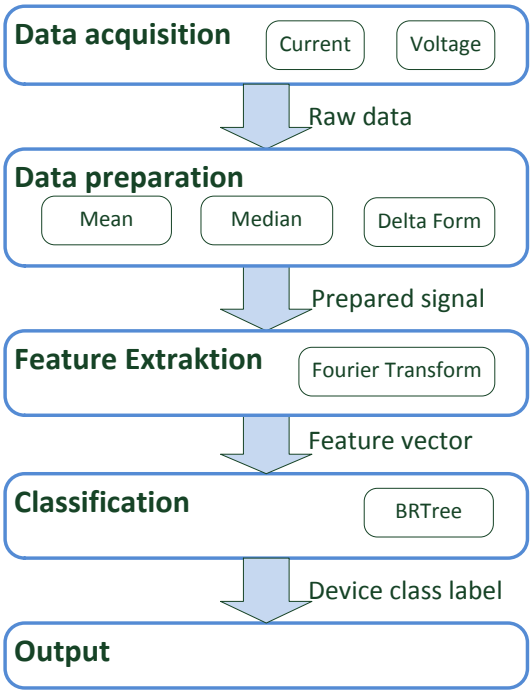

Figure 1 Processes to recognize the devices implemented in the NIALM Framework

According to [8] and [9] the sampling rate of the sensors is a very critical part of NIALM: the higher the sampling rate, the better appliances are recognized. In our application we measured voltage and current at a sampling rate of 50 kHz. For comparison, this high sampling rate was subsampled to different lower rates. Although it is possible to increase signal quantization with down sampling methods, no quantization correction was performed in our case to stay as close as possible to an industrial application. The evaluation of the sampling rate is further detailed in [5].

We use a sampling rate of 5 kHz in our application. On one hand it is a value that standard low power microcontrollers are still able to capture and process; and on the other hand it achieves high recognition accuracy. According to Armel [8] 16 to 32 devices can be accurately recognized with a sampling rate of 5 kHz. She did not provide recognition accuracy information achieved with this amount of devices. Instead of recognizing up to 32 devices with 5 kHz, [5] separated all possible devices to 9 classes to maximize the recognition performance.

Furthermore using 5 kHz allows the calculation of different features using algorithms such as Fourier Transform (FT), Wavelet Transform (WT), Root Mean Square (RMS) or offset (OF) values. Such detailed features cannot be calculated with macro level frequencies.

Data preparation

The aim of data preparation is to get stable signals. This chapter presents different methods of noise reduction to decrease the recognition complexity.

A NIALM system measures the sum of the power consumption of all simultaneously running devices (i.e. the sum curve). Assuming all N devices have only one single device state and a maximum number of k devices running simultaneously, the algorithm has to be trained with C classes. Thus it trains on all device state combinations. The combinatory equation (1) shows the dependencies.

$$C = \binom{N+k}{k} \tag{1}$$

With k = 5 and N = 11 already 4368 classes have to be trained. Since this is not computationally feasible, most NIALM researchers use delta curves to reduce the exponential complexity. A delta curve is defined as the difference between two measurements and assumes that at any point in time during two measurements only one device changes its state between these measurements. If a device changes its state, the delta curve only shows the state change and not taking the running

devices into account. Thus, the NIALM algorithm only knows individual device states instead of all possible device combinations. The graph in Figure 4 shows our proposal to calculate the delta curve from the sum curve. The algorithm is detailed in the following:

1. Sampling of current and voltage over two fundamental periods of the 50 Hz AC signal (40ms). See typical result for a current waveform in Figure 2. This 50 kHz signal is down sampled in further processes to 5 kHz.
2. Compute average over both periods resulting in a single period. The curve resulting from averaging the current is shown in Figure 3.
3. Repeat step 1 and 2 every second.
4. Take 3 consecutive averaged periods and apply a median function at every point of the period. The resulting curve is called window and can be seen in Figure 4.
5. Subtract two windows with a distance of four windows in between. See an example in Figure 4.
6. Run the feature calculation and the classification algorithm on each subtracted window.

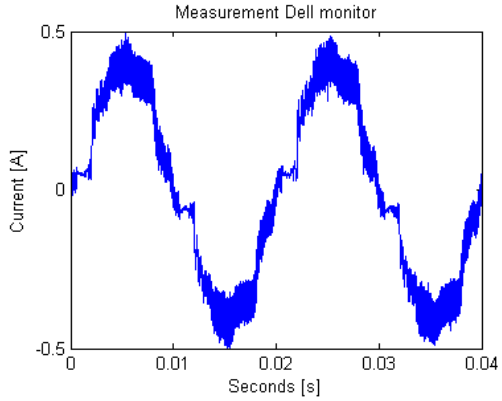

Figure 2 Raw data sample from our current sensor

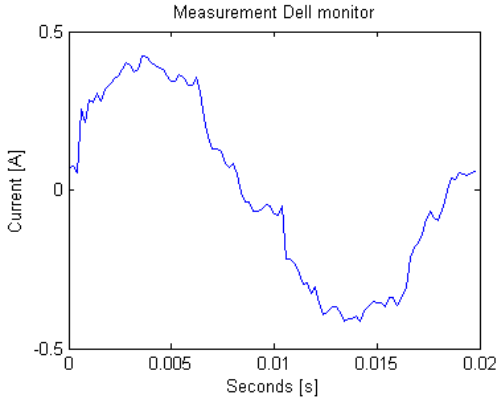

Figure 3 Down sampled signal with mean value calculation from raw data sample

In Figure 4 each small rectangle (M1 to M12) represents one mean curve as it is shown in Figure 3. These curves are sum curves thus representing all simultaneously running devices. Three of these sum curves are calculated in a window with the function $W_i = median(M_i, M_{i-1}, M_{i-2})$. The yellow triangle indicates an event where a device is switched on or off. All green windows (combination of three rectangles representing a single measurement period from step 2) show the sum curves of the running devices before the event whereas the blue windows show the new sum curve after. Red windows are undefined because they capture the intermediate current changes: When a device is switched on/off, it takes time before its power consumption settles. We use these intermediate windows to further increase recognition accuracy.

Using a distance of four windows when subtracting two windows in step 5, allows us to recognize the same device state change several times: the 4th window before the event is subtracted from the first window after the event, the third window before the event from the 2nd after the event, the 2nd before the event from the 3rd after the event, etc. Thus the event shows up on average four times (see Figure 4). Because of that the recognition accuracy is improved. The same event may affect 3 to 5 measurements depending on how long it takes the device state change to settle.

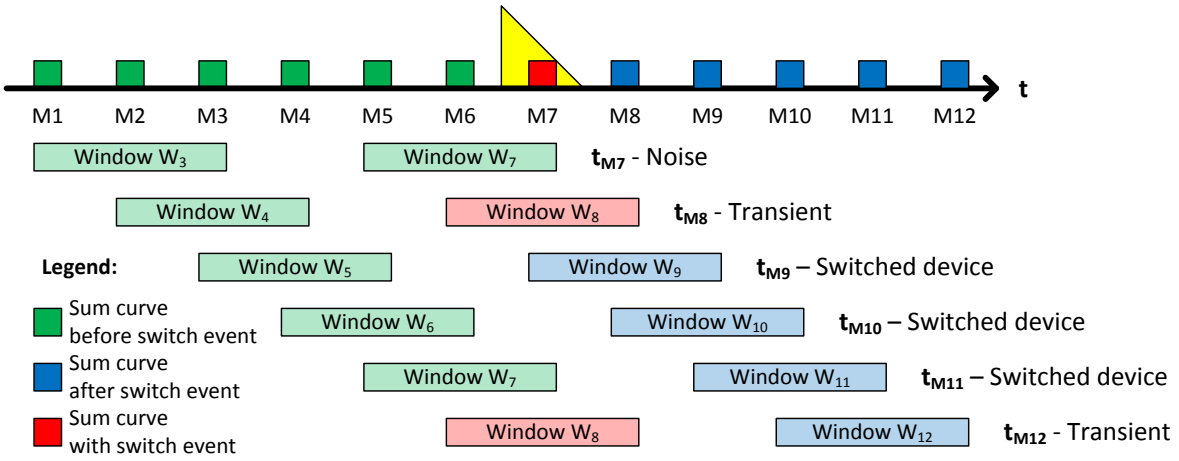

Figure 4 Event detection process to get stable events multiple times

Feature extraction

The measured micro level data makes it possible to calculate a variety of different features from the input vectors of voltage and current. Features are important for machine learning algorithms to compress the data to important property characterizing the individual devices. However, the more features that are used the higher the complexity and computation time. Additionally, with an increasing numbers of features the risk of algorithm over-fitting increases. Over-fitted classification algorithms tend to classify samples wrong if they just differ slightly from the training data. One reason for over-fitted systems is redundant information, which commonly appears in large feature vectors. The feature evaluation in our work focused on small feature vectors to increase robustness.

A feature vector evaluation shows that the parameters of the WT achieve the best recognition accuracy. We looked at the recognition performance of different classification algorithms comparing parameter distributions and using evaluation tools such as WEKA [10]. However, this feature vector is computationally expensive what makes it difficult to calculate in real-time on a low power micro-controller. To guarantee real-time performance a FT is used instead, which performs almost as well as the WT.

Low order odd harmonics up to the 11th order are used from the FT. In addition, the complex power value was added to the input vector. In total, the feature vector contains eight parameters. Theoretically a sampling rate of 1.1 kHz would be sufficient to calculate this feature vector, but comparison in [5] found that a slightly higher sampling rate also increases the quality of low frequency features.

Another reason for using low order harmonics in our feature vector is that many modern devices use DC converters. A common low power DC converter has a characteristic current consumption as given in equation (2).

$$I_{AC} = \frac{4 \cdot I_{DC}}{\pi} \cdot \sum_{k=1}^{\infty} \frac{\sin\big((2k-1)\omega t\big)}{2k-1} \tag{2}$$

Equation (2) states that the amplitude of each harmonic is inversely proportional to its order. That means it is difficult to measure high-order harmonics with a limited quantization of the signal. On the other side it also shows that the FT is a suitable tool to describe electronic devices with its harmonics.

For our purpose, the most important information of the FT signal is located in the absolute part [5]. The phase information of higher harmonics has low relation to the device itself. If the phase information is ignored, the feature calculation must be performed after the delta curve calculation so that the sum curve criterion [5] remains satisfied. This criterion is important to eliminate the influence of the other sum curve devices. Our method calculates the delta curve from the voltage and current values before the feature vector is extracted. In favour of more accurate values the memory efficient variant has been renounced, which would calculate the feature vector before the delta curve.

Device identification

The extracted feature vectors from the previous steps are used to classify the devices using machine learning algorithms. Different supervised machine learning algorithm concepts which adopt their behaviour from experience were studied in [5]. The best performance is reached with the BFTree algorithm [10]. It builds a decision tree that trains with the Best-First method. Basically, a BFTree is a binary tree with logarithmic complexity in memory-use and computation.

Training data

The recognition algorithm was chosen with the constraint of being able to train the recognition algorithm with a single switched on device at a time. This has the advantage that manufacturers of NIALM can train their system in advance so that it works out-of-the-box on the customer's side from a user perspective. This process is possible through a labelling method with a limited device topology as explained in [5]. Thus, the device topology must not contain the actual user devices for the algorithm training; it easily can be done with the manufactures device topology.

Therefore each device was measured isolated from other devices first. 32 devices and 64 different device states were measured. Only device states with a minimum stable duration of about 10 seconds were captured. This includes low energy consumption states (e.g. standby), highly variable devices (e.g. running computers) and variable loads (e.g. dimmable lamps or charging computers). In total, 3177 samples of isolated devices were measured. These samples were then used to train the recognition algorithm. In this dataset, the amount of measurements of devices with large device state variation is much higher. This results in a highly imbalanced data set. An equal weighted cost matrix is used to balance the data to ensure that the algorithm does not optimize its behaviour according to the occurrence of different device measurements. So each class is equally likely to get recognized.

In our application, the training data was captured over a short time period. Considering each device state separately, the individual measurement are just a few seconds apart from each other. Therefore, the noise in these measurements differ just little compared to measurements from other days with different environment variables. Algorithm training with small variance on the data, as used in our application, often suffers from over-fitted algorithm systems.

Test data and validation method

Test data was measured as close as possible to real applications. Compared to the training set, the sum of different devices using electrical power simultaneously was captured. As a result, the test set is independent from the training data and can be used to detect over-fitting of the algorithm. A challenge in NIALM is to obtain corresponding data for the evaluation of the algorithm, as long time intervals have to be labelled with true switching events. Without labelled data a system cannot be validated, because it is unknown whether an event was correctly detected by the NIALM system from the sum curve. Figure 5 explains our validation tool in more details. In future work our system will get validated on public data such as from [7].

A selection of devices of the training set was connected to our NIALM data acquisition system. A KNX actor switched the devices randomly on and off and logged the corresponding time of these switch events. The NIALM system measures just the sum curve and tries to detect the individual device states. The validation was done by comparing the log file from the KNX actor, which shows the true events, with the power log file recorded by the NIALM system.

The test measurements were spread over 6 days, thus incorporating noise conditions from different days and daytimes. Compared to the training set, no cost matrix has been used, because real environment data would be imbalanced as well.

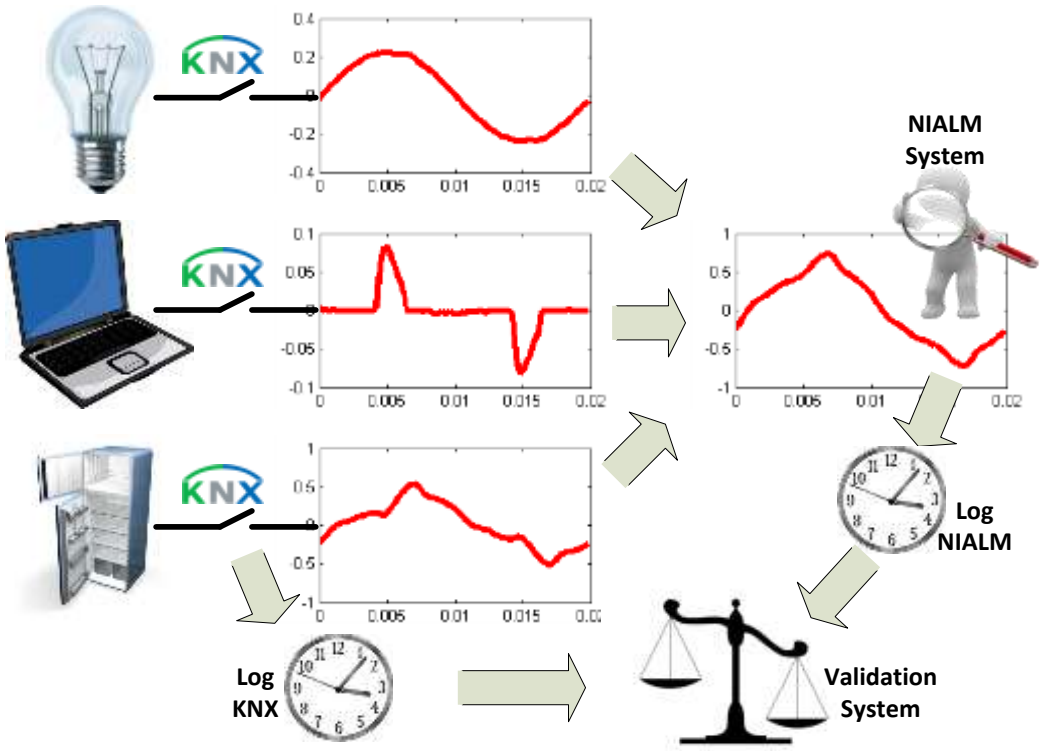

Figure 5 Validation of the NIALM System with KNX to generate a predefined behaviour of the devices

Labeling

To get accepted by users, NIALM has to achieve a minimum overall recognition accuracy of 80 to 90% [11]. This accuracy includes the event detection and the classification step. The challenge is that most households exhibit a rising number of electronic devices. One way is the user has to update the algorithm with every new device at home. Such systems suffer from decreasing performance rate over time as the total number of trained devices increases. Another way is the algorithm has to be trained to all possible devices in advance from the manufacture. Both approaches are unrealistic for the mass market. The user at home has rarely over all device states (e.g. fridge) to train new electronic devices. For the manufacturer, it would be very cost intensive initial and maintains work as there are already too many different devices on the market.

In our approach, we categorize the data for training the algorithm according to the normed current form. The norming condition was defined such that the mean current over the measurement time interval equals one. Equation (3) shows how the normed current form is calculated: $i(n)$ is the measured current vector, $i_{Normed}(n)$ is the normed current form and N is the amount of samples in a measurement.

$$i_{Normed}(n) = \frac{N}{\sum_{k=1}^{N} i(k)} \cdot i(n) \tag{3}$$

At the moment, we categorize devices into 9 classes based on a similar wave form of the normed current signal. The classes are described in more detail in [5]. Based on these classes, the NIALM algorithm is able to inform users on whether they spend most of the energy bill on lighting, electronic devices or on other device categories. This method has the advantage of a constant number of classes such that the recognition performance doesn't depend on the number of devices in a household. Thus adding a new device neither requires retraining nor affects recognition performance. Another advantage is that statistically, a lower amount of classes to recognize usually increases the recognition performance. However there is a trade-off between the amount of information available for the user and the recognition performance.

Working with delta curves allows the algorithm to identify one switching action at a time. If more than one device changes its state simultaneously, a NIALM algorithm based on delta curves can ideally only identify one of them. This typically happens with e.g. switching on power strips. With normed current curve classification the algorithm is able to classify all simultaneously switched devices as long as they belong to the same class.

Noise

The exact noise amount and form is always an unknown value. Each component of the household's power grid can be a source of noise for the NIALM measurement system. Long connections have different impedances or reflection of signals. The unforeseeable consumption part of a device is interpreted as noise:[A1] the algorithm interprets the part of a signal that differs from their training curves as noise. The most significant noise source originates from switched on appliances. Devices like a common halogen light bulb are devices with low noise injection, but there are also devices with high consumption variation. That means noise differs from measurement to measurement. Noise also changes due to environment variables.

Combining the algorithms of event detection and classification poses the problem of how to train the machine learning algorithm to noise. In real applications, it is common that environment variables change over time and thus also noise changes over time. Noise samples that represent all these changes are needed to make the algorithm resilient to noise. These samples are added on the recognition algorithm. A comparison of different types of noise signals will be revealing about it. The following types of noise are compared:

The different types of used noise are the following:

Training data: The noise is calculated from the training dataset by subtracting two measurements of a certain device state from each other. This process is repeated up to 20 times with randomly chosen device states. Finally, the differences are summed up to one total noise curve.

Test noise: The sum curve of six switched on devices was measured periodically. The difference between two randomly chosen measurements was then used as noise.

XX mA: Artificial noise is calculated. The form of noise is white Gaussian noise with a variance of 20, 40, or 80 mA.

Results

A comparison of the recognition performance with different noise signals is shown in Table 1 and Table 2. The rows list different noise datasets that include measured and artificial noise. The columns list the different training datasets. For these, the measured isolated devices were overlaid with artificial noise. Typically noise on training data reduces the recognition performance of the algorithm, and also reduces the chance of over-fitted systems. A performance increase with more noise on the training data typically is a sign for an over fitted-system. All comparisons distinguish between event and total recognition. The event detection differentiates between device and no device whereas total recognition also tries to detect the correct label.

In Table 1 each of the 3998 test measurements from the test set was evaluated separately. The order of the data does not matter. Thus each sample was treated as an event. These results are based on highly imbalanced data. For example 2754 measurements show just noise signals. So if an algorithm classifies all measurements as noise it already reaches a recognition accuracy of 69%. The result of this evaluation procedure is shown in Table 1.

Table 1 Comparison of the recognition performance of the NIALM algorithm with different noise signals. The event column shows the percentage of correctly recognized events and the total column the percentage of also correctly recognizes classes

Training / Noise	No noise		20 mA		40 mA		80 mA	
	Event	Total	Event	Total	Event	Total	Event	Total
Training data	92.8%	81.2%	**93.2%**	**85.6%**	**94.1%**	**88.5%**	**96.5%**	**90.1%**
Test noise	90.7%	79.1%	78.5%	72.1%	81.0%	74.9%	84.5%	77.6%
20 mA	88.4%	76.2%	88.6%	80.3%	-	-	-	-
40 mA	95.7%	83.5%	-	-	**96.4%**	**90.4%**	-	-
80 mA	52.7%	39.2%	-	-	-	-	**97.8%**	**92.7%**

Users are generally not interested in how many samples with noise are classified as noise (Formula 4 cell: True negative). They are interested in how many switching events occurred and which device class changed its state. Based on the data in Table 1, the results in Table 2 show an evaluation where TN is factored out. Formula (4) describes the updated calculation of the performance rating.

$\boldsymbol{classified\ as \rightarrow}$	$\boldsymbol{Event}$	$\boldsymbol{NoEvent}$
$\boldsymbol{Event}$	$TP\ (True\ Positive)$	$FN\ (False\ Negative)$
$\boldsymbol{NoEvent}$	$FP\ (False\ Positive)$	~~$TN\ (True\ Negative)$~~

$$Performance = \frac{TP}{TP + FP + FN} \tag{4}$$

When TNs are ignored the result is more likely to reflect the interests of the user. To improve recognition accuracy a basic error correction method was applied to ignore single misclassifications. This correction is possible due to the chosen kind of delta curve calculation. Therefore four measurements are taken on average during an event (see Figure 4). If those four measurements contain the same classification result at least three times it is classified as a new event and otherwise as noise. The algorithm is shown below:

1. Determine the absolute majority of device classes amongst the last 4 windows. (A majority is reached if at least three of the measurements are recognized as the same device class)
    a. If no majority exists, classify it as noise
    b. If a majority exists classify it as that device class, if not noise
2. Look at previous event
    a. Event, if different to previous
    b. No new event

Table 2 Comparison of the recognition performance of the NIALM algorithm with TN ignored and correction algorithms applied.

Training / Noise	No noise		20 mA		40 mA		80 mA	
	Event	Total	Event	Total	Event	Total	Event	Total
Training data	82.3%	59.3%	81.2%	69.3%	**90.8%**	**83.2%**	**90.6%**	**86.4%**
Test noise	83.3%	61.8%	65.7%	60.7%	69.1%	63.5%	73.5%	69.4%
20 mA	78.8%	56.8%	80.9%	67.1%	-	-	-	-
40 mA	87.0%	62.7%	-	-	**91.9%**	**83.4%**	-	-
80 mA	41.2%	26.8%	-	-	-	-	**93.0%**	**87.7%**

The test data contains a total of 288 true events. Recognition accuracies above 90% are coloured green in Table 1 and Table 2. 90% is a value that leads to user acceptance according to Zeifmann [11].

Discussion

Compared to most NIALM approaches we used one algorithm for event detection and classification. The advantage is that all devices are known by the single algorithm at all times. Hence the algorithm can determine if a single sample is a known device or noise.

We have developed a built-in dynamic threshold for noise classification; the majority of research into NIALM systems [12], [13], [3] uses static thresholds to recognize events. Although the sample is over an event threshold and therefore may be classified as a device class, our algorithm compares it with trained device classes. If it is not similar to one of the known classes, it still can be still can be classified as noise.

Further if an event detection algorithm incorrectly detects an event and hands it over to the classification algorithm, the latter has to classify this incorrect event. Furthermore, the classification algorithm always delivers a result even if the to be classified sample has no close match in its database. A comparison with convenient algorithms has not been done because so far no standard approach to compare the result has been established. Traditional algorithms usually use different kinds of algorithms for the event detection and the classification to increase the performance.

Algorithm over-fitting

Table 1 and Table 2 show similar results concerning the over-fitting of the algorithm: as more noise is overlaid on the training data, the results get better. [A2]This is a sign for an over-fitted system. That means even a slight change of a measurement can lead to a misclassification by the algorithm. In other words the measured noise on the training data differs from the noise on the test data.

To reduce the incorrect classifications further, real world data should be captured over several days and at different times. However, even with training data over a long time period we still recommend to overlay artificial noise: although noise reduces the information content of the sample, it makes the algorithm more stable. In other words noise decreases the signal-to-noise ratio which makes it more difficult to get the information from the signal. Caution is required, that the algorithm does not learn specific noise types instead of the content itself.

In our results we compared different kinds of noise overlaid on the training data. The more training data we have with different noise, the better the algorithm learns the device itself. We believe that the recognition accuracy will increase again with combinations of different noise types.

Another question is the amount of noise the algorithms get trained on. A combination of different amplitudes is recommended. The variance of the highest amplitude we compared was 80 mA. That means there is 18.4 W noise power on each signal. The validation also considers an 8 W LED bulb. Therefore higher noise heavily decreases the recognition of such low power devices.

Another important result is that it still is possible to get good performance with data that lead to over-fitted systems. This is an important fact for users trying to train their system in a short time period: by overlaying their data with noise the classification performance is increased. [A3]

Training noise

Our algorithm recognizes noise. While in traditional algorithms everything below a threshold is classified as noise, our approach is more complex as noise has its distinct class just as devices.

In Table 1 and Table 2 we compare the impact of different noise signals ranging from real measured to artificial white Gaussian noise on recognition accuracy. The best result could be reached with strong artificial white Gaussian noise. Training an algorithm with a combination of several noise signals may further improve performance.

Performance evaluation

In Table 1 we classified each measurement independently. Usually, in real domestic environments switching events from devices do not occur very often. Most of the measurement variations are due to noise and not switched on/off devices. Users are mostly interested in these switching events and not in noise. Therefore it is important that correctly classified noise is not considered in the performance

evaluation. Table 1 shows the performance of each sample compared to Table 2 which shows the performance of the events with the same correction algorithms. The algorithms in both evaluations were trained with balanced data although we used highly imbalanced data sets. With this small correction algorithm almost identical results can be achieved based on events, such as with the algorithm involving all samples individually. It is expected that the results based on events can be even higher with heuristic algorithms. Heuristics base on experience. E.g. a switched on application can't be turned on again or typically switch on duration or times could be considered.

Event detection

The confusion matrix from formula (5) shows the best classification results (Table 2). The performance of the event detection is maximized if the training data is overlaid with 80 mA white Gaussian noise. The same has been applied to noise itself. So noise has been trained with 80 mA artificial white Gaussian noise:

$$\begin{array}{lcc} classified\ as \rightarrow & Event & NoEvent \\ Event & 281 & 7 \\ NoEvent & 14 & (2696) \end{array} \quad (5)$$

In (5) all NoEvents that have been classified correctly as NoEvents are ignored as it is not interesting for the user and would distort the result. Despite that still a recognition accuracy of 93% could be reached.

Conclusion

In this paper we looked at ways to increase total recognition performance of NIALM systems. A typical NIALM approach uses a two-step algorithm for event detection and device classification. Since the first step's output is the second step's input, errors can be propagated. In our approach, a single step algorithm is used. To combine both steps a noise class is added to the existing (device) classes. Rather than first filtering noise before recognizing the actual device class as done by most NIALM approaches, our algorithm classifies each input as one of the defined classes which also includes a distinct noise class. To increase algorithm stability it was trained on data overlaid with noise. The results show that our algorithm classified devices with 87.7% accuracy and also correctly detected 93% of the events. We used a BFTree [10] and added artificial white Gaussian noise with a variance of 80 mA to all device classes when training the BFTree. We plan to explore using heuristics to potentially further increase recognition accuracy. In sum, our work considerably improves the recognition of household devices' and therefore better allows users to understand their energy consumption.

References

[1] VSE, "Wege in die neue Stromzukunft," Verband Schweizerischer Elektrizitätsunternehmen, 2012. [Online]. Available: http://www.strom.ch/uploads/media/VSE_Wege-Stromzukunft_Gesamtbericht_2012.pdf. [Accessed 02 02 2015].

[2] G. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE,* vol. 80, no. 12, pp. 1870-1891, 1992.

[3] J. Liang, S. Ng, G. Kendall and J. Cheng, "Load signature study x00A1;V part I: Basic concept, structure and methodology," in *Power and Energy Society General Meeting, 2010 IEEE*, 2010.

[4] J. Liang, S. Ng, G. Kendall and J. Cheng, "Load signature study x00A1;V part II: Disaggregation framework, simulation and applications," in *Power and Energy Society General Meeting, 2010 IEEE*, 2010.

[5] M. Mathis, A. Andrushevich, A. Rumsch, R. Kistler und A. Klapproth, «Improving the Recognition Performance of NIALM Algorithms through Technical Labeling,» in *Embedded and Ubiquitous Computing (EUC), 2014 12th IEEE International Conference on*, 2014.

[6] O. Parson, "Unsupervised Training Methods for Non-intrusive Appliance Load Monitoring from Smart Meter Data," 2014.

[7] J. Z. Kolter and M. J. Johnson, "Redd: A public data set for energy disaggregation research," in *proceedings of the SustKDD workshop on Data Mining Applications in Sustainability*, 2011.

[8] K. C. Armel, A. Gupta, G. Shrimali and A. Albert, "Is disaggregation the holy grail of energy efficiency? The case of electricity," *Energy Policy,* vol. 52, no. 0, pp. 213-234, 2013.

[9] A. Zoha, A. Gluhak, M. A. Imran and S. Rajasegarar, "Non-Intrusive Load Monitoring Approaches for Disaggregated Energy Sensing: A Survey," *Sensors,* vol. 12, no. 12, pp. 16838-16866, 2012.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD explorations newsletter,* vol. 11, no. 1, pp. 10-18, 2009.

[11] M. Zeifman, "Disaggregation of home energy display data using probabilistic approach," *Consumer Electronics, IEEE Transactions on,* vol. 58, no. 1, pp. 23-31, 2012.

[12] K. Anderson, M. Berges, A. Ocneanu, D. Benitez and J. Moura, "Event detection for Non Intrusive load monitoring," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, 2012.

[13] M. E. Berges, E. Goldman, H. S. Matthews and L. Soibelman, "Enhancing electricity audits in residential buildings with nonintrusive load monitoring," *Journal of industrial ecology,* vol. 14, no. 5, pp. 844-858, 2010.

[14] M. Mathis, A. Andrushevich, A. Rumsch, R. Kistler und A. Klapproth, «Improving the recognition performance of NIALM algorithms through technical labeling,» 2014.