Machinaal leren voor het niet-invasief opmeten van stroomverbruik in een woning

Machine Learning for Non-Intrusive Load Monitoring

Leen De Baets

Promotoren: prof. dr. ir. T. Dhaene, prof. dr. ir. C. Develder
Proefschrift ingediend tot het behalen van de graad van
Doctor in de ingenieurswetenschappen: computerwetenschappen

UNIVERSITEIT
GENT

Vakgroep Informatietechnologie
Voorzitter: prof. dr. ir. B. Dhoedt
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2017 - 2018

ISBN 978-94-6355-109-0
NUR 984
Wettelijk depot: D/2018/10.500/27

Ghent University
Faculty of Engineering and Architecture
Department of Information Technology

imec
Internet Technology and Data Science Lab

Dissertation for acquiring the grade of
Doctor of Computer Science Engineering

Dankwoord

"You don't obtain a PhD on your own. "
– My IDLab colleagues

Vier jaar onderzoek kunnen afsluiten met een PhD, is de spreekwoordelijke kers op de taart. Dat dit mogelijk was door de hulp en ondersteuning van anderen, mag zeker gezegd worden. Vandaar, met plezier, dit dankwoord.

"Nothing clears up a case so much as stating it to another person."
– Sherlock Holmes, *Silver Blaze*

Tom, als promotor was het gegeven vertrouwen in mij en jouw enthousiasme voor mijn onderzoek een drijfveer om door te blijven gaan. *Chris*, als promotor zorgde jouw oog voor detail er steeds voor dat mijn analyses en resultaten evolueerden in de juiste richting. *Dirk*, als officieuze promotor, waren jouw kalmte en ideeën de redding in paniekmomenten.

Many other researchers crossed my path during my PhD. *My examination board*, the given feedback, the interesting discussion, and the enthusiasm were an important scientific confirmation. *The NILM community*, the organized events were the opportunity to get feedback on proposed ideas and to find new ones. *Smappee*, the collaboration with Ghent University gave my PhD a practical meaning making it easier to push through. *Mario Berges*, your hospitality in the summer of 2017 made it possible to continue my research abroad in a most cordial and instructive way. It was a pleasure.

"There are always some lunatics about. It would be a dull world without them."
– Sherlock Holmes, *The Red Headed League*

SUMO Lab, the lunches, the occasional dinners, the small talk, the learning about different cultures,. . . I enjoyed it all and I will think back to the time we spent together with a smile. *All the colleagues* with whom I share memories (think of the kubbing at the IDLab-day, organizing the IDLab-quiz, the mini-golf at the sports day, the cleaning after the international food event, the reception on TechBoost, experiencing sunrises in iGent, the jogging at noon, NIPS at 2016, organizing 'de schattenjacht' and the tedX events, . . .), the experience with you was unforgettable.

> ***"If I didn't understand I was being asked to be best man, it is because I never expected to be anybody's best friend."***
> – Sherlock Holmes, *The sign of three*

Els en Lloyd, jullie eeuwige interesse, de leuke spelavonden en de talrijke mascarpone desserten kunnen me alleen doen hopen dat het verleden een teken van meer is in de toekomst.

Cedric en Martijn, al sinds het begin van onze studies stonden jullie klaar om verhalen te vertellen, te klagen, problemen op te lossen en zoveel meer. Het is mij een eer jullie te kennen.

> ***"It has long been an axiom of mine that the little things are infinitely the most important."***
> – Sherlock Holmes, *A Case of Identity*

Moeder en vader, jullie realistische kijk op de wereld hebben me geleerd om 'mislukkingen' te minimaliseren en om steeds verder te blijven gaan. De dagelijkse ondersteuning, al 27 jaar lang, wordt uitermate geapprecieerd. *Broers en zussen*, jullie jongste zus zijn, is een leerrijk avontuur. Samen uitkijken naar dit hoogtepunt was een evidentie voor jullie, waarvoor dank. *Benjamin*, jouw gretigheid om voortdurend nieuwe dingen te ontdekken, werkt nog steeds aanstekelijk. *Liesbeth*, jouw passie voor wiskunde en lesgeven, enthousiasmeerden mij als kind al om vele problemen aan te pakken en op te lossen. *Greet en Jordi*, als voormalig onderzoekskoppel waren/zijn jullie het meest begrijpende luisterend oor voor Jeroen en mij, ook vanuit Cambridge is jullie raad goud waard. De *schoonfamilie*, jullie hebben me nieuwe werelden leren kennen, onder andere deze van het betere bier en het skiën, een aangename afwisseling op het gekende.

> ***"You have a grand gift for silence, Watson. It makes you quite invaluable as a companion."***
> – Sherlock Holmes, *The Complete Sherlock Holmes*

Jeroen, altijd ben je daar en sta je voor me klaar. Een luxe die me duidelijk werd toen ik drie maanden alleen in Pittsburgh was en jij me nog steeds ondersteunde, vertrouwde, en lief had. Ik ben benieuwd wat onze toekomst zal brengen. Zoals Captain Picard zou zeggen, 'Engage'.

Gent, 27 april 2018
Leen De Baets

Table of Contents

List of Figures

List of Tables

List of Acronyms

AC	Air Conditioner
BB	Balanced Boosting
CNN	Convolutional Neural Network
CFL	Compact Fluorescent Lamp
CSV	Comma-Separated Values
dB	decibel
DL	Deep Learning
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
EI	Expected Improvement
EFD	Elliptical Fourier Descriptors
FP	False Positives
FN	False Negatives
GOF	Goodness-Of-Fit
GLRT	Generalized Likelihood Ratio Test
GNB	Gaussian Naive Bayes
I	Inductive
ILB	Incandescent Light Bulb
JSON	JavaScript Object Notation
kNN	k Nearest Neighbors

LR	Logistic Regression
LDA	Linear Discriminant Analysis
LHD	Latin Hypercube Design
NILM	Non-Intrusive Load Monitoring
NN	Neural Network
NL	Non-Linear
PoI	Probability of Improvement
PLAID	Plug-Level Appliance Identification Dataset
R	Resistive
RF	Random Forest
RI	Rand Index
SBO	Surrogate-Based Optimization
SUMO	Surrogate Modelling
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
TP	True Positives
TN	True Negatives
VI	Voltage-Current
VPA	Variable Power Appliances

Samenvatting

Stel jezelf een wereld voor waarin je nooit de lampen vergeet uit te doen, waarin je wordt gewaarschuwd als huishoudelijke toestellen het einde van hun leven hebben bereikt en dringend vervangen moeten worden. Stel jezelf een wereld voor waarin de door jou geproduceerde groene energie efficiënt gebruikt wordt in een slim elektriciteitsnetwerk in plaats van gewoon gedumpt te worden. Deze schijnbaar kleine winsten zullen ons niet alleen geld besparen, maar ze zullen ons ook in staat stellen om de opwarming van de Aarde tegen te gaan, de grootste uitdaging van onze planeet en haar inwoners. Een hulpmiddel om deze wereld te bereiken is **NILM**, wat de Engelse afkorting is voor het niet-invasief opmeten van het stroomverbruik in een woning, het onderwerp van mijn onderzoek. NILM meet het totale energieverbruik van een huishouden zonder invasief te zijn: het gebruikt slechts één sensor. De verzamelde data wordt vervolgens gebruikt om te bepalen welke huishoudelijke toestellen aanwezig zijn, welke al dan niet actief zijn en voor hoe lang. Het omzetten van deze data naar kennis, resulteert in bruikbare inzichten in uw elektriciteitsrekening. Zo kunnen bijvoorbeeld flexibele toestellen gedetecteerd worden. Dit zijn toestellen waarbij alleen de eindtijd van belang is en die dus kan starten wanneer groene energie beschikbaar is. Een voorbeeld van zo een toestel is de vaatwasser. Als deze apparaten gedetecteerd worden in uw huishouden, dan kan dit leiden tot een goedkopere en schonere energierekening.

Er bestaan verschillende soorten NILM. Mijn proefschrift concentreert zich op de **op gebeurtenis gebaseerde methoden**, die de volgende stappen omvatten: (1) data verzamelen, (2) activering of deactivering van toestellen (gebeurtenissen) detecteren, (3) kenmerken extraheren, en (4) classificeren van huishoudelijke toestellen.

Om gegevens te verzamelen, worden twee manieren toegepast: het gebruik van publieke datasets en het opmeten van onze eigen data. Publieke datasets kunnen in twee groepen onderverdeeld worden: degene met lage en degene met hoge frequentie, waarbij de eerste minder informatie bevat dan de laatste. Voorzover ons bekend is, heeft geen enkele huidige dataset zowel toestel afzonderlijk als geaggregeerde data opgemeten met een hoge frequentie. Om een waardevolle bijdrage te leveren aan de NILM-gemeenschap, **hebben we zowel toestel afzonderlijke als**

geaggregeerde data opgemeten met een hoge frequentie en publiek beschikbaar gemaakt. Daarnaast geven we ook een systematische beschrijving van de meetopstelling, zodat deze dataset eenvoudig en consistent uitgebreid kan worden.

In deze datasets worden houshoudelijke toestellen geactiveerd en gedeactiveerd. Telkens wanneer dit gebeurt, noemen we dit een gebeurtenis in de data. Men zou verwachten dat de detectie van deze gebeurtenissen onafhankelijk zou zijn van de andere actieve toestellen. Eerder ontwikkelde methoden lijken echter gevoelig te zijn voor de verandering in het basisverbuik veroorzaakt door het actief zijn van geen of meerdere toestellen en falen om sommige gebeurtenissen te detecteren wanneer andere toestellen actief zijn. **We stellen twee methoden voor die beter in staat zijn om gebeurtenissen te detecteren, ongeacht het feit of andere apparaten actief zijn**.

In op gebeurtenis gebaseerde NILM-methoden verstaat men onder classificatie het toewijzen van een toestelnaam aan een nieuwe gebeurtenis, op basis van eerder verkregen data bestaande uit gebeurtenissen geassocieerd met toestelnamen. Deze gebeurtenissen hebben **kenmerken**, die op verschillende manieren afgeleid kunnen worden. In dit proefschrift hebben we als kenmerken gekozen voor **VI-afbeeldingen**, die of binaire waarden of waarden tussen 0 en 1 bevatten. Deze VI-afbeelding wordt gemaakt door een tweedimensionale grafiek te maken van de stroom en de spanning, en deze vervolgens te bedekken met een gemaasd netwerk. Verschillende toestellen produceren verschillende VI-afbeeldingen.

Om de toestellen die de afbeeldingen produceren correct te classificeren, gebruiken we twee klassieke beeldverwerkingmethoden. **De eerste methode creëert nieuwe karakteristieken van de VI-afbeelding door contouren in de afbeelding te vinden en de elliptische Fourierdescriptoren (EFD's) daarvan te berekenen.** Deze nieuwe karakteristieken, EFD's, worden vervolgens gebruikt als invoer voor klassieke machinaal leermethoden. Deze methode presteert vergelijkbaar met de methoden uit de literatuur, maar in vergelijking met deze methoden heeft het aanzienlijk minder opslagruimte nodig. **De tweede methode maakt gebruik van de totale VI-afbeelding als invoer voor een convolutioneel neuraal netwerk** en presteert beter dan methoden uit de literatuur, maar in vergelijking met deze methoden vereist het meer opslagruimte. Het toevoegen van het **stroomverbruik als karakteristiek** verbetert de prestatie. De beste prestatie wordt verkregen door het stroomverbruik als enige invoer te nemen.

Het is bekend dat classificatiemethoden gevoelig zijn aan niet-gebalanceerde data. In mijn onderzoek is de data niet gebalanceerd door het feit dat sommige toestellen meer gemeten worden (de meerderheid) dan andere (de minderheid). Als gevolg worden fouten in de meerderheid vaker meegeteld dan fouten in de minderheid, wat hoogst ongewenst is. In dit werk worden verschillende methoden geprobeerd om de data gebalanceerd te maken. **We hebben vastgesteld dat bij gebruik van de juiste classificatiemethode die de VI-afbeelding als invoer**

gebruikt, deze methoden niet leiden tot een aanzienlijke verbetering van de prestaties in vergelijking met wanneer de classificatiemethode wordt toegepast op de originele dataset. Bovendien hebben we geconstateerd dat voor toestelclassificatie die VI-afbeeldingen als invoer nemen meer metingen dan noodzakelijk aanwezig zijn in de publiek beschikbare dataset PLAID.

Nieuwe/niet-geïdentificeerde toestellen hebben de neiging om in huishoudens te verschijnen, maar voor zover wij weten, zijn geen van de huidige NILM-methoden in staat om deze niet-geïdentificeerde toestellen te detecteren. De meeste classificatiemethoden zullen aan de bijbehorende gebeurtenissen een label toewijzen dat overeenkomt met het toestel met de meest vergelijkbare kenmerken. Clustermethoden zoals DBSCAN, die gebruik maken van de dichtheid van de data om clusters te creëeren, kunnnen deze niet-geïdentificeerde toestellen aanwijzen als uitschieters die niet tot een bepaalde cluster behoren. Dit kan enkel gebeuren op voorwaarde dat de toestellen voldoende verschillend zijn in hun karakteristieken. **We stellen voor deze karakteristieken te leren door een siamese neurale netwerk te trainen**. We laten zien dat deze methode wel in staat is om niet-geïdentificeerde toestellen te detecteren. Wanneer echter nieuwe metingen van eerder geïdentificeerde toestellen worden geclassificeerd, wordt een aanzienlijk deel ook als niet-geïdentificeerd gemarkeerd maar ligt het meetpunt wel in de buurt van de juiste cluster.

Als conclusie geldt dat dit werk innovatief onderzoek verricht in elk van de eerder genoemde stappen in op gebeurtenis gebaseerde NILM methoden: we verzamelen data, detecteren gebeurtenissen op een robuuste manier, extraheren karakteristieken uit afbeeldingen en voeren beeldclassificatie uit, met de mogelijkheid om eerder niet-geïdentificeerde toestellen te kunnen detecteren.

Summary

Imagine a world where you never forget to switch off the lights, where you are alerted when appliances within your household have reached the end of their life and need to be replaced. Imagine a world where your renewable energy is used efficiently within a smart grid instead of being dumped. Aggregating these seemingly marginal gains will not only save you money, it will also allow you to actively battle global warming, the major challenge confronting planet earth and its inhabitants. A helpful tool to assist us in reaching this world is called **NILM**, short for **non-intrusive load monitoring**, the topic of my research. NILM measures the total power consumption of a household without being intrusive, using only one sensor. The data captured this way is used to deduce which appliances are present, which are on/off and for how long. Transforming this information into knowledge will allow you to gain actionable insights into your electricity bills. To give a specific example, flexible appliances can be detected. These are appliances where only the finishing time matters and which thus can start whenever green energy becomes available. An example of such an appliance is the dishwasher. Detecting these appliances within your household allows you to change your life nudging you towards a cheaper and cleaner energy bill.

NILM comes in different flavours. My dissertation focuses on the **event-based approach**, which takes the following steps: (1) gathering data, (2) detecting activation or deactivation of appliances (events), (3) extracting features, and (4) classifying appliances.

In these datasets, appliances are activated and deactivated. Every time this happens an event occurs in the data. One would expect this detection to be the same when none or several appliances are active. However, previous methods seem to be sensitive to this change in base load consumption, and miss some events when other appliances are active. **We propose two methods capable of detecting events regardless of the activation of other appliances**, the voting χ^2 and cepstrum method.

Classification within event-based NILM consists of assigning an appliance type to a new event, based on training data consisting of events with known appli-

ance type. These events are characterised using **features**. Choosing these features can be done in multiple ways. In this thesis, we have chosen for **pixelated and weighted pixelated VI images**, containing binary values and values between 0 and 1 respectively. This VI image is created by first building the two dimensional graph of the VI trajectory (plotting the current against the voltage in steady state after the event) and then overlaying it with a pixel mesh. Different appliances will exhibit different VI images.

To correctly classify the observed images into the appliance types producing them, we use two classic methods from the image classification field. **The first method creates new features from the pixelated VI image by finding contours within the image and calculating the elliptical Fourier descriptors (EFDs) from them.** These new features, EFDs, are then used as input for classic machine learning methods, namely logistic regression, random forests, and neural networks. This method obtains performance comparable to state-of-the-art. However, it leads to a significant reduction in needed storage space when compared to the state-of-the-art. **The second method uses the total image as input for a convolutional neural network** and outperforms the state-of-the-art but requires more storage space than the state-of-the-art. Adding **the current consumption as an extra feature**, improves the performance. The best performance is obtained when using the current consumption as only input.

It is well-known that classification methods are sensitive to imbalanced data. In my research, this imbalance is caused by the fact that some appliance types are more measured (the majority), than others (the minority). As a consequence, the cost of misclassification depends on the frequency of appliances in the dataset, which is highly undesirable. Several approaches have been tried to counteract this imbalance: over- and under sampling, synthesising samples, balanced bootstrapping, and adjusting the error function. **We found that when using the correct classifier and the pixelated VI image as input, these methods do not lead to a significant improvement in performance** compared to when the classifier is applied onto the imbalanced dataset. Moreover, we found that for the task of appliance identification and when the VI image is used as input, more measurements than necessary are present in PLAID, a publicly available dataset.

New/unidentified appliances have a tendency to show up in households, but to our knowledge, none of the current NILM methods are capable of detecting these unidentified appliances. Most classification methods will assign the corresponding events a label corresponding to the appliance type with the most similar features. Density-based clustering algorithms like DBSCAN would be able to label the unidentified appliances as outliers provided that the different appliances are clearly separated in the feature space. **We propose to learn this feature space by training a siamese neural network**. We show that this method is capable of detecting unidentified appliances. However, when new samples of previously identified appliances are classified, a significant part is also labeled as unidentified but lies in the proximity of the correct cluster.

To conclude, this research covers innovative research in each of the afore mentioned steps in event-based NILM: we gather data, detect events in a robust manner, extract features from images, and perform image classification, allowing for the detection of previously unidentified appliances.

1 Introduction

"My mind rebels at stagnation, give me problems, give me work!"
– Sherlock Holmes, *The Sign of Four*

My research can be found at the intersection of two fields: non-intrusive load monitoring and machine learning. After a concise introduction to these fields in Sections 1.1 and 1.2 respectively, the research goals and outline of this thesis are discussed in Section 1.3. To conclude, the resulting publications are listed in Section 1.4.

1.1 Non-intrusive load monitoring

Non-Intrusive Load Monitoring (NILM) identifies the per-appliance energy consumption by first measuring the aggregated energy trace at a single, centralized point in a home and then disaggregating this power consumption into individual devices using machine learning techniques. An example is shown in Figure 1.1.

The next subsection motivates the deployment of NILM. Afterwards, different flavors of the input data are listed with their advantages and disadvantages. To conclude, the necessary steps to obtain the final result are discussed.

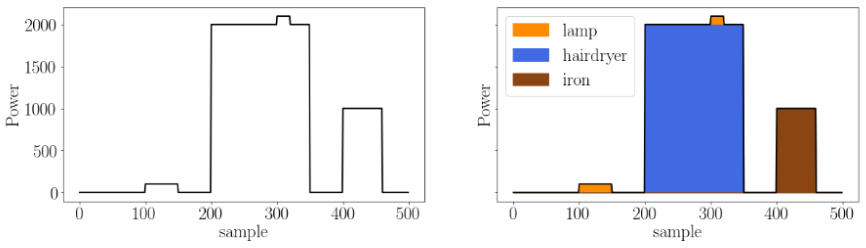


Figure 1.1: An example of an aggregated energy trace disaggregated into the power consumption for individual devices.

1.1.1 Motivation

A major, if not the most important, motivation for NILM is that it facilitates energy monitoring, and thus ultimately contributes to realizing climate targets. As of October 2014, EU leaders agreed upon the following three key targets for 2030 [1]: at least

- 40% cuts in greenhouse gas emissions,
- 27% share for renewable energy, and
- 27% improvement in energy efficiency.

Energy monitoring proves an useful aid for reaching these targets by providing consumers an accurate and detailed view of their energy consumption:

- providing power consumption information to households has been shown to save up to 12% of electrical energy, thereby reducing emissions [2] (note that this also applies to non-residential buildings [3]),
- knowledge concerning present appliances and their usages in the households allow us to assess and exploit flexible power consumption, which is essential for the construction of demand response systems that can increase the penetration of distributed renewable energy sources,
- energy monitoring is a major prerequisite for energy efficiency measures [4].

NILM offers us an elegant way to monitor energy use in a cost-effective way, without the need to rely on expensive per-device monitoring equipment.

Different companies, e.g., Smappee (Belgium) and Verv (UK), have been created around this need for personalized eco-feedback. They have designed a power meter that is easily installable at the main power plug. Statistical insights based on this meter data is presented to the end-user via an application in an online fashion.

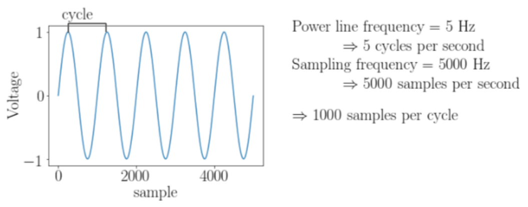


Figure 1.2: An example of a voltage signal captured for one second with a sampling frequency of 5 kHz on a power line with frequency 5 Hz resulting in 1000 samples per cycle.

Online as in if a lamp is turned on, this is immediately detected. Besides ecological motivations, these applications also intend to *make the life of the consumer easier*, e.g., they can notify her when one forgets to switch off the coffeemaker while leaving the house for work, or when a machine is worn out. Though these tools are already on the market, there is ample room for improvement. Some of the challenges to be tackled in order to make NILM more complete are explained in Section 1.3.

Moreover, the information obtained through NILM can also be *valuable for health care applications* [5]. Daily habits of the elderly can be defined using a list of devices active during an ordinary day. This allows for the quick detection of aberrant behaviour. For example, if an elderly person switches on the oven in the middle of the night, this might point to the onset of dementia. Using NILM provides a less intrusive solution than body sensor networks, as only one sensor present in the house is necessary instead of multiple sensors on the body.

1.1.2 Data

NILM relies on the aggregated energy trace measured at a single and centralized point in the home. This aggregated energy trace can be measured at different frequencies and consists of several metrics. *All these metrics can be obtained from two signals, or waveforms: the instantaneous current (i) and voltage (v) signal.* These signals are continuous time series build up by repeating the same form (a cycle) and has a sampling frequency which determines the number of samples per second. The number of cycles per second is determined by the power line frequency. Across the world this is 50 Hz, except for the America and large parts of Asia, where it is 60 Hz. An example of all these variables is given in Figure 1.2.

Different metrics can be calculated from i and v, such as instantaneous, active, and reactive power. In this work, besides i and v, the active power p is mainly used. One sample p_j is calculated as follows:

$$p_j = \frac{1}{n}\sum_{k=1}^{n} i_{j,k} v_{j,k} \tag{1.1}$$

where $i_{j,k}$ and $v_{j,k}$ are respectively the $k$the sample of the jth cycle of current and voltage, and n is the number of samples in a cycle. As the active power is calculated from cycles, the maximal sampling frequency is 50 or 60 Hz, depending on the region of capturing.

Typically, two classes of NILM data are considered:

High-frequency data, data sampled at a high frequency, consist of i and v signals sampled at a frequency of more than 1 kHz. An advantage of this high frequency data is its capture of a lot of information pertaining to active appliances, making it *ideal for appliance detection*. Unfortunately, the use of high-resolution data leads to excessive storage requirements, leading to a significant increase in the processing time of the algorithms.

Low-frequency data, data sampled at a low frequency, can consist of one or more metrics as explained above (e.g., active power p), or of current and voltage signals sampled at a frequency lower than 1 kHz. Advantages of low-frequency data are its limited storage requirements and processing time compared to when using high-frequency data. Furthermore, it still *contains enough information to detect activation and deactivation of appliances*. Considering the exact sampling frequency is important as for example two appliances are activated within 10 seconds from each other, then this will result in two events happening at the same time in a power trace sampled at 0.1 Hz or lower.

Besides the aggregated data, public data sets often offer the ground truth, the so-called *labelled data*. This can consist of power consumption signals or metrics of all active appliances separately (the submetered data) or labels indicating the activation and deactivation of appliances. Often, the data sets contain data pertaining to one or more houses.

1.1.3 Required steps in NILM

NILM has two possible outcomes : (1) a list of appliances and their timestamped activation or deactivation, or (2) a power consumption estimation which can be expressed, a.o., as an individual power consumption profile for each appliance or as a cumulative power use over a certain period of time (e.g., kWh). The first output can be derived from the second, or the other way around, provided that the

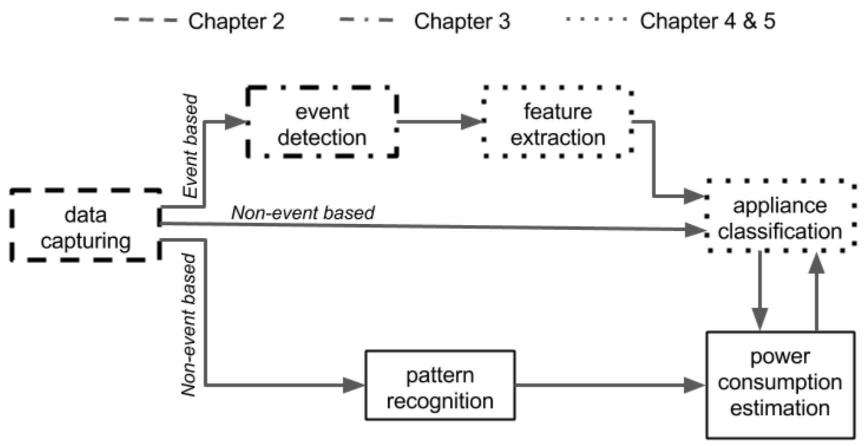


Figure 1.3: A schematic overview of the steps in NILM and the corresponding chapters.

average power drawn from the appliances are known. However, it is not necessary to calculate them both. These outputs allow us to answer meaningful questions, a.o., 'Which appliances are active now?', 'How much power does it take to make this delicious toast?', etc.

The approaches to obtain these outcomes can be divided in two categories: event and non-event based methods. The event based ones are only capable of producing the first type of output, while the non-event based ones can output both types. A schematic overview is given in Figure 1.3. In the following paragraphs, these two approaches are explained in more detail.

Event based methods

In 1992, Hart was the first to describe an event based workflow for NILM [6]. First, it detects state transitions (events) and then it extracts features. Finally it matches these features with unique signatures of appliances using machine learning techniques, i.e., appliance classification.

Event detection Each time a device is activated, power is drawn from the net, while power drawing stops when the device is deactivated. As a result, both activation and deactivation of an appliance are visible within the current and power signal. For some appliances, more specifically those containing a powerful motor (e.g., a vacuum cleaner), activation will also be visible in the voltage signal: its magnitude is smaller for a few cycles. This is illustrated in Figure 1.4. After activation/deactivation of an appliance, current cycles have a higher/lower maximum,

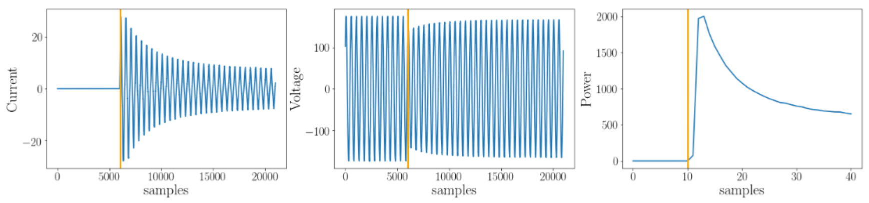


Figure 1.4: An event caused by activating the vacuum cleaner in (a) the high frequent current signal, (b) the high frequent voltage signal and (c) the low frequent power signal.

and a step/event can be detected in the power signal.

Feature extraction Once the event is detected, features can be extracted. These are properties of the appliances causing the events and must be chosen in such a way that different appliances will exhibit different features. Each appliance is made up of different electrical components such as transistors, capacitors, and resistors. Switching on the appliance creates a unique transient signature. As a consequence, features based on this transient signature can be used to separate different appliances. A prerequisite is that high frequency data is needed to capture this transient behavior. These are features that describe the temporal behaviour of the transient (e.g. the transition spike, the length of the transient), or contain frequency-domain characteristics (e.g. harmonics information obtained from a fast Fourier transform, coefficients that can be extracted from the spectral envelope or a wavelet transform on the raw current signal). In addition, features from the steady-state behaviour of the appliance can be extracted. This extraction can be done on high frequency data, but also on low frequency data, e.g., the change in active and reactive power, noise level characteristics, and other statistical features [7].

Appliance classification In order to automatically label the features extracted from an event, we need to construct a classification model. Such models are explained in the next section. A property of these methods is the need to determine unambiguously which appliance has triggered an event, based upon the values of the extracted features. Moreover, they also must be parsimonious and showcase good generalization capabilities (i.e., low misclassification rates) over previously unseen examples not used during model building. In our case, these generalization capabilities take on two forms: unseen samples from the same house containing the same appliances, or unseen samples from another house.

It is reported [8] that each link in the chain of power measurements, event detection and feature extraction can introduce a certain number of error and inaccuracy, so it is important that the classifier showcases robustness against noisy data.

Non-event based methods

If the goal is to output the power consumption estimation for each individual appliance, the non-event based methods disaggregate the power consumption of different appliances using techniques such as latent variable model, time series motif mining, and blind source separation. In essence, they try to match a pattern. A limitation of these methods is their high resource demand, especially for usage on embedded devices.

Non-event based methods are also able to classify active appliances in a given input. Given a window of several minutes or hours, it determines which appliances are active by comparing this current window with known patterns of appliances. The big difference with event-based methods is that now also a negative output is possible in the scenario where nothing is active. These non-event based methods also prove more suitable to detect complex multi-state (mostly wet) appliances, such as washing machines, tumble dryers, dishwashers, etc. These follow certain long running programs whereby a sequence of subsystems with various loads are switched on and off over time during its operation. By focusing on a window of multiple hours instead of a couple of seconds (as is the case in event based methods), it can detect that appliances of a similar nature follow a similar, yet unique pattern and thus construct an alike power consumption.

1.2 Machine learning

Previous section made clear that a lot of data is produced in NILM. To efficiently analyze this data, machine learning can be used. Tom M. Mitchell [9] offers a clear and concise definition of machine learning: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”.

Subsection 1.2.1 presents a machine learning taxonomy containing the different flavours of the task T. Once a task is chosen, the need arises for performance measures P such that future evaluation and comparison is possible, see Subsection 1.2.2. Subsection 1.2.3 explains an important property of machine learning programs, i.e., generalization and Subsection 1.2.4 shows how program parameters can be optimized.

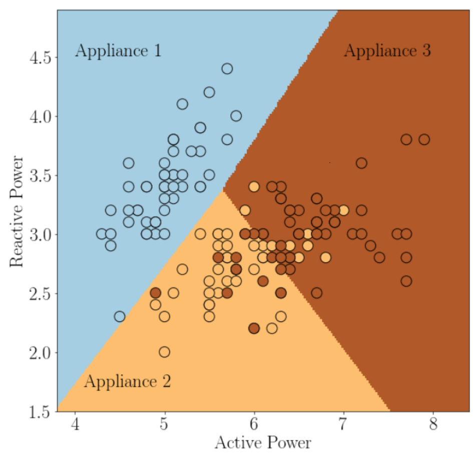
1.2.1 Machine learning taxonomy

Different types of machine learning techniques exist, organizable into a taxonomy based on the type of data available during training. This results in three main types: supervised, unsupervised, and reinforcement learning. Though several other types exist, such as semi-supervised and active learning, we focus on supervised and unsupervised methods in this thesis.

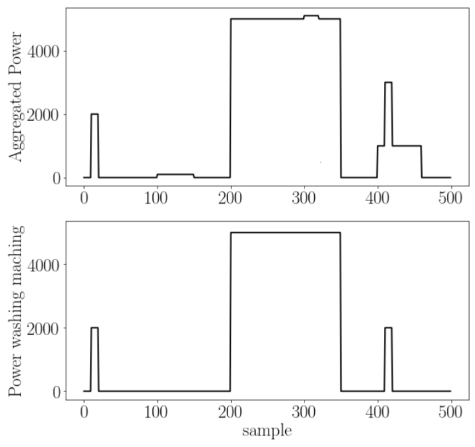
Supervised learning Supervised learning uses labelled training data consisting of input vectors and their corresponding target vectors. After training, the learned model tries to predict the target vectors of new input vectors. Depending on whether the target vectors are classes or floating point numbers, the supervised learning task is a classification or a regression. For example, assume one wants to predict the presence of active appliances by modelling the relationship between active and reactive power, see Figure 1.5a. Based on this artificial data, a classification can be made by determining boundaries between different appliances, making it possible to predict the appliance name of a new sample. An example of regression can be found in Figure 1.5b, where one wants to predict the power consumption of a washing machine by modelling the aggregated power consumption. On this data, regression can be performed allowing for power consumption prediction of the washing machine in a new aggregated signal.

Unsupervised learning Unsupervised learning uses training data consisting only of input vectors. The corresponding target vectors are unknown. From this unlabelled data we automatically extract structure. For example, assume one wants to know whether there are different appliances present given a certain active and reactive power. In the artificial case of data depicted in Figure 1.5c, one would conclude that there are three types of appliances present.

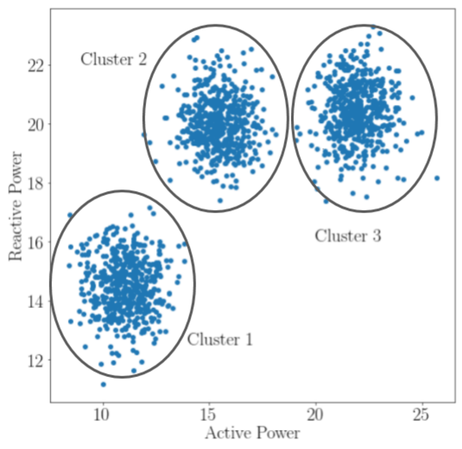
Reinforcement learning Sometimes gathering and labelling training data is practically impossible. Just think of teaching a helicopter how to fly. In the scenario where labelled or unlabelled training data is needed, an almost infinite number of possible actions (lifting off, turning left, etc) can be recorded, which would prove impossible in practice. Moreover, it is hard to decide which data is essential in learning how to fly. Instead of relying on faulty human judgment, a reward can be defined for the helicopter when it performs something good (like flying) and a discount when it does something bad (think crashing). This kind of learning is called reinforcement learning.



(a)



(b)



(c)

Figure 1.5: An example of (a) classification, (b) regression, and (c) clustering.

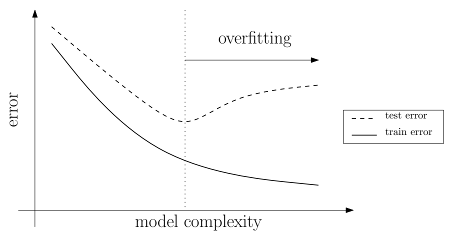


Figure 1.6: An example when generalization decreases with increasing model complexity.

1.2.2 Performance metric

In order to evaluate the developed methods and allow for the comparison of multiple models, a performance metric needs to be defined. The choice of metric(s) depends on the applied machine learning technique and desired outcome. When classes are predicted, the metric will depend on the number of correctly classified samples. For example, the accuracy and the F-measure can be used. When regression is performed and the output is continuous, the metric can depend on the difference between predicted and real value. When no labels are available to calculate this performance metric, the verdict can be based on the shape of the solution, e.g. the closeness of the detected clusters.

If multiple classes are present in the data, the performance metric can be expressed for all classes combined, or for each class separately. It is also possible to give more importance to one of the classes, such that mistakes in predicting one class are more heavily penalized than others.

On a final important note, when comparing two models, the same performance metric needs to be used while making sure they are tested on the same unseen samples.

1.2.3 Generalization and model complexity

Generalization is the ultimate outcome of successful learning [10, 11]. This consists of the ability to accurately perform a task T on a new experience E' while achieving a certain level of performance (measured by a given metric P) after being trained with experience E. The learner does not have to know all the possible experience E in order to perform well. Machine learning techniques prove most powerful in situations where they can shed light on new problems, never encountered before and produce testable predictions.

Complex models are not necessarily better than simple ones. Figure 1.6 plots the error on the experience E used for training (training error) and the error on new experience E' (test error) against the complexity of the learned model. The training error decreases with increasing model complexity. Initially this is accompanied by a decrease in test error. However, when our model becomes overly complex, it overfits the training experience E, which essentially means it follows the errors, or noise, in the training experience E too closely, leading to an increase in test error. Ultimately, we want to find the optimal level of model complexity in order to avoid overfitting.

1.2.4 Parameter optimization

A model, and its complexity, are determined by the number of used parameters. Optimizing those values is needed in order to avoid overfitting and thus obtain the best possible model. This problem is also known as parameter optimization.

One approach for testing different parameter configurations uses grid search, which is simply an iteration over all possible value combinations. Quite often, the curse of dimensionality is unavoidable while optimizing the parameters. To illustrate, if only 3 parameters need to be optimized, each taking on n values, this explodes to n^3 possibilities. Another approach, Surragote-Based Optimization (SBO), tackles this issue and constructs a compact model (a surrogate) for the evaluation metric that is locally accurate in the regions of interest (the optimum) trying out a limited number of values for the parameters.

To detect the best parameter configuration, one can perform cross-validation on the training experience E whereby one part is used for building the model and the other part for evaluation, or one can evaluate using a new experience E'' (different from the experience E') for the ultimate testing, this is the so-called held-out validation set. The parameter configuration leading to the highest value of the performance metric P on this validation set is chosen.

1.3 Research goals and outline

A major problem with existing publicly available data sets in NILM is that all of them showcase major limitations, a.o., incorrect event annotations, undocumented preprocessing applied to the data, unavailability of submetered or contextual data, different power line frequency, inadequate sampling resolution, etc. **Can data be collected in a consistent way, and in addition be easily extendable?** In Chapter 2, we describe the collection process of a high frequency data set with accurate annotations of the ground truth. Appliances representing different types and categories (e.g., resistive, reactive, inductive loads or a combination thereof) are used.

In the present thesis, the focus lies on event based methods. The first step consists of finding the events, often by using the power signal. Event detection methods for this signal are usually based on fast edge detection algorithms. Well-known statistical methods are, e.g., Generalized Likelihood Ratio (GLR), where the statistical distribution of samples in 2 adjacent sliding windows is compared. Although these algorithms have shown to result in good overall performance, it was also found that these methods are rather sensitive to specific parameter settings within the algorithm, such as the threshold of the test statistic, the sliding window size, the power base load, the noise level of the data, and settings of the noise filter. **How can we develop event detection methods that are robust against parameter changes?** Chapter 3 clarifies this problem and offers two event detection methods that are robust against these parameter settings: the voting χ^2 method and the cepstrum method.

After the detection of events, these are characterized by features and the appliances responsible for the events are identified. A general overview of different features can be found in [12], showcasing the ongoing difficulties and challenges in distinguishing household appliances with similar electrical characteristics. **The question which features are best to characterize the appliances remains**. In Chapter 4, the VI trajectory is used for describing the post event steady-state behaviour. Instead of extracting features, such as asymmetry, and looping direction from this trajectory (as in [13]), the image itself is used as a feature. To this end, we first use a classic object recognition method based on contours and elliptical Fourier descriptors. Second, the image is classified using convolutional neural networks. In many cases, this classification task is complicated by data imbalances caused by differences in the total number of measurements per individual appliance type. **How does this appliance imbalance influence the performance of the classifier and do approaches that address the imbalance improve the performance?** Several approaches will be discussed in Chapter 4 and benchmarked in a systematic way.

A lot of classification algorithms described in literature, and the ones presented in Chapter 3, are unable to handle unlabelled data corresponding to previously unseen appliances. In practice, such events will be assigned a label and a power consumption corresponding to the appliance with the most similar features. Methods capable of handling unseen appliances (corresponding to unlabelled classes) are related to unsupervised methods. In clustering, for example, one could argue that points assigned to a cluster correspond to the appliance label of that cluster, whereas outliers represent novel categories of appliances, not observed before. None of the present clustering algorithms described in NILM research exploit this capability to detect previously unseen appliances, and none are capable of clustering with high accuracy on appliance-level. For the latter, it is necessary for features to occur in clearly separated groups. **How can a feature space be learned in such a way that different appliance types form different clusters?** Chapter 5 presents the use of siamese neural networks that are capable of learning a new feature space wherein different appliances are clearly clustered. Additionally, a workflow is presented that is capable of recognizing an unseen appliance when it is projected onto the newly learned feature space.

1.4 Publications

1.4.1 Journal papers

- **De Baets, L.**, Ruyssinck, J., Develder, C., Dhaene, T., & Deschrijver, D. (2017). "On the Bayesian optimization and robustness of event detection methods in NILM". *Energy and Buildings*, 145, 57-66.

- **De Baets, L.**, Ruyssinck, J., Develder, C., Dhaene, T., & Deschrijver, D. (2018). "Appliance classification using VI trajectories and convolutional neural networks". *Energy and Buildings*, 158, 32-36.

- **De Baets, L.**, Develder, C., Dhaene, T., & Deschrijver, D. "Detection of unidentified appliances in non-intrusive load monitoring using siamese neural networks". submitted to *International Journal of Electrical Power & Energy Systems*.

- **De Baets, L.**, Jingkun, G., Develder, C., Dhaene, T., Berges, M., & Deschrijver, D. "PLAID: Plug Load Appliance Identification Dataset". submitted to *Scientific Data*.

- Vansteenkiste*, T., Ruyssinck*, J., **De Baets, L.**, Decruyenaere, J., De Turck, F., Ongenae, F., & Dhaene, T. "Early detection of sepsis through the prediction of positive blood cultures using long short-term memory neural networks". Submitted to *Artificial Intelligence in Medicine*, (*Contributed equally)

1.4.2 Conference papers

- **De Baets, L.**, Van Gassen, S., Dhaene, T., & Saeys, Y. (2015). "Unsupervised trajectory inference using graph mining". In the proceedings of the *International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics*, Naples (Italy), Springer International Publishing, pp. 84–97.

- **De Baets, L.**, Ruyssinck, J., Deschrijver, D., & Dhaene, T. (2016). “Event detection in NILM using cepstrum smoothing”. In the proceedings of the *3rd International Workshop on Non-Intrusive Load Monitoring*, Vancouver (Canada), pp. 1–4.

- van der Herten, J., Depuydt, F., **De Baets, L.**, Deschrijver, D., Strobbe, M., Develder, C., Dhaene, T., Bruneliere, R., & Rombouts, J.-W. (2016). “Energy Flexibility Assessment of an Industrial Coldstore Process”. In the proceedings of the *IEEE International Energy Conference*. Leuven (Belgium), pp. 1–6.

- **De Baets, L.**, Ruyssinck, J., Deschrijver, D., & Dhaene, T. (2016). “Cepstrum analysis applied on event detection in NILM”. In the proceedings of the *25th Belgian-Dutch Conference on Machine Learning*, Kortrijk (Belgium), pp. 1–3.

- **De Baets, L.**, Ruyssinck, J., Develder, C., Dhaene, T., & Deschrijver, D. (2017) “Optimized Statistical Test for Event Detection in Non-Intrusive Load Monitoring”. In the proceedings of the *IEEE International Conference on Environment and Electrical Engineering and IEEE Industrial and Commercial Power Systems*, Palermo (Italy), pp. 1–5.

- **De Baets, L.**, Develder, C., Deschrijver, D., & Dhaene, T. (2017). “Automated classification of appliances using elliptical fourier descriptors”. In the proceedings of the *IEEE International Conference on Smart Grid Communications*, Dresden (Germany), pp. 1–6.

- **De Baets, L.**, Jingkun, G., Develder, C., Dhaene, T., Berges, M., & Deschrijver, D. (2017). “Handling imbalance in an extended PLAID”. In the proceedings of the *5th IFIP Conference on Sustainable Internet and ICT for Sustainability*, Funchal (Portugal), pp. 1–5.

1.4.3 Abstracts & posters

- **De Baets, L.**, Dhaene, T., & Saeys, Y. (2015). “Computational inference of developmental chronology using flow cytometry data”. In the proceedings of the *30th Congress of the International Society for Advancement of Cytometry*, Glasgow (Scotland) ,pp. 165-165.

- **De Baets, L.**, Cannoodt, R., Ruyssinck, J., De Preter, K., Dhaene, T., & Saeys, Y. (2014). “Identifying novel neuroblastoma oncogenes using semi-supervised learning”. Presented on *9th Benelux Bioinformatics conference proceedings*. Esch-sur-Alzette (GD Luxembourg).

- **De Baets, L.**, Ruyssinck, J., Develder, C., Deschrijver, D., & Dhaene, T. (2016). " 'Appliance recognition from VI-trajctor images using neural nets'. Presented on *the 2016 European NILM Workshop*, London (United Kingdom).
- **De Baets, L.**, Ruyssinck, J., Peiffer, T., Decruyenaere, J., De Turck, F., Ongenae, F., Dhaene, T. (2016). "Positive blood culture detection in time series data using a BiLSTM network". Presented on *NIPS Workshop on Machine Learning for Clinical Data Analysis*, Barcelona (Spain).
- **De Baets, L.**, Ruyssinck, J., Develder, C., Deschrijver, D., & Dhaene, T. (2017). "Automated classification of appliances using elliptical Fourier descriptors". Presented on *the 2017 European NILM Workshop*, London (United Kingdom).

References

[1] *European Commission,Climate Strategies & Targets*. http://ec.europa.eu/clima/policies/strategies/2030/index_en.htm. Accessed: 2016-08-01.

[2] Karen Ehrhardt-Martinez, Kat A Donnelly, Skip Laitner, et al. *Advanced metering initiatives and residential feedback programs: a meta-review for household electricity-saving opportunities*. American Council for an Energy-Efficient Economy Washington, DC, 2010.

[3] Shiva R Iyer, Maathangi Sankar, P Venkata Ramakrishna, Venkatesh Sarangan, Arunchandar Vasan, and Anand Sivasubramaniam. *Energy disaggregation analysis of a supermarket chain using a facility-model*. Energy and Buildings, 97:65–76, 2015.

[4] K Carrie Armel, Abhay Gupta, Gireesh Shrimali, and Adrian Albert. *Is disaggregation the holy grail of energy efficiency? The case of electricity*. Energy Policy, 52:213–234, 2013.

[5] José M Alcalá, Jesús Ureña, Álvaro Hernández, and David Gualda. *Assessing Human Activity in Elderly People Using Non-Intrusive Load Monitoring*. Sensors, 17(2):351, 2017.

[6] George William Hart. *Nonintrusive appliance load monitoring*. Proceedings of the IEEE, 80(12):1870–1891, 1992.

[7] Andreas Reinhardt, Paul Baumann, Daniel Burgstahler, Matthias Hollick, Hristo Chonov, Marc Werner, and Ralf Steinmetz. *On the accuracy of appliance identification based on distributed load metering data*. In Sustainable Internet and ICT for Sustainability (SustainIT), 2012, pages 1–9. IEEE, 2012.

[8] Stephen Makonin. *Approaches to Non-Intrusive Load Monitoring (NILM) in the Home*. Technical report, 2012.

[9] Tom M Mitchell et al. *Machine learning. WCB*, 1997.

[10] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[11] Stuart Russell and Peter Norvig. *Artificial Intelligence: A modern approach*. Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs, 25:27, 1995.

[12] Michael Zeifman and Kurt Roth. *Nonintrusive appliance load monitoring: Review and outlook*. IEEE transactions on Consumer Electronics, 57(1), 2011.

[13] Hong Yin Lam, GSK Fung, and WK Lee. *A novel method to construct taxonomy electrical appliances based on load signatures*. IEEE Transactions on Consumer Electronics, 53(2), 2007.

2 Datasets

"It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts."
– Sherlock Holmes, *A Scandal in Bohemia*

De Baets, L., Jingkun, G., Develder, C., Dhaene, T., Berges, M., & Deschrijver, D. "PLAID: Plug Load Appliance Identification Dataset", submitted to Scientific Data.

2.1 Introduction

The first step in NILM is gathering the data, see Figure 2.1. Data can be gathered by (1) consulting public datasets, or (2) measuring and annotating it yourself. A major problem with existing publicly available datasets in NILM is that all of them showcase limitations, a.o., incorrect event annotations, undocumented preprocessing applied to the data, unavailability of submetered or contextual data, different power line frequency, inadequate sampling resolution, etc. This chapter presents new data that diminish some limitations.

The Plug-Level Appliance Identification Dataset (PLAID) is a public dataset consisting of voltage and current measurements from different electrical household appliances sampled at 30 kHz. All appliances are monitored *individually*: they are

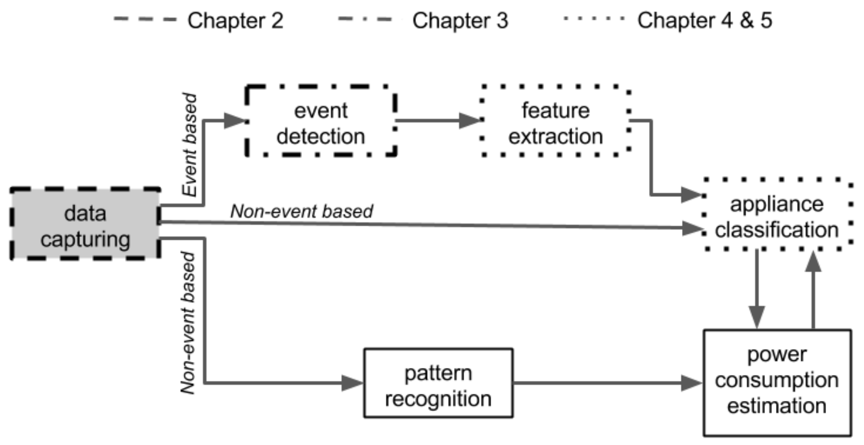


Figure 2.1: A schematic overview of the steps in NILM, situating this chapter.

submetered and the data traces captured over a few seconds include the activation of the appliances. Additionally, some of them are also monitored when active *simultaneously*: their aggregated consumption is measured and the data captured over a few minutes contains the activation and deactivation of a subset of the appliances. Activations and deactivations are characterized by events in the current and voltage signals.

In total, 17 different appliance types (e.g., refrigerators, microwave ovens, etc.) are measured in 65 different locations for the submetered data, and 13 different appliance types (a subset from those used for the submetered data) are measured at one single location for the aggregated data. Not all appliance types are available in all different locations. In total, the dataset contains 330 different appliances (i.e., different appliance models for each of the 17 different appliance types). For some appliances (approximately 10% of them), multiple operating modes were monitored.

The dataset has grown over the years: [1] published in 2014 and [2] published in 2017 contain, respectively, 71% and 25% of the currently available submetered data. This paper compiles all of the previous PLAID publications and presents some additional data comprising another 4% of submetered data, as well as extra aggregated data. The added appliances and location are different as compared to previous versions, and include new appliance types. Our goal is to continuously expand this dataset by incorporating additional measurements of appliances at different locations. To facilitate this goal, this paper describes the technical procedure to consistently replicate the setup.

PLAID can be used in two ways. First, the high resolution submetered appliance measurements (30 kHz) can be used to automate the labelling of submetered data, enabling the possibility for appliance classification (i.e., being able to classify appliance types from just voltage and current measurements). This knowledge is interesting for smart plugs [3] that are used for smart grid and building-level energy management applications such as automated load control [4] and load scheduling [5]. In addition to appliance classification, this data can also be used to create an appliance power consumption inventory. As the submetered data is captured in different houses, the generalization of the labelling methods across houses can be tested. Second, the high resolution aggregated appliance measurements (30 kHz) can be used to learn how to dissagregate the total current consumption measured at the main feed of a household at high frequency. This is known as non-intrusive load monitoring (NILM) [6]. Two important steps in NILM are event detection [7] and load identification [8]. This dataset provides the means to learn and implement both tasks on high frequency data. The obtained information can also be used to identify energy consumption and to monitor the deterioration of appliances.

Table 2.1 shows similar datasets that are publicly available. PLAID is distinct because it contains submetered and aggregated data sampled at a frequency higher than 1 Hz. Only two other datasets (WHITED [9] and COOLL [10]) contain submetered data sampled at a frequency higher than 1 Hz. All the others, like ACS-F2 [11] and Tracebase [12], contain submetered data sampled at a frequency lower than 1 Hz. From these last datasets, only two, i.e., REDD [13], and UK-DALE [14], contain aggregated data sampled at a frequency higher than 1Hz. All the others, i.e. DRED [15], Dataport [16], REFIT [17] and AMPds2 [18] contain aggregated data sampled at a frequency lower than 1 Hz.

2.2 Methods

First, the hardware used to monitor the appliances is described. Next, we describe the selected appliances and their occurrence in the different households. The last two subsections explain how the appliances are submetered and aggregated.

2.2.1 Monitoring set-up

All electrical measurements were was collected using a National Instruments (NI-9215) data acquisition card [19]. The NI-9215 includes four simultaneously sampled analog input channels paired with a 16-bit analog-to-digital converter (ADC) that we use to collect voltage and current measurements. These are stored in a computer via a USB connection, as shown in Figure 2.2.

	Sampling Frequency				Appliance	# of buildings
	Submetered		Aggregated		operating modes	
	$< 1\,\mathrm{Hz}$	$\geq 1\,\mathrm{Hz}$	$< 1\,\mathrm{Hz}$	$\geq 1\,\mathrm{Hz}$		
PLAID		✓		✓	multiple	65
WHITED [9]		✓			on, off	
COOLL [10]		✓			on, off	
ACS-F2 [11]	✓				on, off	
Tracebase [12]	✓				on, off	
REDD [13]	✓			✓	on, off	2
UK-DALE [14]	✓			✓	on, off	6
DRED [15]	✓		✓		on, off	
Dataport [16]	✓		✓		on, off	1200+
REFIT [17]	✓		✓		on, off	20
AMPds2 [18]	✓		✓		on, off	

Table 2.1: An overview of PLAID and similar datasets in terms of submetered data sampled at a frequency $< 1\,Hz$ or $\geq 1\,Hz$, aggregated data sampled at a frequency $< 1\,Hz$ or $\geq 1\,Hz$, different appliance operating modes and the number of different buildings.

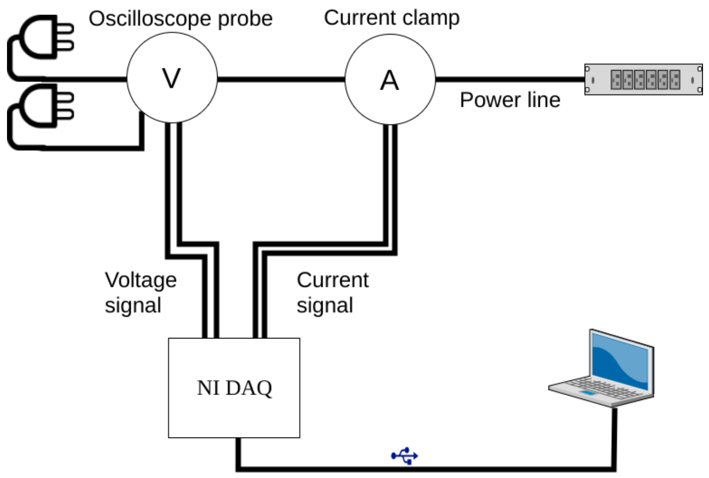


Figure 2.2: The measurement set-up for capturing the data.

To measure the different appliances, these were connected to the power strip. This power strip has a negligible amount of power consumption as a small lamp was burning indicating the activity of the power strip. As a consequence, this small load is measured during the data collection. From this power strip, the current and voltage are measured.

Current is measured with a Fluke i200 AC current clamp [20] that has a cut-off frequency of 10 kHz, allowing us to sample signals with frequency content up to 5 kHz according to the Nyquist-Shannon sampling theorem [21]. It is important to note that if the current is sampled at a high frequency, it is necessary to have a clamp with a high cut-off frequency. Some of the existing datasets with high sampling frequency did not account for this (e.g., BLUED [22] used a current transformer with a cut-off frequency of $\sim$ 300 Hz). The Fluke i200 is connected to the NI-9215, see Figure 2.2.

Voltage is measured with a Pico-TA041 Oscilloscope probe [23]. The TA041 is an active differential probe suitable for high common-mode voltage measurement applications up to $\pm$700 V (DC + peak AC). It can be used with signal frequencies of up to 25 MHz [23]. As it is an active probe, it requires power. Because the active probes significantly reduce capacitive loading, they are able to achieve fast signal measurements with much better signal fidelity making them well suited for high frequency measurements. As with the current clamp, the Pico-TA041 is connected to the NI-9215, see Figure 2.2.

The NI-9215 converts the analog voltage and current signals into digital signals and sends them via an USB-connection to a computer. Libraries for different programming languages (e.g., Python, C++, MATLAB, and LabVIEW) can be used to communicate with the NI-9215 under the condition that the correct drivers are installed. We used MATLAB and LabVIEW and stored the data in comma-separated values (CSV) files. Reference scripts for replicating this process are also made available as part of the dataset.

2.2.2 Selected homes and appliances

In total, 17 appliance types were measured at 65 locations. These include one lab environment and 64 households. These households were recruited via an email campaign and mainly consist of graduate student homes. All the households are located in Pittsburgh, Pennsylvania, USA.

Table 2.2 gives an overview of the 17 appliance types, their occurrence in the 65 locations (number of appliances) and the number of times these were monitored/activated (number of instances), both for the submetered and aggregated case. For example, for the refrigerator appliance type, 28 physically different refrigerators are monitored separately multiple times, leading to 100 instances of this appliance type. One of these refrigerators is monitored 79 times when other appliances were active or were turned on. For six appliances types that were located in the lab environment, only one appliance is monitored. Those appliances were also used to generate the aggregate measurements. Note, that there is less data of the blender appliance type compared to the other appliance types, as it broke down in the middle of the experiment.

All the appliances were activated by connecting them to the power strip and turning on the switch if present. However, the following remarks need to be given concerning activation assumptions:

- The blender was kept empty during the experiments.
- The refrigerator was activated after it warmed up by opening the door. This ensured the motor would activate.
- An unknown mode of the refrigerator was activated by plugging in the refrigerator twice shortly after each other. The second time, the unknown mode is activated.
- The soldering iron has a two-phase activation process: around 6 seconds after activation, there is an increase in power consumption. The two events are stored in two separate files, both with the label ‘soldering iron’.

	Submetered			Aggregated	
Appliance type (load)	# of appliances	# of instances	Operating modes	# of appliances	# of instances
AC (NL)	27	218	[high cool, high fan, low cool, low fan]	1	160
Blender (I)	1	2	[off, on]	1	51
Coffeemaker (R)	1	10	[off, on]	1	106
CFL (NL)	45	230	[off, on]	1	104
Fan (I)	31	220	[high, medium, low]	1	102
Refrigerator (I)	28	100	[off, on, unknown]	1	167
Hairdryer (R)	36	248	[high warm, low warm, high hot, low hot]	0	0
Hair iron (NL)	1	10	[off, on]	1	98
Heater (R)	15	85	[high, low]	0	0
ILB (R)	33	158	[off, on]	1	11
Laptop (NL)	46	217	[off, on]	1	90
Microwave oven (NL)	32	229	[high, medium]	0	0
Soldering iron (NL)	1	20	[off, on]	1	218
Vacuum cleaner (I)	15	83	[off, on]	1	98
WM (NL)	16	75	[off, on]	0	0
Water kettle (R)	1	10	[off, on]	1	109
Total		1925			1314

Table 2.2: Summary of the different appliances in PLAID. AC = air conditioner, CFL = compact fluorescent light, ILB = incandescent light bulb, WM = washing machine. R = resistive, I = inductive, NL = non-linear.

2.2.3 Submetered appliances

Each time an appliance is activated, a state transition (event) will happen [7]. When the appliances are monitored individually, i.e., submetered, the activation is measured together with some seconds of the steady state following this activation. This measurement captures the transient start-up containing information of the present electrical components and possible present inertia. The deactivation of the appliances is not measured because then the electrical circuit is disconnected and appliance specific information is no longer present. The recorded steady state duration ranges from 1 to 20 seconds.

Besides monitoring the activation of the appliances, the following meta-data is stored, when available:

- Manufacturing data of the appliance: the brand, manufacturing year, model number, appliance type (first column of Table 2.2), load type, and the rated current, voltage and power consumption values.
- Information concerning the data capturing process: the time of data collection expressed in month and year, the sampling frequency, the total measurement duration, and the specific operating mode that was measured.
- The location identifier, which is a string (e.g., 'house5' or 'CMU lab').

The current and voltage measurements themselves are stored in separate CSV files. The measurement is stored in two columns, one for the the current expressed in ampere and the other one for the voltage expressed in volt. The precision of the numbers is three decimals. As the sampling rate was kept constant, there was no need to associate each measurement with a timestamp. The time that has passed relative to the beginning of the file can be calculated using the sampling frequency (e.g., for a frequency of 30 kHz, the 30000th point occurs one second after the start).

The meta-data is stored in one big JavaScript Object Notation (json) file which contains for each measurement an attribute-value pair with the CSV file name of the measurement file, as attribute and the meta-data of the measurement in question as the value. The meta-data itself is also structured as attribute-value pairs, namely:

```
'appliance': {
     'brand': '',
     'current': '',
     'load':'',
     'manufacture_year': '',
     'model_number': '',
     'notes': '',
     'type': ''
     'voltage': '',
     'wattage': ''},
'header': {
     'collection_time': '',
     'notes': '',
     'sampling_frequency': ''},
'instances': {
     'length': '',
     'status': ''},
'location': ''}
```

2.2.4 Aggregated appliances

To measure the aggregated signals, several appliances are activated one after another. Different from the submetered case, the deactivation is also monitored. This is done because other appliances may still be running after deactivation. The 13 appliances that were present in the lab environment were used to create the aggregated data (see Table 2.2). The goal of this dataset is to capture the signal characteristics for combined operation of appliances. Full coverage of all the combinatorial possibilities would have been impractical. For instance, there are 312 combinations of two appliances that can be made from these 13[1]. Activating more than two appliances each in turn, becomes intractable as the number of combinations grows exponentially with the number of appliances.

To make the amount of combinations more tractable, the following division is used: appliance types can be linear (L) or non-linear (NL) loads. A load is linear if there is a linear relationship between its current drawn and the supplied voltage. Some loads, such as these containing transistors and other electronics, do not behave in this way and are called non-linear loads. The linear loads can be resistive (R), capacitive (C) or inductive (I). Examples of a resistive, capacitive, and inductive loads are respectively a light bulb, a battery, and a motor. An example of a non-linear load is a computer. The grouping for the appliances present in the lab are given in the first column of Table 2.2 between brackets. As can be seen, there are no purely capacitive loads available, leaving the following groups: R, I and NL. The following combinations in and between the groups are measured:

- *Two different appliances of the same group* are selected (e.g., A and B) and

[1]$(4 \cdot \binom{13}{2})$ combinations. The multiplication factor 4 refers to the different order in which 2 appliances can be activated and deactivated under the condition that first the 2 appliances must be activated before one can be deactivated.

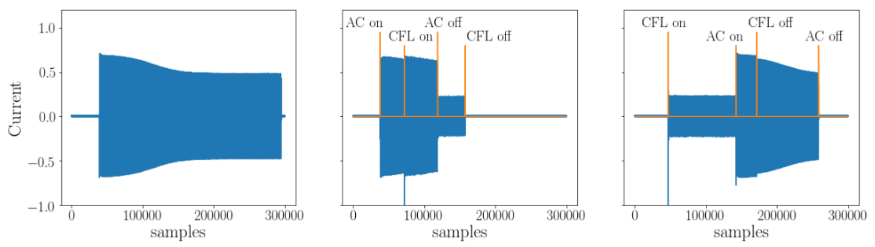


Figure 2.3: (a) The current consumption of an AC is shown (`submetered/9.csv`). (b) The CFL is activated during the transient of the AC (`aggregated/558.csv`). (c) The CFL is deactivated in the transient behaviour of the AC (`aggregated/559.csv`).

combined in all possible ways under the condition that first the two appliances must be activated before one can be deactivated. All possible selections of appliances A and B for each group are measured. For example, for the resistive group consisting of 4 appliances, there are 6 different selections of two appliances A and B, and each is combined in 4 ways, leading to 24 $(= 6 \cdot 4)$ measurements.

- *Two different appliances, each of a different group,* are selected and combined in all possible ways under the condition that first the two appliances must be activated before one can be deactivated (see above). All possible selections of two different appliances, each of a different group are measured. As the resistive, inductive and non-linear group consists of 4, 5, and 4 appliances respectively, this leads to 56 $(= 4 \cdot 5 + 4 \cdot 4 + 5 \cdot 4)$ selections of two different appliances. As each selection is combined in four possible ways, in total there are 224 $(= 56 \cdot 4)$ measurements. Note that some of the combinations with the blender are missing because it broke down before the end of the experiments.

- *Three different appliances, each from one group,* are selected and combined in a random way under the conditions that three appliances must be all activated before one is deactivated and that the order of activation is the same as deactivation. As the number of possible appliance selections and combinations is too large to cover exhaustively, a random generator is used to select the three appliances and their order. This is repeated 60 times.

Combining the appliances in this way allows us to investigate the influence that appliances of the same or different groups have on each other. Investigation of this data will point out if further elaborating this dataset is necessary. Each of these measurements is only done once.

A special case of aggregating appliances is when an appliance is (de)activated

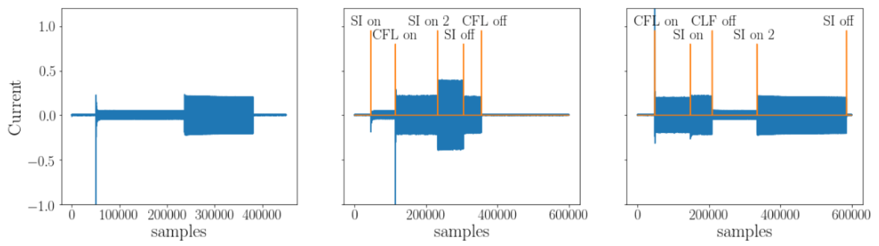


Figure 2.4: (a) The current consumption of the soldering iron (SI) is shown (`submetered/12.csv`). (b) The CFL is activated during the first phase of activation of the SI (`aggregated/484.csv`). (c) The CFL is deactivated during the first phase of activation of the SI. (`aggregated/485.csv`).

during the transient behavior of another appliance. In Figure 2.3a, an example is given of the transient behavior of the air conditioner. When an appliance is (de)activated during the transient phase, it is seen that its behavior before/after the event is different. The AC is the only appliance in PLAID with a sufficiently large and slow transient behavior that makes it possible to simultaneously (de)activate appliances. The other appliances (except for the blender, laptop charger, refrigerator and refrigerator defroster) were either activated or deactivated at 5 different random time instances during the transient of the AC. An illustration is shown in Figure 2.3b and 2.3c. In the end, 80 $(= 8 \cdot 5 + 8 \cdot 5)$ measurements for this special case are captured. This was not done for the blender as it already broke down and not for the laptop charger, refrigerator and refrigerator defroster as these appliances are activated by connecting the plug to the power line, and it was not feasible to accomplish this within the time frame wherein the transient behavior takes place.

Another special case is when the soldering iron with the two-phase activation process is used (see Figure 2.4a). In the previously described measurements, other appliances are only (de)activated when the soldering iron reached the second step of its activation. To complete the dataset, we also captured data where appliances are (de)activated during the first step of the soldering iron's activation. More specifically for an appliance A two measurements are captured in the following manner:

- Appliance A is activated between the first and second step of the soldering iron's activation. Once the activation of both appliances is complete, the soldering iron and A are deactivated each in turn, as shown in Figure 2.4b.
- Appliance A and the soldering iron are activated each in turn. Then, appliance A deactivated in between the first and second step of the soldering iron's activation, as shown in Figure 2.4c.

For each appliance type, the above measurements are only done once, as repeating the experiments would result in almost identical events, since the time and the current consumption between the two activation steps is always the same. This is done for every other appliance, resulting in 24 $(= 2 \cdot 12)$ measurements.

As mentioned earlier, the measurements are stored in CSV files. Table 2.3 gives an overview of the files corresponding to each experiment. The meta-data follows the same structure as for the submetered data and extends it by adding an array of appliances monitored in the file. Each appliance is characterized by its manufacturing data (see meta-data of submetered data), and timestamps of activation and deactivation. The timestamps are expressed using indices from which the time passed since the start of the file can be calculated using the known sampling frequency of 30 kHz. The index represents the moment the appliance is activated and not the moment the appliance reaches steady state. Note that the soldering iron induces two events when it is activated, one for each activation phase, and both are labelled. Just as for the meta-data of the submetered data, the meta-data of the aggregated data is structured as attribute-value pairs. The additions are put in italic:

```
'appliances': [{
      'brand': '',
      'current': '',
      'load': '',
      'manufacture_year': '',
      'model_number': '',
      'notes': '',
      'on': '',
      'off': '',
      'type': '',
      'voltage': '',
      'wattage': ''}, ... ],
'header': {
      'collection_time': '',
      'notes': '',
      'sampling_frequency': ''},
'instances': {
      'length': '',
      'status': ''},
'location': ''}
```

files	experiment
1 - 474	2 or 3 appliances active, on/off outside transient, on/off in second activation phase of soldering iron
475 - 554	AC and other appliance, on/off during AC transient, on/off in second activation phase of soldering iron
555 - 576	Soldering iron and other appliance, on/off outside transient, on/off in first activation phase of soldering iron

Table 2.3: An overview of which files correspond to which experiment.

2.2.5 Known issues

Some issues are present in PLAID. When monitoring the appliances individually in the 2014 version (the submetered files with identifiers going from 1 to 1074), the calibration was not checked every time when the set-up changed places. As an example, the histogram in Figure 2.5 shows the distribution of maximal current and voltage values for the vacuum appliance type, indicating a great variation in the values as the maximal current values range from 0.54 A to 34.65 A and the maximal voltage values range from 9.14 V to 171.8 1V. Some of the variance in the values can be explained by the fact that there are 15 different vacuum cleaners, but the smallest values suggest a calibration error. As a consequence, a data normalization step is needed for further processing. This must be done by the user. Two possibilities to normalize the data are rescaling and standardization. The last one is more appropriate when there are outliers in the data.

Table 2.2 also shows that the data is very imbalanced (e.g., 85 instances for the heater appliance type compared to the 230 instances for the compact fluorescent lamp appliance type). This imbalance needs to be considered in evaluation of, e.g., automatic classification [2].

An additional minor issue is that the meta-data concerning the manufacturing of the appliances is quite often left blank by the measurer, as can been seen in Table 2.4. Having this information could be valuable for comparing the power consumption between different generations of appliances or different brands.

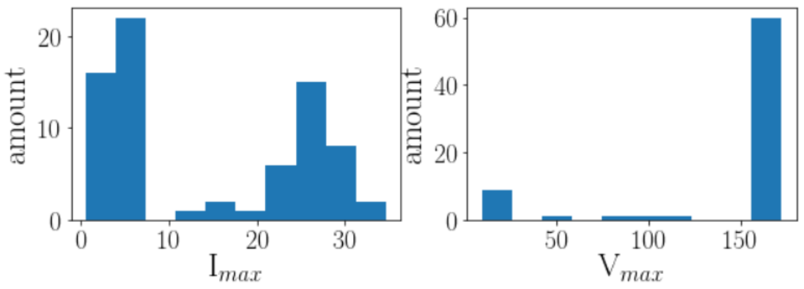


Figure 2.5: The histograms of maximal current and voltage values for the measured vacuum cleaners in steady state.

Meta-data	Submetered #/Total (%)		Aggregated #/Total (%)	
brand	824/1925	(42.81%)	1254/1305	(96.09%)
current consumption	450/1925	(23.38%)	759/1305	(58.16%)
manufacturing year	24/1925	(1.25%)	0/1305	(0.00%)
model number	582/1925	(30.23%)	90/1305	(6.9%)
on	N.A.		1305/1305	(100%)
off	N.A.		1305/1305	(100%)
voltage consumption	655/1925	(34.03%)	1087/1305	(83.30%)
wattage	453/1925	(23.53%)	700/1305	(53.64%)
capturing moment	1925/1925	(100%)	576/576	(100%)
sampling frequency	1925/1925	(100%)	576/576	(100%)
total time	1925/1925	(100%)	576/576	(100%)
measured mode	1925/1925	(100%)	576/576	(100%)
location identifier	1925/1925	(100%)	576/576	(100%)
appliance type	1925/1925	(100%)	1305/1305	(100%)

Table 2.4: The number of instances for which the metadata fields are completed. Note that for the aggregated data, the total number of instances for the manufacturing meta-data is larger than for the other meta-data, this because multiple appliances can be activated at the same time.

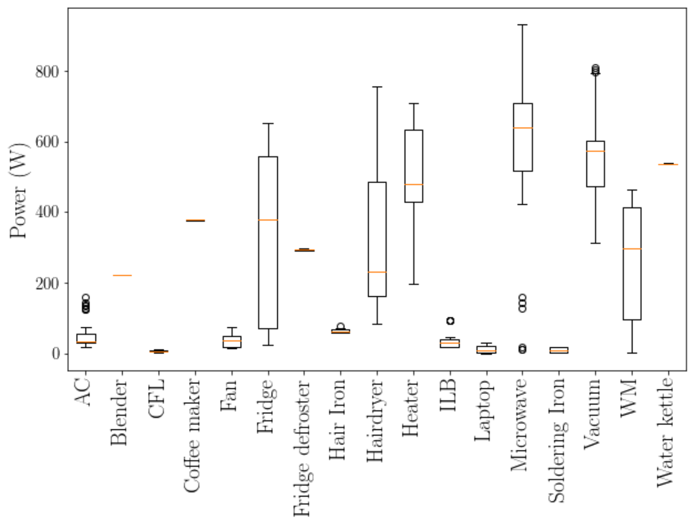


Figure 2.6: The power draw (W) of the appliances present in the dataset. Per appliance type, a boxplot of the power consumption is shown.

2.3 Technical validation

PLAID can be used for different use cases that involve appliance recognition from electrical data. An advantage of this dataset is that the same appliance type is measured in different houses. In this section, we check if different appliances of the same type have a similar power profile, using the submetered data that was correctly calibrated. This can give insight whether or not it is justified to combine data from different brands within the same appliance type.

In Figure 2.6, the active power consumption for the appliance types is shown. The active power for one cycle is calculated from the current and voltage signal in the following manner:

$$P = \frac{1}{n}\sum_{i=1}^{k} I_i V_i \tag{2.1}$$

where I_i and V_i are respectively the ith sample of current and voltage of a steady state cycle of respectively the current and voltage. Figure 2.6 shows that the power draw of same type appliances between different brands can vary significantly. For example, the power consumption of the microwave oven varies from around 500 W to 1000 W. This implies that the appliance recognition generalization on different houses will not be straightforward and other features must be examined as well.

2.4 Usage notes

The PLAID data is provided in CSV files and can be extracted using the common programming languages and software packages (e.g., Python, MATLAB).

As mentioned earlier, the dataset concerning the submetered data has grown over time. The data of [1, 24] corresponds to file identifiers 1-1074, while [2] uses 1-1793. For this paper, additional submetered data is captured, which is stored in files 1794-1925. Note that only for the files 1075-1793, multiple operating modes are considered (not only binary on/off).

References

[1] Jingkun Gao, Suman Giri, Emre Can Kara, and Mario Bergés. *PLAID: a public dataset of high-resoultion electrical appliance measurements for load identification research: demo abstract*. In proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings, pages 198–199. ACM, 2014.

[2] Leen De Baets, G Jingkun, Chris Develder, Tom Dhaene, M Berges, and Dirk Deschrijver. *Handling imbalance in an extended PLAID*. In Proceedings of the 5th IFIP Conference on Sustainable Internet and ICT for Sustainability (SustainIT), pages 1–5, 2017.

[3] Hicham Elzabadani, A Helal, Bessam Abdulrazak, and Erwin Jansen. *Self-sensing spaces: smart plugs for smart environments*. In Proceedings of the 3rd International Conference on Smart Nomes and Health Telematics, pages 91–98, 2005.

[4] Amir-Hamed Mohsenian-Rad and Alberto Leon-Garcia. *Optimal residential load control with price prediction in real-time electricity pricing environments*. IEEE transactions on Smart Grid, 1(2):120–133, 2010.

[5] Pengwei Du and Ning Lu. *Appliance commitment for household load scheduling*. IEEE transactions on Smart Grid, 2(2):411–419, 2011.

[6] I Abubakar, SN Khalid, MW Mustafa, Hussain Shareef, and M Mustapha. *Application of load monitoring in appliances energy management–A review*. Renewable and Sustainable Energy Reviews, 67:235–245, 2017.

[7] Leen De Baets, Joeri Ruyssinck, Chris Develder, Tom Dhaene, and Dirk Deschrijver. *On the Bayesian optimization and robustness of event detection methods in NILM*. Energy and Buildings, 145:57–66, 2017.

[8] Leen De Baets, Joeri Ruyssinck, Chris Develder, Tom Dhaene, and Dirk Deschrijver. *Appliance classification using VI trajectories and convolutional neural networks*. Energy and Buildings, 158:32–36, 2018.

[9] Matthias Kahl, Anwar Ul Haq, Thomas Kriechbaumer, and Hans-Arno Jacobsen. *Whited-a worldwide household and industry transient energy data set*. In Proc. 3rd International Workshop on Non-Intrusive Load Monitoring, 2016.

[10] Thomas Picon, Mohamed Nait Meziane, Philippe Ravier, Guy Lamarque, Clarisse Novello, Jean-Charles Le Bunetel, and Yves Raingeaud. *COOLL:*

Controlled On/Off Loads Library, a Public Dataset of High-Sampled Electrical Signals for Appliance Identification. arXiv preprint arXiv:1611.05803, 2016.

[11] Antonio Ridi, Christophe Gisler, and Jean Hennebert. *ACS-F2A new database of appliance consumption signatures*. In Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of, pages 145–150. IEEE, 2014.

[12] Andreas Reinhardt, Paul Baumann, Daniel Burgstahler, Matthias Hollick, Hristo Chonov, Marc Werner, and Ralf Steinmetz. *On the accuracy of appliance identification based on distributed load metering data*. In Sustainable Internet and ICT for Sustainability (SustainIT), 2012, pages 1–9. IEEE, 2012.

[13] J Zico Kolter and Matthew J Johnson. *REDD: A public data set for energy disaggregation research*. In Workshop on Data Mining Applications in Sustainability (SIGKDD), San Diego, CA, volume 25, pages 59–62, 2011.

[14] Jack Kelly and William Knottenbelt. *The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes*. Scientific data, 2:150007, 2015.

[15] Akshay SN Uttama Nambi, Antonio Reyes Lua, and Venkatesha R Prasad. *Loced: Location-aware energy disaggregation framework*. In Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments, pages 45–54. ACM, 2015.

[16] Oliver Parson, Grant Fisher, April Hersey, Nipun Batra, Jack Kelly, Amarjeet Singh, William Knottenbelt, and Alex Rogers. *Dataport and NILMTK: A building data set designed for non-intrusive load monitoring*. In Signal and Information Processing (GlobalSIP), 2015 IEEE Global Conference on, pages 210–214. IEEE, 2015.

[17] David Murray, Lina Stankovic, and Vladimir Stankovic. *An electrical load measurements dataset of United Kingdom households from a two-year longitudinal study*. Scientific data, 4:160122, 2017.

[18] Stephen Makonin, Bradley Ellert, Ivan V Bajić, and Fred Popowich. *Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014*. Scientific data, 3:160037, 2016.

[19] *The National Instruments (NI-9215) data acquisition card*. https://www.ni.com/data-acquisition/. Accessed: 2018-02-8.

[20] *The Fluke i200*. https://en-us.fluke.com/products/all-accessories/Fluke-i200s.html. Accessed: 2018-02-8.

[21] Claude Elwood Shannon. *Communication in the presence of noise*. Proceedings of the IRE, 37(1):10–21, 1949.

[22] Kyle Anderson, Adrian Filip, Diego Benitez, Derrick Carlson, Anthony Rowe, and Mario Berges. *BLUED: A fully labeled public dataset for event-based non-intrusive load monitoring research*. In 2nd Workshop on Data Mining Applications in Sustainability (SustKDD), page 2012, 2011.

[23] *The Pico-TA041 Oscilloscope probe*. https://www.picotech.com/accessories/active-oscilloscope-probes/25-mhz-700-v-differential-probe. Accessed: 2018-02-8.

[24] Jingkun Gao, Emre Can Kara, Suman Giri, and Mario Bergés. *A feasibility study of automated plug-load identification from high-frequency measurements*. In Signal and Information Processing (GlobalSIP), 2015 IEEE Global Conference on, pages 220–224. IEEE, 2015.

3
Event Detection

"You see, but you do not observe. The distinction is clear."
– Sherlock Holmes, *A Scandal in Bohemia*

De Baets, L., Ruyssinck, J., Develder, C., Dhaene, T., & Deschrijver, D. (2017). "On the Bayesian optimization and robustness of event detection methods in NILM". Energy and Buildings, 145, 57-66

De Baets, L., Ruyssinck, J., Develder, C., Dhaene, T., & Deschrijver, D. (2017) "Optimized Statistical Test for Event Detection in Non-Intrusive Load Monitoring". In the proceedings of the IEEE International Conference on Environment and Electrical Engineering and IEEE Industrial and Commercial Power Systems, Palermo (Italy), pp. 1–5.

De Baets, L., Ruyssinck, J., Deschrijver, D., & Dhaene, T. (2016). "Event detection in NILM using cepstrum smoothing". In the proceedings of the 3rd International Workshop on Non-Intrusive Load Monitoring, Vancouver (Canada), pp. 1–4.

De Baets, L., Ruyssinck, J., Deschrijver, D., & Dhaene, T. (2016). "Cepstrum analysis applied on event detection in NILM". In the proceedings of the 25th Belgian-Dutch Conference on Machine Learning, Kortrijk (Belgium), pp. 1–3.

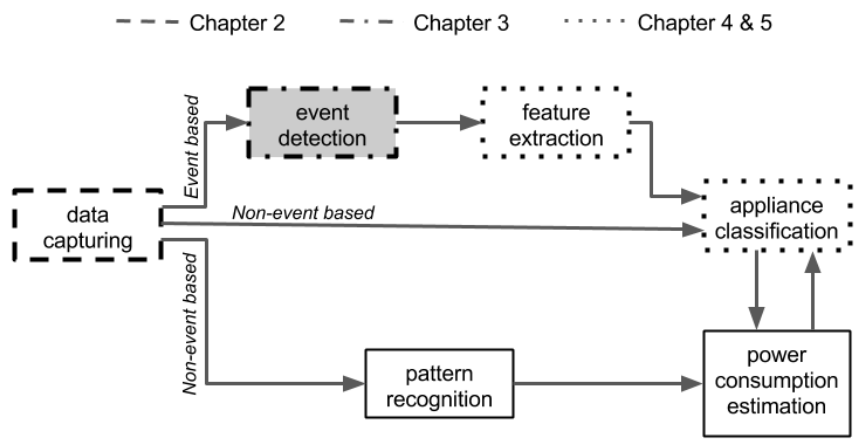


Figure 3.1: A schematic overview of the steps in NILM, situating this chapter.

3.1 Introduction

This chapter focuses on the development of event detection methods in NILM, see Figure 3.1. An important property of these methods is their robustness against differences in the base load (i.e., the background consumption of devices already consuming power when the event to be detected occurs): performance decay if high power consuming devices are on is unwanted. To the best of our knowledge, this property has not yet been thoroughly investigated. This chapter shows that one of the most commonly used event detection method, chi-squared goodness-of-fit (χ^2 GOF), lacks this robustness. Two alternative methods robust to changes in the base load are proposed. The first method is an adapted version of the standard χ^2 GOF method [1], which is extended with a voting mechanism [2]. The second robust method is a new method that uses smoothed frequency components to detect events.

A second contribution of this chapter concerns parameter optimization of event detection methods. A standard but slow approach is a brute-force exhaustive search that tries out all the possible parameter configurations and selects the best one. In this chapter, this process is optimized by introducing Surrogate-Based Optimization (SBO) [3].

The remainder of the chapter is structured as follows: Section 3.2 introduces a brief overview of related work, Section 3.3 specifies the preprocessing of the input. Section 3.4 describes the state-of-the art NILM event detection method, discusses its lack of robustness towards base load power differences and proposes a robust adaptation, the voting χ^2 GOF method. Section 3.5 discusses the newly proposed robust event detection method. In Section 3.6, a SBO algorithm is proposed to identify optimal model parameter configurations for the statistical tests and in Section 3.7, the performance of the newly presented methods is benchmarked. A conclusion is presented in Section 3.8.

3.2 Related work

Event detection methods In 1992, together with the first event-based work-flow for NILM, Hart described an event detection method that relied on monitoring changes in active and reactive power [4]. A better method to detect events in the active power is the Generalized Likelihood Ratio Test (GLRT) [2, 5], which tests if two neighboring windows representing consecutive time frames share a common distribution. The possible presence of an event in two neighboring windows is determined by calculating a decision statistic from the natural log of a ratio of probability density functions in those neighbouring windows. Another method is the χ^2 GOF test. It detects events by assuming, like the GLRT, that two neighbouring windows share a common distribution. A χ^2 test statistic is applied on two neighbouring windows and an event is assumed if the null hypothesis is rejected [1]. This χ^2 GOF test is widely used [1, 5–7] and reasons to adopt this method are its simplicity and improved performance as reported in other studies [1, 6–8].

In addition to these heuristics, more computationally expensive machine learning algorithms are available. Hidden Markov Models (HMMs) [9] formulate the problem of detecting events as finding an ideal set of non-overlapping intervals in which the observations are as heterogeneous as possible. These unknown intervals are the hidden states of the HMM. A disadvantage is that the number of states needs be given or must be predicted. Support vector machines (SVMs) [10] fit models on short segments of the signal, all learned simultaneously using a coupling term that forces neighbouring models to be similar. Bayesian methods [11] work by estimating the run length at every data point. The run length represents the time since the last event. The run length can be inferred given (1) an underlying predictive model whose parameters change when an event occurs, and (2) a hazard function which describes how likely an event is, given an observed run length. This hazard function needs to be given, but the model can be created if it is assumed that the data in each segment is independent and identically distributed (i.i.d.) with respect to some distribution. It must be noted that for HMMs and SVMs, the event detection is a side effect of the approach, and not a separate module in the algorithm itself.

This chapter focuses on the χ^2 GOF method as event detection method as it is widely used, simple, and has good performance [1, 5–8].

Benchmark dataset The most commonly used benchmark for event detection is the BLUED dataset, consisting of the aggregated active power signal from a family residence in the United States for a whole week [12]. In this dataset the steady-state power consumption never exceeds 500W for phase A and 1500W for phase B. However, it is likely that much higher power values occur in households, e.g., when electrical heaters (easily consuming 1500W) are used. This chapter will show that the performance of the χ^2 GOF method decreases rapidly if a base load is added to the power consumption, while the performance of the proposed voting χ^2 GOF and cepstrum method remains the same under similar conditions.

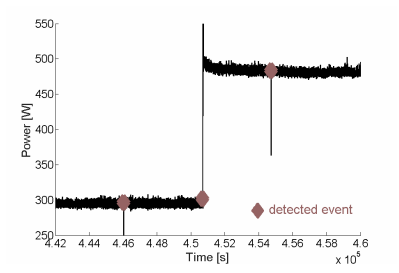
Parameter optimisation The χ^2 GOF method, the proposed voting χ^2 GOF method and the cepstrum method are parametric and require an optimization step to tune the parameters in order to minimize misdetection rates. This tuning can be done in a supervised way provided that enough data is available. Tuning can also be done in an unsupervised manner requiring a cost function such that the algorithm with the optimal model parameters has the lowest cost [13]. Either way, all model parameter configurations need to be checked (brute force approach) and the number of possibilities grows with the number of model parameters and the size of their ranges. A computationally efficient procedure will be introduced in Section 3.6 to optimize the model parameters in a reduced amount of time compared to the traditional brute force approach.

3.3 Denoising power signals using median filter

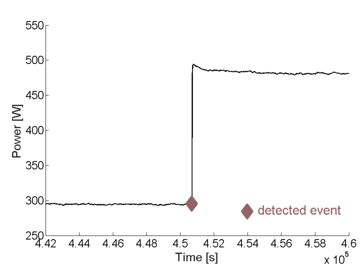
As input for the χ^2 GOF method, the power signal of a household is taken. This section explains the preprocessing needed before for this signal can be used by the event detection method. This preprocessing will also be done for the proposed voting χ^2 GOF and cepstrum method.

Definition of the power signal A power signal measures the amount of energy consumed per unit time. Thus if an appliance is turned on or off (i.e., an event occurs), the power signal will either increase or decrease. In the Americas and parts of Asia, the maximal power frequency is set to 60 Hz and in the rest of the world it is set to 50 Hz. Note that event detection can also be applied on other signals characterising the events defined by the turning on and off an appliance, e.g., the voltage/current measurements.

Denoising In [14] it is reported that noise or spikes in the power signal can trigger false detection of transitions, which can significantly hamper the performance



(a)



(b)

Figure 3.2: An example power trace [12] with noise is given in (a). The noise is falsely detected as an event. Figure (b) shows the power trace after applying the median filter with $m = 30$ *samples.*

of the event detection and thus successful load disambiguation of individual appliances. Therefore, it is important to remove the noise by preprocessing the data. In digital image processing, a similar preprocessing step is also needed and quite often this is done by the median filter, as it can remove impulsive noise while preserving sharpness of the edges [15]. Each sample in the power signal p_i is replaced by the median of its m neighbours:

$$p_i = median(p_{i-m/2} + p_{i-m/2+1} + ... + p_{i+m/2-1} + p_{i+m/2}) \tag{3.1}$$

As an example, Figure 3.2 shows that the standard χ^2 GOF event detection method identifies noise as an event if the signal is not filtered. It is found that the effectiveness of the median filter depends on the choice of its window size m. Therefore, this model parameter must be optimized (preferable in an efficient manner), as discussed further in Section 3.6.

3.4 Voting χ^2 GOF method

The standard χ^2 GOF method [1] detects events by relying on the fact that the distribution of power values before/after the occurrence of an event are different. To assess this difference, a probabilistic χ^2 test can be used. Assume two consecutive non-overlapping windows $q = (q_1, q_2, \cdots, q_n)$ and $p = (p_1, p_2, \cdots, p_n)$, each containing n data samples from the power signal. Then, an event occurs at the end of window q with a confidence level of $100(1-\alpha)\%$ and $n-1$ degrees of freedom, if

$$l_{GOF} = \sum_{i=1}^{n} \frac{(q_i - p_i)^2}{p_i} > \chi^2_{\alpha,n-1}. \tag{3.2}$$

The values of the $\chi^2_{\alpha,n-1}$ distribution can be looked up in a table[1] .

[1]http://sites.stat.psu.edu/~mga/401/tables/Chi-square-table.pdf

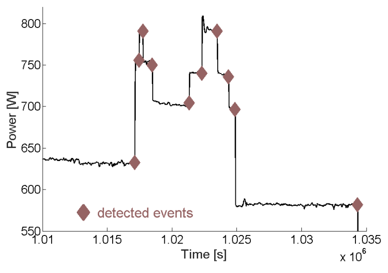
Base line robustness Although this χ^2 GOF method has been shown to be effective, it is not robust against base load changes, as illustrated in Figure 3.3. When the power base level is around 600W, and an appliance using 50W is switched on (as in Figure 3.3a), the event is correctly detected (see Figure 3.3c). However, the event would have been missed if a base load of 1500W was added (see Figure 3.3b and 3.3d). It is seen from equation (3.2) that events are characterized as a change in power $(q_i - p_i)$ relative to the power signal itself (p_i). Therefore, the method is prone to miss smaller events when the base load of the signal is high. This can lead to poor results, as shown in Figure 3.3b. As a solution, a voting mechanism (based on the idea presented in [2]) is proposed in this chapter to solve the problem.

Improved method In the voting χ^2 GOF method, the GOF is calculated for each sample in the power signal as given by equation (3.2). A voting window of length w slides over the resulting time-series of GOF-values and a sample gets a vote if its GOF-value is the highest among all points in the voting window. This results in a maximum of w votes. Each sample receiving at least v_{thr} votes is flagged as an event. As illustrated in the example in Figure 3.3e and 3.3f, the voting χ^2 GOF method is able to improve the detection ratios compared to the standard approach. The results section (Section 3.7) shows the robustness of the voting method against changes in base load in more detail.

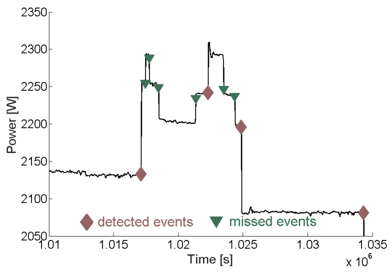
Parameter configurations Both the normal and voting χ^2 GOF method are sensitive to model parameter configurations, i.e., the confidence level α, the window size n, and extra for the voting method: the voting window size w and voting threshold v_{thr}. In [1] a suggestion is given for determining the window size n. However, small changes in the model parameter configurations can lead to missed events. Figure 3.4 shows an example for the normal χ^2 GOF method where three events are detected when the window size $n = 40$, but only two events are detected when the window size $n = 20$. It is thus beneficial to optimize the model parameters in an efficient way, which can be done using surrogate based optimization, see Section 3.6.

3.5 Cepstrum method

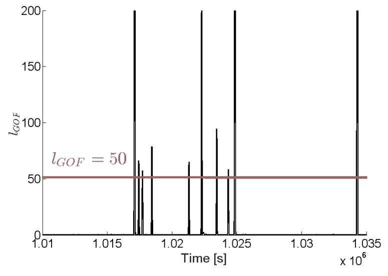
The previous sections investigated the power signal in the time domain. Alternatively, an analysis can be performed in the frequency domain using, e.g., cepstrum analysis. Cepstrum analysis was first introduced in 1963 to analyze the echoes within seismic signals produced by earthquakes [16]. Since then, it has proven to be a potent technique in several domains. One application is passive sonar, which involves listening to the environment without sending signals in order to



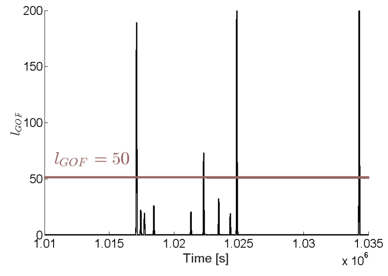
(a)



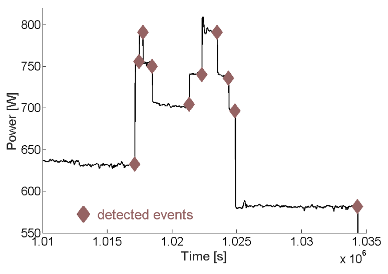
(b)



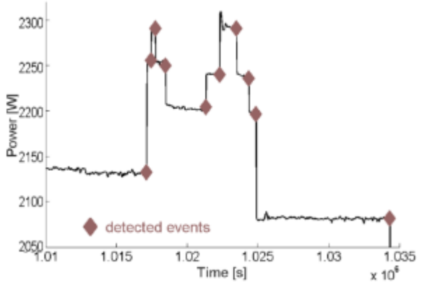
(c)



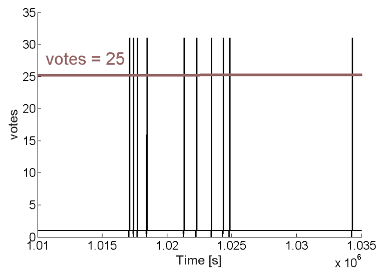
(d)



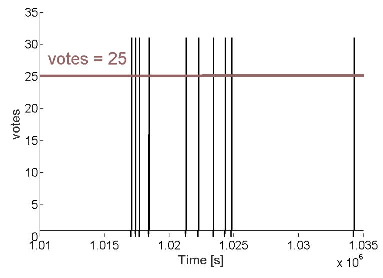
(e)



(f)



(g)



(h)

Figure 3.3: (a) A sample power trace from [12] and its detected events using the standard χ^2 GOF method with $n = 40$. (c) The corresponding l_{gof} values. (e) The detected events using the voting χ^2 GOF method with $w = 30$ and $v_{thr} = 25$. (g) The corresponding votes. Figures (b), (d), (f) and (h) show the same information for cases where a base load of $1500W$ is added.

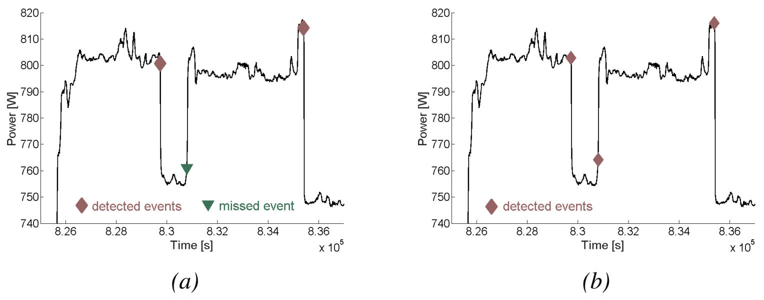


Figure 3.4: A sample power trace from the BLUED dataset [12], after applying a median filter with $m = 30$, with events detected by the standard χ^2 GOF method with $\alpha = 90\%$ for different window sizes: (a) $n = 20$, (b) $n = 40$. In the first case, only two events are detected, while three events are detected when $n = 40$.

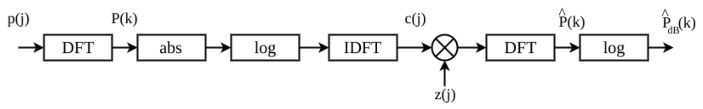


Figure 3.5: A schematic overview of the transformation from a time signal to spectral smoothed dB-scaled frequency components.

detect objects [17]. Another application is speech recognition [18], where cepstrum analysis has been successfully applied to increase the robustness of various algorithms. In the context of NILM, recent work has demonstrated the usefulness of cepstrum analysis for appliance recognition [19], especially when multiple devices are (de)activated simultaneously. Here, cepstrum analysis is used for event detection, rather than for appliance recognition.

Robust Cepstrum Method When using the cepstrum method, events are detected in the frequency domain where smoothing occurs in the quefrency domain, rather than the time domain. The different steps are outlined in Figure 3.5. Consider a window x of length n from a power signal p,

$$p = (p_i, p_{i+1}, ..., p_{i+n}) \tag{3.3}$$

where events need to be detected. First, this window will be converted from the time to the frequency domain, by using the Fourier transform:

$$P(k) = \sum_{j=1}^{n} p(j)\ e^{-2\pi ikj/n}\ ,\ 0 \leq k < n. \tag{3.4}$$

Then, the inverse Fourier transform is applied to the logarithm of $|P|$, leading

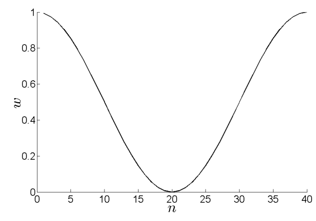


Figure 3.6: The response of the filter z.

to the cepstrum components in the so-called quefrency domain:

$$c(j) = \frac{1}{n} \sum_{k=0}^{n-1} log_{10}(\, |P(k)| \,) e^{2\pi ijk/N} \; , \; 0 \leq j < n. \tag{3.5}$$

These cepstrum components are smoothed by means of a filter z, after which they are transformed back to frequency components by applying the Fourier transformation:

$$\hat{P}(k) = \sum_{j=1}^{n} z(j) \; c(j) \; e^{-2\pi ikj/n} \; , \; 0 \leq k < n. \tag{3.6}$$

The filter z is defined as one minus the Hann window, with a response as visualised in Figure 3.6:

$$z(j) = 1 - 0.5 \; (1 - cos(2\pi j/n)) \; , \; 0 \leq j \leq n. \tag{3.7}$$

Because the relative difference in values of the components is more informative than the absolute difference, the frequency components are converted to a decibel (dB) scale:

$$\hat{P}_{dB}(k) = 20 \; log_{10}(\hat{P}(k)). \tag{3.8}$$

These components are an informative indicator for the absence or presence of events in the time window. This is illustrated in Figure 3.7: if an event is present, all the cepstrum smoothed dB scaled frequency components have higher values (see Figure 3.7b) than when no event is present (see Figure 3.7d). Finally, it is checked whether all frequency components exceed a chosen threshold τ, and declare an event if the following condition holds:

$$\min_{0 \leq k < N} (\hat{P}_{dB}(k)) > \tau \tag{3.9}$$

Note that the threshold τ needs to be optimized in order to achieve high event detection ratios. The efficient optimization of this parameter τ (and others indicated previously) is discussed next, in Section 3.6. The results section (Section 3.7) shows that this method, just like the voting χ^2 GOF method, is robust against changes in the base load.

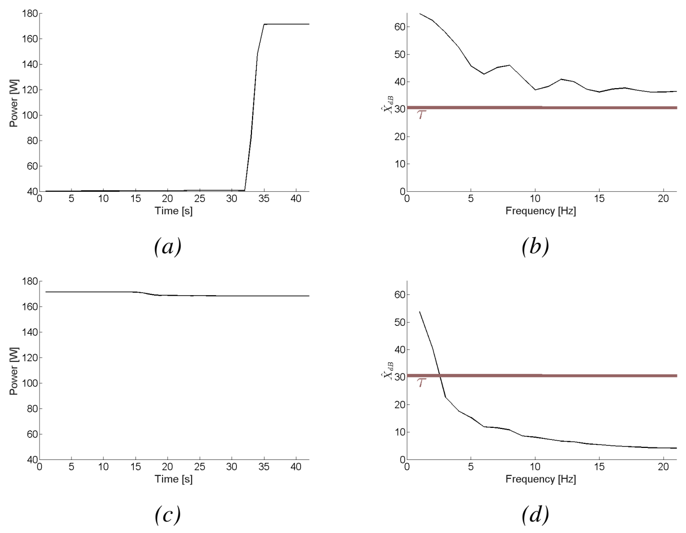


Figure 3.7: An example of a window with size $n = 40$ of a power trace and the corresponding smoothed frequency components $\hat{P}$ of an event, respectively (a) and (b), and a non-event, respectively (c) and (d).

3.6 Efficient surrogate-based model parameter optimization

All the methods described in the previous sections have parameters that need to be optimized in order to achieve high event detection ratios. The total number of model parameter configurations that must be evaluated is very high, see Section 3.7 for the specific numbers. Rather than reducing the granularity of the model parameter ranges, surrogate-based optimization is proposed, which can significantly speed up the process. It is chosen to adopt Surrogate-Based Optimization (SBO) [3], which assumes that smooth changes in the model parameter configurations will lead to smooth changes in detection ratios. Under that condition, an exhaustive search of the overall model parameter space is not required to find the optimum solution. Rather than computing the results for all possible model parameters, a surrogate model of the optimization objective function (the utility function) is generated that is locally accurate in the regions of interest (the optimum). SBO makes use of one of the most popular sequential sampling concepts, namely, the Expected Improvement (EI) measure for optimization with locally accurate surrogate models [3]. The EI measure aims to maximize the utility function by guiding the sequential selection of appropriate model parameter configurations into the direction where the optimum solution is most likely to be found, using Bayesian methods. Once the algorithm discovers a configuration that is sufficiently close to

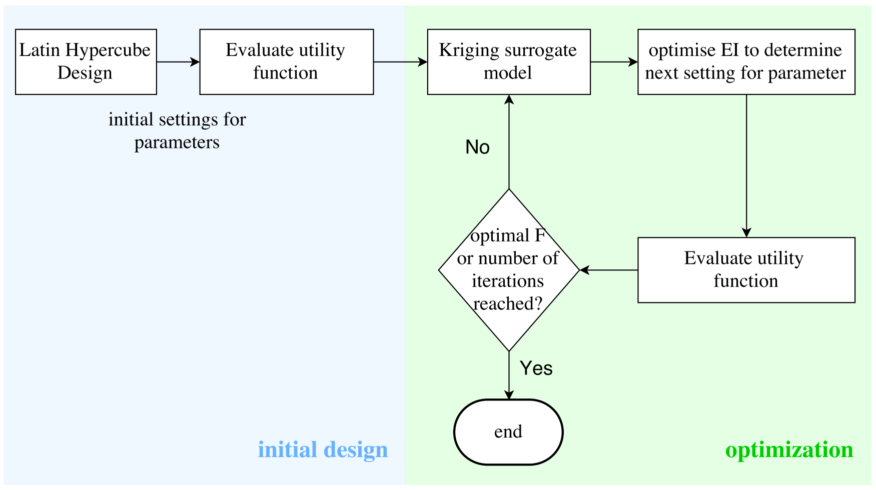

Latin Hypercube Design
Evaluate utility function
initial settings for parameters
Kriging surrogate model
optimise EI to determine next setting for parameter
No
optimal F or number of iterations reached?
Evaluate utility function
Yes
end
initial design
optimization

Figure 3.8: A flow chart explaining SBO.

the optimum, the optimization terminates and the final solution is returned. As a result, only part of all model parameter configurations need to be tested to obtain the optimal solution. SBO has already been applied successfully in other research areas, like e.g., wireless communication [20], electromagnetics [3], and microwave filter design [21].

The different steps of the algorithm are summarized in Figure 3.8. SBO requires a unified utility function that needs to be maximized (i.e., the F-measure as explained further). First, a limited set of calculations are performed on this utility function such that the model parameter space is well sampled to create the initial design. Then, a Kriging surrogate model is built that is sequentially updated with additional configurations as suggested by the EI infill criterion. The EI infill criterion effectively balances between enhancing the global accuracy of the surrogate model (exploration) and improving its accuracy near the optimal solution found so far (exploitation). As the algorithm proceeds, the search is guided towards the optimal solution while limiting the number of possible configurations for the model parameters. As soon as a satisfactory result is found, the optimization is terminated and the best solution is returned.

Definition of the utility function To quantify the performance of event detection methods, the harmonic mean of precision and recall (also known as the F-measure) is used as suggested in [22]. If the considered model parameters are **g**, it is defined as:

$$F(\mathbf{g}) = 2 \cdot \frac{\mathit{precision}(\mathbf{g}) \cdot \mathit{recall}(\mathbf{g})}{\mathit{precision}(\mathbf{g}) + \mathit{recall}(\mathbf{g})} \tag{3.10}$$

$$\mathit{precision}(\mathbf{g}) = \frac{\mathit{TP}(\mathbf{g})}{\mathit{TP}(\mathbf{g}) + \mathit{FP}(\mathbf{g})} \tag{3.11}$$

$$\mathit{recall}(\mathbf{g}) = \frac{\mathit{TP}(\mathbf{g})}{TP(\mathbf{g}) + \mathit{FN}(\mathbf{g})} \tag{3.12}$$

where *precision* is the fraction of detected events that are true and *recall* is the fraction of true events that are detected, *TP* are the true-positives (correctly predicted events), *FP* are the false-positives (incorrectly predicted events), *FN* are the false-negatives (undetected events). The goal of the optimization procedure is to choose the model parameters **g** in such a way that the utility function is maximized.

Evaluation of the initial configurations for model parameters First, a limited number of configurations K for model parameter **g** are evaluated using Equation (3.11) and (3.12) to determine corresponding values of precision and recall. To this end, an optimized Latin Hypercube Design (LHD) was used because of its space-filling properties [23]. Next, the corresponding F measures are calculated. This leads to the configurations:

$$S = \{(\mathbf{g}_k, F(\mathbf{g}_k)), k = 1 \dots K\}. \tag{3.13}$$

Generation of a Kriging surrogate model With the calculated F-measures, a Kriging model is built. Kriging models are part of a broader class of approximation methods, called Gaussian Processes (GP), and have a particular importance in SBO. While traditional approximation methods predict only a single function value, GP methods can predict the uncertainty of a function value as the realization of a normally distributed random variable $Y(\mathbf{g}) \sim N(\mu(\mathbf{g}), \sigma(\mathbf{g}))$ where $\mu(\mathbf{g})$ denotes the predicted value ($\mu(\mathbf{g}) = F(\mathbf{g})$) and $\sigma(\mathbf{g})$ denotes the prediction variance. This property is exploited by the EI infill criterion to guide the sequential sampling, as shown in the next section. More details about Kriging can be found in literature, e.g., [24].

Expected Improvement infill criteria Once a Kriging model is built, the EI measure determines the optimum location of the next infill point **g** that contains the parameter configurations used to configure the event detection method. First, the EI quantifies the Probability of Improvement (PoI), the amount of improvement that is expected to occur when a certain configuration is explored as compared to the optimal value found so far. The EI is calculated by considering every possible improvement over the current best value F_{max}, multiplied with the associated

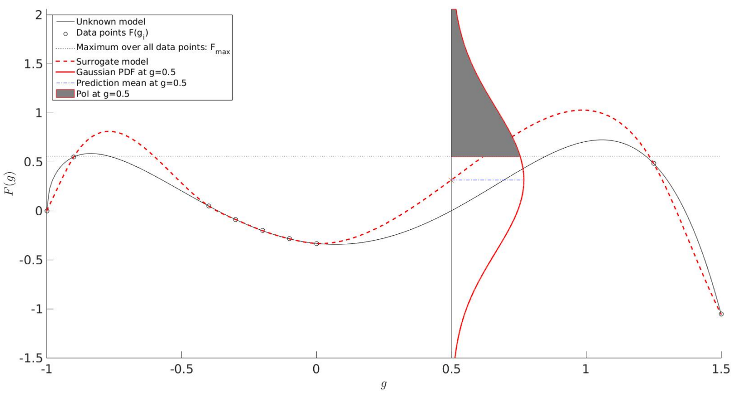


Figure 3.9: A graphical illustration of expected improvement: a surrogate model (dashed line) is constructed based on some data points (circles) of an unknown function $F(\mathbf{g})$. For each point the surrogate model predicts a Gaussian probability density function (PDF). An example of such a PDF is drawn at $g = 0.5$.

likelihood. If $\phi(.)$ denotes the probability density function of a random variable, then the EI can be written in integral form as follows [25]:

$$E(I(\mathbf{g})) = \int_{F_{max}}^{\infty} I(\mathbf{g}) \cdot \phi(Y(\mathbf{g}))\, dY, \tag{3.14}$$

where the improvement $I(\mathbf{g})$ of $Y(\mathbf{g})$ over F_{max} is defined as

$$I(\mathbf{g}) = max(Y(\mathbf{g}) - F_{max}, 0). \tag{3.15}$$

A graphical illustration of the EI concept is given in Figure 3.9 where one model parameter is optimized. Note that the EI function (3.14) corresponds to the first moment of the shaded area in Figure 3.9. Once a configuration of **g** is found for which the $E(I(\mathbf{g}))$ is maximal, its corresponding F-measure is calculated and added as a new data sample to the set S. Based on the additional information, the Kriging model is rebuilt and the process is repeated until a satisfactory solution is found, i.e., until the maximum is reached (in our case $F = 1$), or when the maximum number of iterations is exceeded.

An implementation of the SBO routine is available in the Surrogate Modelling (SUMO) Toolbox [26, 27] (available online on `http://sumo.intec.ugent.be`).

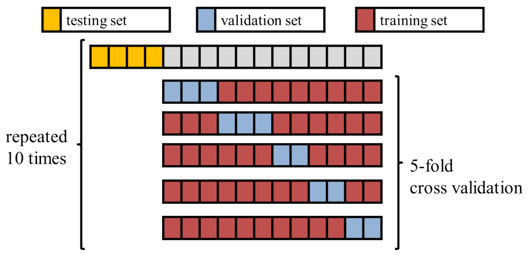


Figure 3.10: A schematic overview of the optimization procedure.

3.7 Results and discussion

Dataset In this section, the robustness of the proposed methods against different baseload levels is tested on the BLUED dataset [12]. The aggregated power signal sampled at 60Hz from a family residence in the United States for a whole week is considered. Every state transition of each appliance is manually labeled, providing the ground truth. The considered house has a two-phase power consumption, where 904 transitions are recorded in phase A and 1578 in phase B. Each phase has its own properties, e.g., phase B is more noisy than phase A. For that reason, phase A and B are optimized and tested separately. For each method, the data is passed through a median filter, as explained in Section 3.3.

Cross validation Performance is evaluated on 20% of the data, whereas the remaining 80% is used for training. Performance is reported averaged over 10 runs (each with a random 20% test split). For training, 5-fold cross validation is used on the other 80% to set the optimal parameter values. The overall set-up is summarized in Figure 3.10. Note that for the division, the trace of an entire day is taken as a whole unit.

Trained model parameters The model parameters and the ranges under consideration are listed in Tables 3.1, 3.2, and 3.3 for the standard χ^2 GOF method, the voting χ^2 GOF method and the cepstrum method respectively. As the power frequency is 60 Hz, this means there is a sample every 0.02 seconds. Consequently, when the window size n varies from 1 to 100, it covers a time window from 0.02 seconds to 2 seconds. The total number of model parameter configurations to be evaluated is very high: the choices listed in Table 3.1-3.3 amount to 50000 $(= 100 * 100 * 5)$, 10000000 $(= 100 * 100 * 100 * 10)$, and 500000 $(= 100 * 100 * 50)$ for the standard χ^2 GOF method, the voting χ^2 GOF method, and the cepstrum method respectively. Since the approximate running time of the algorithms is 214, 1.635, and 529 seconds, this results in an overall optimization time of 124, 189.236, and 3.061 days respectively on a modern dual core machine.

standard χ^2 GOF				
name		range	optimization time / configurations	
window median filter	m	$[1, 100]$	brute force	124 days / 50000
window event detection	n	$[1, 100]$	SBO	5.9 hours / 100
confidence level	α	$\{90, 95, 97.5, 99, 99.9\}$		

Table 3.1: On the left are the present model parameters **g** *and range for the standard χ^2 GOF event detection. On the right are the optimization time for the model parameters and the number of parameter configurations needed to be checked.*

voting χ^2 GOF				
name		range	optimization time / configurations	
window median filter	m	$[1, 100]$	brute force	189.2 days / 10^7
window event detection	n	$[1, 100]$	SBO	45 hours / 100
window voting system	w	$[1, 100]$		
voting threshold	v_{thr}	$w * [0.1, 1]$		

Table 3.2: On the left are the present model parameters **g** *and range for the voting χ^2 GOF event detection. On the right are the optimization time for the model parameters and the number of parameter configurations needed to be checked.*

This makes a brute-force optimization of the model parameters practically infeasible. Therefore, training is done using SBO, as explained in Section 3.6. To begin with, $K = 10$ configurations for the model parameters are evaluated and used to build the Kriging model. Based on this model, one new configuration for the parameters is proposed and evaluated with the F-measure. The result is used to update the Kriging model. This is done 90 times, resulting in 100 evaluated configurations. Adding more iterations did not prove to be useful in practice, as the F-measure did not significantly improve and the suggested parameter configurations were all in the same neighbourhood. A good (but possibly local) optimum is thus found.

Test use cases To find out if each method is robust against changes in the base load of the power signal, the three methods are applied in three different use cases. In all cases, the data is preprocessed with a median filter, and a base load of re-

cepstrum				
name		range	optimization time / configurations	
window median filter	m	$[1, 100]$	brute force	3.1 days / 500000
window event detection	n	$[1, 100]$	SBO	14 hours / 100
threshold	τ	$[1, 50]$		

*Table 3.3: On the left are the present model parameters **g** with their respective abbreviation and range for the cepstrum method. On the right are the optimization time for the model parameters and the number of parameter configurations needed to be checked.*

spectively 0W, 1500W, and 3000W is added to the power signal. In practice, such high base load conditions arise when multiple high-power devices are operating in the background, such as electrical heaters (that can easily consume 1500W). Considering the results of these use cases, a conclusion can be made about each method's robustness.

Robustness of standard χ^2 GOF method The results of the standard χ^2 GOF method when applied on the first three use cases are given in Figure 3.11, showing the spread of the F-measure caused by running the 5-fold cross validation ten times. When no offset is added, the performance for phase A is almost perfect ($F \approx 0.98$) and for phase B the performance is $F \approx 0.80$. However, this changes when the base load is increased by adding an offset to the signal. The F-measure keeps dropping as the offset increases, indicating that the method is not robust against higher base loads.

Robustness of voting χ^2 GOF method The results of the voting χ^2 GOF method when applied on the first three use cases are given in Figure 3.11. As can be seen, the voting χ^2 GOF method gives comparable results to the standard χ^2 GOF method for phase A and B when no offset is added. When offsets are added to the signal, the F-measure remains the same, indicating the robustness of the voting χ^2 GOF method. This in contrast to the standard χ^2 GOF method.

Robustness of cepstrum method The results of the cepstrum method when applied on the first three use cases can be found in Figure 3.11. The F-measure for phase A when no offset is added to the base load, is the same as the previous methods ($F \approx 0.98$). For phase B, it is $F \approx 0.81$, which is a bit higher than the previous methods. When an offset is added to the base load, the F-measure remains the same ($F \approx 0.81$), just like the voting χ^2 GOF method, indicating the robustness of the cepstrum method.

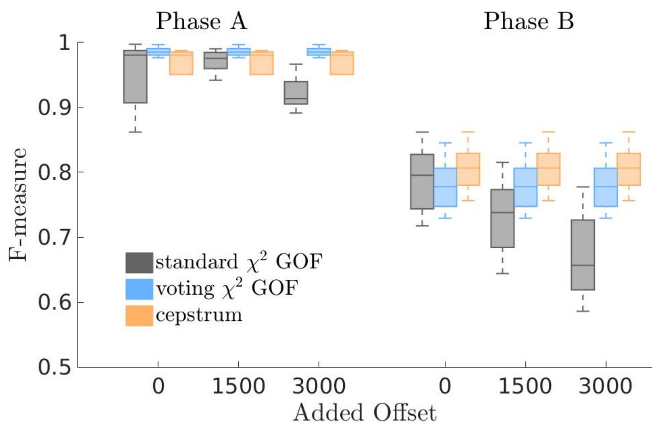


Figure 3.11: The F-measure when detecting events with the standard χ^2 GOF, voting χ^2 GOF, and cepstrum method.

Timing improvement due to SBO Comparing the running time of SBO and the brute-force approach, it is found that for the standard χ^2 GOF method it is reduced from approximately 124 days to 5.9 hours, for the voting χ^2 GOF method from 189.2 days to 45 hours, and for the cepstrum method from 3.1 days to 14 hours, resulting in a speed up factor of approximately 500, 100000, and 5000 respectively. This is caused by the fact that the number of evaluated parameter configurations is reduced from 5000, 10000000, and 500000 (for the standard χ^2 GOF method, the voting χ^2 GOF method, and the cepstrum method respectively) to 100 for all methods while maintaining a good F-measure. This timing analysis assumes that the method to which we compare (here, the brute-force approach) includes no background information. Other smarter methods, like random sampling, will lead to a smaller timing improvement due to SBO.

3.8 Conclusion

Two event detection methods have been proposed, namely (1) the voting χ^2 GOF method, and (2) the cepstrum method. Each method is robust against base load differences compared to the standard χ^2 GOF method. For example, when a base load of 3000W (which corresponds to the power consumption of two typical electrical heaters) is added to the power signal, the voting χ^2 GOF method leads to a performance increase of $7-12\%$ in terms of F-measure, while Cepstrum reaches $7-15\%$ larger F-measure values, in comparison to the standard χ^2 GOF method.

In order to obtain optimal parameter configurations for these methods, a workflow using surrogate-based optimization is proposed. Timing results confirm that the parameter optimization process can be sped up: in our experiments there is a speed up with a factor up to 100000 between the standard brute force and the surrogate-based optimization.

References

[1] Yuanwei Jin, Eniye Tebekaemi, Mario Berges, and Lucio Soibelman. *A time-frequency approach for event detection in non-intrusive load monitoring*. In Proceedings of SPIE, volume 8050, pages 80501U–80501U, 2011.

[2] Kyle D Anderson, Mario E Bergés, Adrian Ocneanu, Diego Benitez, and José MF Moura. *Event detection for non intrusive load monitoring*. In IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society, pages 3312–3317. IEEE, 2012.

[3] Ivo Couckuyt, Frederick Declercq, Tom Dhaene, Hendrik Rogier, and Luc Knockaert. *Surrogate-based infill optimization applied to electromagnetic problems*. International Journal of RF and Microwave Computer-Aided Engineering, 20(5):492–501, 2010.

[4] George William Hart. *Nonintrusive appliance load monitoring*. Proceedings of the IEEE, 80(12):1870–1891, 1992.

[5] Mario Berges, Ethan Goldman, H Scott Matthews, and Lucio Soibelman. *Learning systems for electric consumption of buildings*. In Computing in Civil Engineering (2009), pages 1–10. 2009.

[6] Chuan Choong Yang, Chit Siang Soh, and Vooi Voon Yap. *Comparative study of event detection methods for non-intrusive appliance load monitoring*. Energy Procedia, 61:1840–1843, 2014.

[7] Chuan Choong Yang, Chit Siang Soh, and Vooi Voon Yap. *A systematic approach to ON-OFF event detection and clustering analysis of non-intrusive appliance load monitoring*. Frontiers in Energy, 9(2):231–237, 2015.

[8] Yung Fei Wong, Y Ahmet Şekercioğlu, Tom Drummond, and Voon Siong Wong. *Recent approaches to non-intrusive load monitoring techniques in residential settings*. In Computational Intelligence Applications In Smart Grid (CIASG), 2013 IEEE Symposium on, pages 73–79. IEEE, 2013.

[9] The Minh Luong, Vittorio Perduca, and Gregory Nuel. *Hidden Markov Model Applications in Change-Point Analysis*. arXiv preprint arXiv:1212.1778, 2012.

[10] Guillermo L Grinblat, Lucas C Uzal, and Pablo M Granitto. *Abrupt change detection with one-class time-adaptive support vector machines*. Expert Systems with Applications, 40(18):7242–7249, 2013.

[11] William Gu, Jaesik Choi, Ming Gu, Horst Simon, and Kesheng Wu. *Fast Change Point Detection for electricity market analysis*. In Big Data, 2013 IEEE International Conference on, pages 50–57. IEEE, 2013.

[12] Kyle Anderson, Adrian Filip, Diego Benitez, Derrick Carlson, Anthony Rowe, and Mario Berges. *BLUED: A fully labeled public dataset for event-based non-intrusive load monitoring research*. In 2nd Workshop on Data Mining Applications in Sustainability (SustKDD), page 2012, 2011.

[13] Michele Volpi, Devis Tuia, Gustavo Camps-Valls, and Mikhail Kanevski. *Unsupervised change detection with kernels*. IEEE Geoscience and Remote Sensing Letters, 9(6):1026–1030, 2012.

[14] Tianji Wu and Mani Srivastava. *Low-cost appliance state sensing for energy disaggregation*. In Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, pages 53–55. ACM, 2012.

[15] Ioannis Pitas and Anastasios N Venetsanopoulos. *Nonlinear digital filters: principles and applications*, volume 84. Springer Science & Business Media, 2013.

[16] Bruce P Bogert. *The quefrequency analysis of time series for echoes: Cepstrum pseudo-autocovariance, cross-cepstrum, and saphe cracking*. Time series analysis, pages 209–243, 1963.

[17] PV Kiran, Amit Kumar Verma, and G Appala Naidu. *Application of cepstrum in passive sonar*. International Journal of Engineering Research and Applications (IJERA), pages 1919–1924, 2012.

[18] Hans-Günter Hirsch and David Pearce. *The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions*. In ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW), 2000.

[19] Seongbae Kong, Youngwook Kim, Rakkyung Ko, and Sung-Kwan Joo. *Home appliance load disaggregation using cepstrum-smoothing-based method*. IEEE Transactions on Consumer Electronics, 61(1):24–30, 2015.

[20] Michael T Mehari, Eli De Poorter, Ivo Couckuyt, Dirk Deschrijver, Jono Vanhie-Van Gerwen, Daan Pareit, Tom Dhaene, and Ingrid Moerman. *Efficient global optimization of multi-parameter network problems on wireless testbeds*. Ad Hoc Networks, 29:15–31, 2015.

[21] Prashant Singh, Dirk Deschrijver, Davy Pissoort, and Tom Dhaene. *Accurate hotspot localization by sampling the near-field pattern of electronic devices*. IEEE Transactions on Electromagnetic Compatibility, 55(6):1365–1368, 2013.

[22] Stephen Makonin and Fred Popowich. *Nonintrusive load monitoring (NILM) performance evaluation*. Energy Efficiency, 8(4):809–814, 2015.

[23] V Roshan Joseph and Ying Hung. *Orthogonal-maximin Latin hypercube designs*. Statistica Sinica, pages 171–186, 2008.

[24] Ivo Couckuyt, Slawomir Koziel, and Tom Dhaene. *Kriging, co-kriging and space mapping for microwave circuit modeling*. In Microwave Conference (EuMC), 2011 41st European, pages 444–447. IEEE, 2011.

[25] Donald R Jones. *A taxonomy of global optimization methods based on response surfaces*. Journal of global optimization, 21(4):345–383, 2001.

[26] Dirk Gorissen, Ivo Couckuyt, Piet Demeester, Tom Dhaene, and Karel Crombecq. *A surrogate modeling and adaptive sampling toolbox for computer based design*. Journal of Machine Learning Research, 11(Jul):2051–2055, 2010.

[27] Joachim van der Herten, Ivo Couckuyt, Dirk Deschrijver, and Tom Dhaene. *Adaptive classification under computational budget constraints using sequential data gathering*. Advances in Engineering Software, 99:137–146, 2016.

4 Appliance Identification

"You know my methods. Apply them."
– Sherlock Holmes, *The Sign of Four*

De Baets, L., Develder, C., Deschrijver, D., & Dhaene, T. (2017). "Automated classification of appliances using elliptical fourier descriptors". In the proceedings of the IEEE International Conference on Smart Grid Communications, Dresden (Germany), pp. 1–6.
The published method and results are presented in Sections 4.2, and 4.3. Subsection 4.3.5 discusses novel results.

De Baets, L., Ruyssinck, J., Develder, C., Dhaene, T., & Deschrijver, D. (2018). "Appliance classification using VI trajectories and convolutional neural networks". Energy and Buildings, 158, 32-36.
The published method and results are presented in Sections 4.2, and 4.4. Subsection 4.4.4 discusses novel results.

De Baets, L., Jingkun, G., Develder, C., Dhaene, T., Berges, M., & Deschrijver, D. (2017). "Handling imbalance in an extended PLAID". In the proceedings of the 5th IFIP Conference on Sustainable Internet and ICT for Sustainability, Funchal (Portugal), pp. 1–5.
The published method and results are presented in Sections 4.6.

Section 4.5 presents a novel method and the corresponding results.

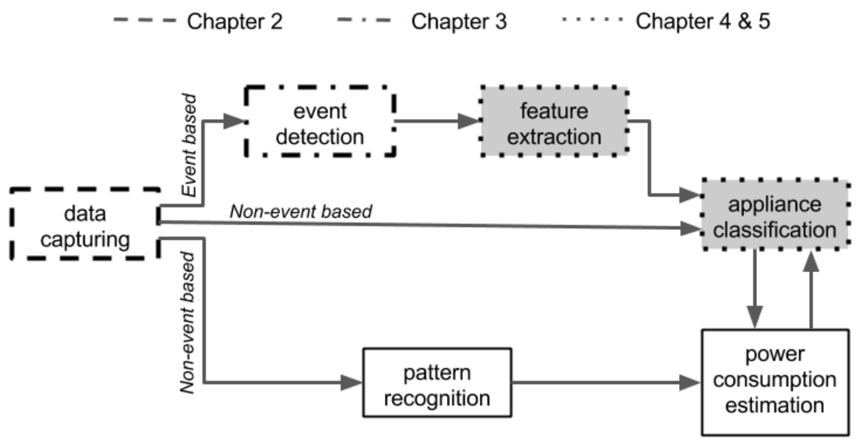


Figure 4.1: A schematic overview of the steps in NILM, situating this chapter.

4.1 Introduction

In the previous chapter, we detected events using a robust method. In the next step, we extract features from steady state behaviour after the event and use them to classify the appliances responsible for the events, see Figure 4.1.

The type of extracted features heavily depends on the sampling rate of the measurements. When using low frequency data ($\leqslant$ 1 Hz), the most common features are the power levels and the on/off durations [1]. A drawback of this approach is that only energy-intensive appliances can be detected. This can be alleviated by performing fine-grained measurements at the cost of an increased data storage rate and more complex data analytics, i.e., the voltage and current signals are measured. From these signals, features like the harmonics [2] and the frequency components [3] from the steady-state and transient behavior can be calculated. More recently, the possibility to consider voltage-current (VI) trajectories has also been considered [4–6]. Once the features are extracted, they can be fed into different classification methods, like support vector machines (SVM) [7], decision trees [8], or nearest neighbors [9]. In order to distinguish appliances based on their VI trajectories, powerful measuring devices must be used that are able to sample high frequency data.

In this chapter, the problem of classifying appliances based on their VI trajectories is addressed as an image recognition problem. Section 4.2 explains the two possibilities to represent the trajectory as an image resulting in an pixelated and weighted pixelated VI image. Section 4.3 describes a classical method for image recognition that: (1) finds the contours, (2) calculates the elliptic Fourier descriptors of the contours, and (3) trains machine learning methods using these elliptical Fourier descriptors. Section 4.4 solves the image recognition problem using convolutional neural networks (CNNs). Such networks are often used for classification tasks in computer vision, due to their excellent discriminative power in classifying images [10]. It is shown that a CNN approach can also be valuable in a NILM context to discriminate active appliances based on the weighted pixelated VI image. Section 4.5 explains how the current consumption can be added as an extra feature, or used as only feature.

Ideally, to test the proposed methods, a dataset having high frequency aggregated and high frequency sub-metered v and i signals should be used. However, when these methods were developed, no public dataset in existence included both. For this reason, both the 2014 version of PLAID [11] and WHITED [12] are considered as datasets to benchmark the methods as they both contain high frequency sub-metered data. This research on appliance classification was a first step towards a more realistic NILM setting starting from the aggregated power measurements. It was a very meaningful step, as typically appliances are turned on/off one at a time, and the single appliance current (and thus VI trajectory) can be extracted from the aggregated measurements by considering the difference in current before and after the event. Later, we extended PLAID with high frequency aggregated data and the results on this data confirms the practical feasibility of extracting measurements. In addition, also other recent work [13] has validated this idea.

An additional problem for appliance classification is the imbalance present in the available datasets: some appliance types are represented by more measurements (the majority classes) than others (the minority classes), as can be seen in Table 4.1. This can influence the performance achieved by a particular classifier trained using this data. Section 4.6 investigates and handles this imbalance in PLAID.

4.2 VI image

The VI trajectory of an appliance is obtained by plotting the voltage against the current for a defined time period when the appliance is turned on and in steady state. It is shown in [4] that manually extracted features from the VI trajectory such as the enclosed area, slope of the middle segment, etc., can be used to classify appliances. However, extracting features from the VI trajectory proved not to be straightforward.

As an alternative, the VI trajectory can be converted into a pixelated VI im-

appliance type	PLAID	WHITED	COOLL
AC	66	10	
CFL	175	20	
Fan	115	60	40
Hairdryer	156	60	80
ILB	114	60	80
Vacuum	38	40	140
Washing Machine	26	10	
Router			20

Table 4.1: The number of instances for some selected appliances in the 2017 version of PLAID, WHITED, and COOLL.

age ($n \times n$ matrix) by meshing the VI trajectory. In [5, 6], each cell of the mesh is assigned a binary value that denotes whether or not it is traversed by the trajectory, see Figure 4.2d. Based on this pixelated VI image, several features can be extracted and used to classify different power loads [5]. Examples of features are the number of continuums of occupied cells, and the binary value of the left horizontal and central cell. In [6], the pixelated VI image is re-arranged into an input vector that can be fed directly into a classifier, such as logistic regression or random forests, to classify different appliances. Section 4.3 builds further upon these methods by using the pixelated VI image to perform object recognition.

One limit of these features is the compression of the information contained in the VI trajectory into a limited number of correlated summary statistics. As an alternative, this chapter also proposes to represent the VI trajectory as a weighted pixelated image. The necessary processing steps are:

1. Taking the voltage v and the current i when the appliance is active over a certain number of times (the steady-state behaviour),
2. Normalizing such that $[v, i] \in [-1, 1]^2$,
3. Creating the continuous VI trajectory,
4. Overlaying it with a $n \times n$ mesh,
5. Counting for each cell in the mesh the number of trajectory points it contains,
6. Normalizing the values of the cells such that the maximum value of the cells is one.

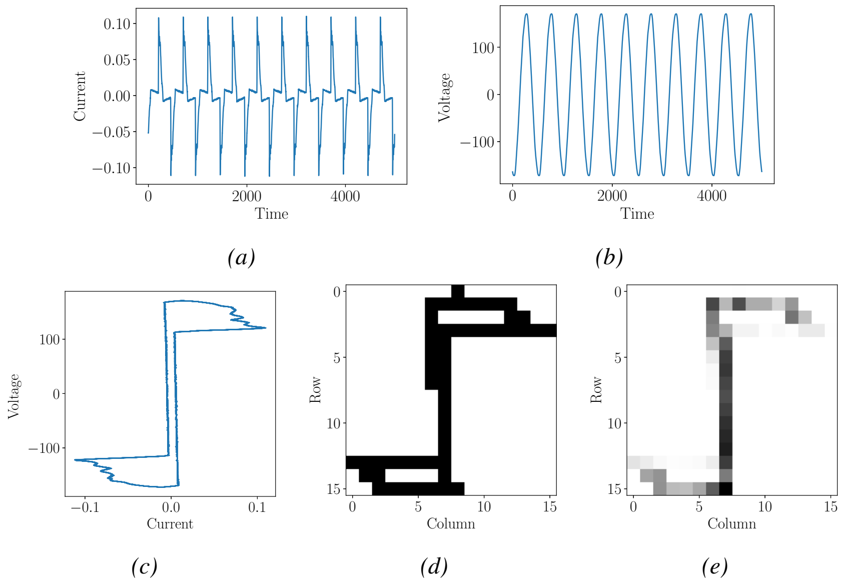


Figure 4.2: The transformation from the continuous VI trajectory of a CFL (c) into the pixelated (d) and weighted pixelated (e) VI image (right) for $n = 15$.

Figure 4.2 illustrates the transformation from the continuous VI trajectory into the pixelated and weighted pixelated VI image. Note that instead of creating an image of the VI trajectory, it is also possible to create an image of a current cycle.

4.3 Object recognition

Object recognition is a fundamental problem in computer vision and the underlying concepts have been applied in multiple fields, such as distinguishing biological species [14], optical character recognition [15], and face recognition [16]. In object recognition, the contours of an object are identified from the image, characterized by elliptic Fourier descriptors and then classified with a label. In this section, object recognition is used to recognize the contours of a VI trajectory in the pixelated image, see Subsection 4.3.1, and to describe them using elliptical Fourier descriptors, see Subsection 4.3.2. Machine learning methods use these descriptors to classify the objects, see Subsection 4.3.3. Subsection 4.3.4 benchmarks the method on the 2014 version of PLAID consisting of high frequency submetered data and Subsection 4.3.5 benchmarks it on the aggregated data of the 2018 version of PLAID. The different steps are visualized in Figure 4.3.

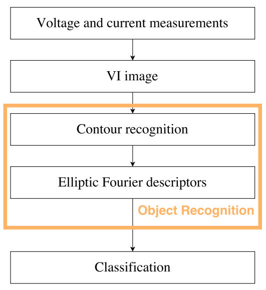

Voltage and current measurements
VI image
Contour recognition
Elliptic Fourier descriptors
Object Recognition
Classification

Figure 4.3: The classification workflow.

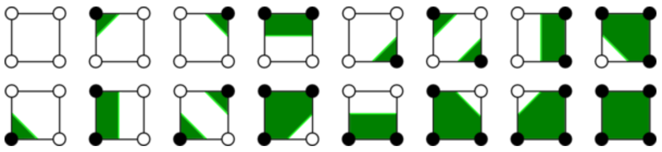


Figure 4.4: The lookup table consisting of sixteen different cases in which parts of the contour are formed.

4.3.1 Contours

The contour of an object in an image is a closed curve that forms the boundary of that object. The marching squares algorithm [17] can be used to find all contours in an image. The basic idea is that every pixel of the image is examined and matched with one of the sixteen cases shown in Figure 4.4. Each case creates at most two edges. Similar results can be obtained using other contour algorithms [18].

Figure 4.5 shows the detected contours of the VI trajectory of a compact fluorescent lamp. This example has three contours. To avoid that the trajectory touches the border, extra pixel rows and columns are added to the sides. Otherwise this would result in two separate outside contours instead of one. Only one contour can be used to classify the appliances because not all appliances have the same number of contours, while this is required for the use of machine learning methods. For that reason, the outer counter is chosen as this contour is a closed curve that resembles the shape of a smoothed VI trajectory. In contrast to the original VI trajectory, all points on the contour are separated uniformly, such that the Euclidean distance between neighbouring points on the contour is the same.

4.3.2 Elliptic Fourier descriptors

Once the contour is identified, elliptical Fourier descriptors (EFD) are used to characterize the corresponding appliance. A brief outline on how to calculate these

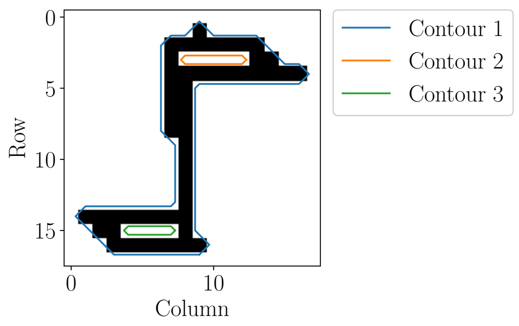


Figure 4.5: The identification of the VI trajectory contours of a CFL. Only the outer contour is used for object classification.

EFDs is given in this section and the reader is referred to [19, 20] for mathematical details.

EFDs define the contour as the sum of a certain number of ellipses (e) required to mimic the shape. Each ellipse is formed by two sets of partial differential equations, each having sine and cosine terms. One set of equations is defined along the x-axis and the other one along the y-axis:

$$x(t) = A_0 + \sum_{i=1}^{e-1} A_i \text{cos}(it) + B_0 + \sum_{i=1}^{e-1} B_i \text{sin}(it) \tag{4.1}$$

$$y(t) = C_0 + \sum_{i=1}^{e-1} C_i \text{cos}(it) + D_0 + \sum_{i=1}^{e-1} D_i \text{sin}(it) \tag{4.2}$$

where A_0, B_0, C_0, and D_0 are the constants, A_i, B_i, C_i, and D_i are the harmonic coefficients of the i^{th} order, and e represents the points sampled from the time axis given by the period 2π. B_0 and D_0 are zero and can be omitted from the formula. This is a transformation from the spatial to the frequency domain.

Each harmonic is thus described by four Fourier coefficients, two each for the x- and y-axis, generating a total of $4 \cdot e$ coefficients. The first harmonic describes the overall shape, location, size, and rotational orientation of the contour. Additionally, consecutive harmonics can be included to capture more detailed information about the contour's complexity. Figure 4.6 shows the reconstruction of the contour when using up to $e = 10$ harmonics. The approximated contour better resembles the original contour when more harmonics are included. Using 3 or more EFDs, it is possible to recreate the contour without requiring prior expertise on the important features to represent it.

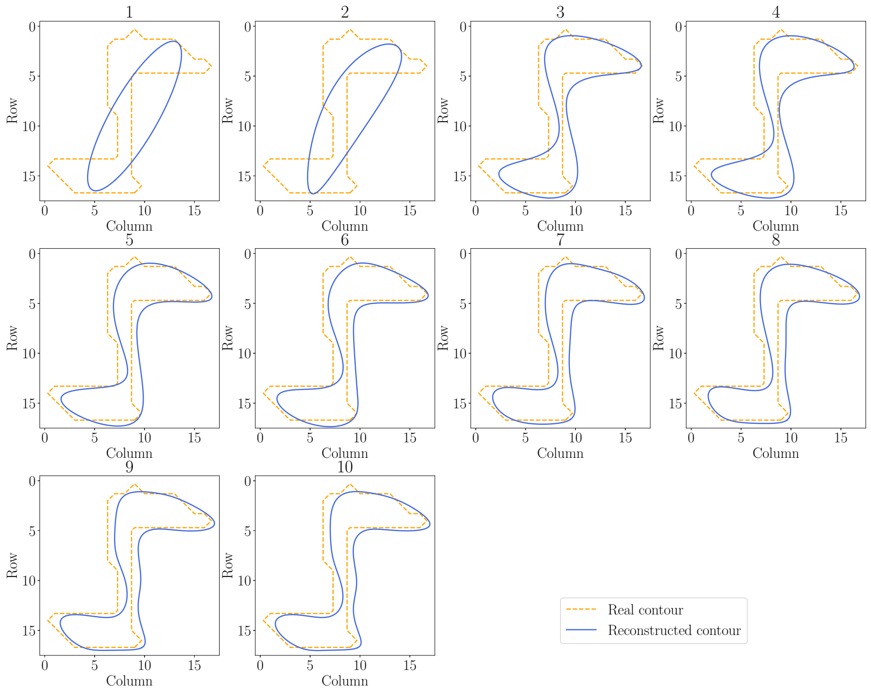


Figure 4.6: The original (orange) contour of the VI trajectory of a CFL together with the approximated (blue) contour of the VI trajectory with increasing number of coefficients.

4.3.3 Object classification

The object recognition results in a vector of size $4 \cdot e$. This vector can be used as input for classification algorithms. This chapter focuses on three methods: logistic regression, random forests, and a simple neural network.

Logistic Regression For a binary problem, logistic regression (LR) predicts the probability that a sample belongs to the ‘1’ class versus the probability it belongs to the ‘0’ class, given one or more independent input variables. For the multi-class problem, a binary logistic regression is fitted for each label, creating a one-versus-all solution. The error function for a binary problem with n samples is:

$$L = -\frac{C}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) + \sum_{i=1}^{m} w_i^2 \tag{4.3}$$

where y_i and $\hat{y}_i$ are respectively the real and predicted output of sample i, and C is a trained regularization parameter that controls the model complexity. When C is small, the first sum in the error function is smaller, implying slower learning and stronger regularization. The second sum consists of the trainable squared weights w_i, one for each input feature m. This sum forms an additional method to prevent overfitting, called L2 regularization, whereby a solution with small weights is preferred over a solution with large weights.

Random Forest A random forest (RF) is an ensemble technique that classifies the data using a collection of decision trees. Each decision tree is trained on a subset of the dataset that has the same size as the original training set, but samples are drawn with replacement. At each node of the decision tree, a feature is selected and the tree is traversed downward (either following left/right branch) by comparing its value to a threshold. Given a new sample, the output of each decision tree is averaged to obtain the final prediction.

The parameters to be trained, are the number of trees t and the maximal number of features f that are considered in each node to decide upon what the optimal split is. In our case, f is at most $4 \cdot e$, the total number of coefficients representing the contour.

Neural Networks The architecture of a neural network (NN) consists of different layers, see Figure 4.7. The first layer is the input layer containing as many nodes as the dimension of a sample (here, $4 \cdot e$). This is followed by one (or more) fully connected layers which are hidden. Each of these layers contains a certain number of nodes that have learnable weights and biases, which define a linear transformation of the node’s inputs, after which a non-linearity is optionally

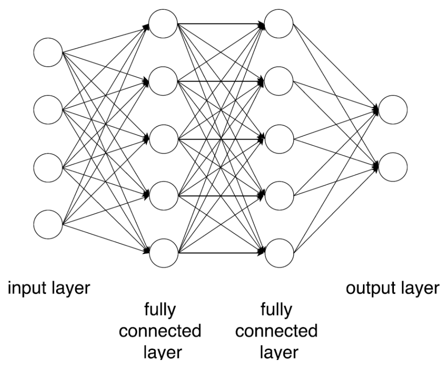


Figure 4.7: A neural network consisting of an input layer, two fully connected layers, and an output layer.

performed. This non-linearity is often obtained by using a rectified linear unit that replaces all negative values by zero. At the end, the output of the last fully connected layer is fed into the output layer. Since the NN is used for classification, the output layer has k nodes, with k equal to the number of classes. The values of the output nodes are chosen between 0 and 1 and sum up to 1, which is achieved by applying the softmax function. In other words, each node represents the probability that the EFDs corresponds to a given label. The output node with the maximal value represents the predicted class.

By providing training data, the weights of the network can be learned in such a way that the predicted output is close to the real output. The learning rate α controls the speed of learning and is a parameter that needs to be trained. A smaller value reduces the risk of overfitting, at the cost of slower learning.

The parameters that are trained for a neural network are the learning rate α, the number of dense hidden layers and the number of nodes in each layer. The last parameters are symbolized by $(h_1, h_2, ..., h_n)$, where h_i represents the number of nodes in the hidden layer i, and n is the number of hidden layers.

4.3.4 Results on submetered data

Benchmark datasets To benchmark the proposed method on submetered data, the 2014 version of PLAID is used, see Chapter 2 and [11].

Evaluation criteria The generalization properties of the model are validated using leave-one-house-out cross-validation as recommended in [6]. Each training set contains data from 54 houses and the test set consists of data from the remain-

Method	Parameter	Range
LR	C	$[10^9, 10^7, 10^5, 10^3, 10^2, 1, 0.01]$
NN	α	$[10^{-5}, 10^{-3}, 1, 10]$
	$[h_1, ..., h_n]$	$[(10), (50), (100),$ $(10, 10), (50, 50), (100, 100)]$
RF	t	$[10, 20, 30, 50, 100, 200]$
	f	$[4 : 4 : 4 \cdot e]$

Table 4.2: The considered parameter ranges.

ing house. This is repeated 55 times. Using leave-one-house-out cross-validation allows us to validate the generalization between different appliances of the same type, as different houses, e.g., contain air conditioners but from different brands.

In order to train the parameters of the machine learning algorithms, grid search is performed. The ranges of the trained parameters are shown in Table 4.2. In order to avoid overfitting while training these parameters, 10-fold cross-validation is performed. The 54 houses are partitioned into 10 equally sized subsets. Each time, 9 of those subsets are used for training and the remaining one is used for model validation. This process is repeated 10 times, with each of the 10 subsets used exactly once as the validation set. The 10 results can then be averaged to obtain a single prediction. The parameters resulting in the best validation score are selected and used on the single test house that was left out in the beginning.

As proposed in [21], the F-measure is used to evaluate the classification performance and as PLAID is imbalanced, this is done for each appliance type separately.

$$F_i = 2 \cdot \frac{precision_i \cdot recall_i}{precision_i + recall_i}, \quad \forall i \in [1, \ldots, a] \tag{4.4}$$

$$precision_i = \frac{TP_i}{TP_i + FP_i} \tag{4.5}$$

$$recall_i = \frac{TP_i}{TP_i + FN_i} \tag{4.6}$$

where TP$_i$, TN$_i$, FP$_i$, and FN$_i$ are respectively the true positives, true negatives, false positives, and false negatives for appliance type i. The number of different appliance types is a. The F-measure for a perfect classifier is 1, whereas a random classifier yields an FF-measure of 0.25. This measure provides more information about the confusion between instances. Its magnitude is mainly determined by the number of correctly labelled samples, but tells us nothing about the instances that are correctly labelled with a 0 (the true negatives). In other words, the precision and recall only focus on the positive class [22]. The TP_i, FP_i and FN_i per appliance type i, are summed for each of 55 test set. In the end, the average of all the F-measures is taken, leading to the so-called macro-average.

$$F_{\text{macro}} = \frac{1}{a} \sum_{i=1}^{a} F_i \tag{4.7}$$

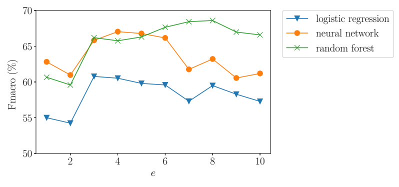
where a is the total number of different appliance types. In addition, the accuracy metric is also reported.

$$\text{Accuracy} = \frac{\#\text{correct predictions}}{\#\text{total predictions}} \tag{4.8}$$

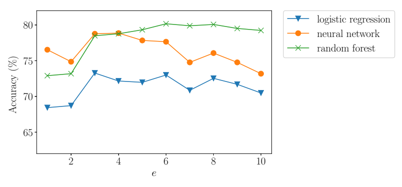
The accuracy for a perfect classifier is 1, whereas a random classifier with k possible classes yields an expected accuracy of $1/k$. This measure does not reveal any insights about the confusion between instances, but captures the information about all classes in one metric. Furthermore, the confusion matrix is plotted showing the correct predictions (the diagonal) and the types of incorrect predictions (the rows represent the predicted class and the columns the real class). This matrix gives a clear view on which appliances are confused with each other. Both the accuracy and the F-measure can be calculated from the confusion matrix.

Classification results In order to obtain the pixelated VI images from the 2014 version of PLAID, the voltage and current are collected over a time interval of 0.33 seconds, resulting in 10000 samples. In these images, the outer contour is detected from which the EFDs are calculated using a varying number of harmonics e. These components are used as input for three classification algorithms: logistic regression, random forests, and neural networks. Figure 4.8 shows the F_{macro}-measure and accuracy when using the different machine learning methods and a varying number of EFDs. The EFDs are calculated from contours that were extracted from 16×16 sized pixelated VI images. Using three or more EFDs does not significanty impact the accuracy for logistic regression and random forests in terms of the F_{macro}-measure and accuracy. In the case of neural networks, the accuracy drops when more than 4 EFDs are used.

The F-measure per appliance and confusion matrix when classifying the appliances using logistic regression, random forests, and neural networks using 3 EFDs, are shown in Figure 4.9. The values in the confusion matrix represent the absolute number of detected appliances. The color represents per appliance (per row) the relative number of detected appliances with respect to the total number



(a)



(b)

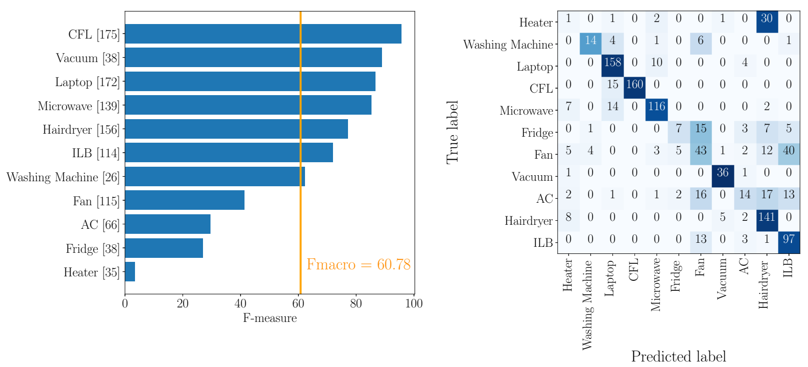
Figure 4.8: The F_{macro} and accuracy of the three classifiers using an increasing number of EFDs e.

of that appliance. The F_{macro} is respectively 60.78%, 66.20%, and 65.83%. For all appliances except the washing machine (when classified by the random forest or neural network), fan, air conditioner (AC), fridge, and heater, the F-measure is higher than F_{macro}. When investigating the confusion matrix in Figure 4.9, it is clear that the washing machine, AC, and fridge are confused with the fan, the fan and AC with the ILB, and the AC, and heater with the hairdryer. Common electrical components can explain some of the confusion: the AC also has a fan, the hairdryer and heater both contain a heating element, and the washing machine, fridge, and fan all have a motor. For all three methods, the same appliance types are difficult to classify. However, for random forest and neural network, the number of correctly classified appliances is higher. The low performing classes are heater, fridge, and AC.

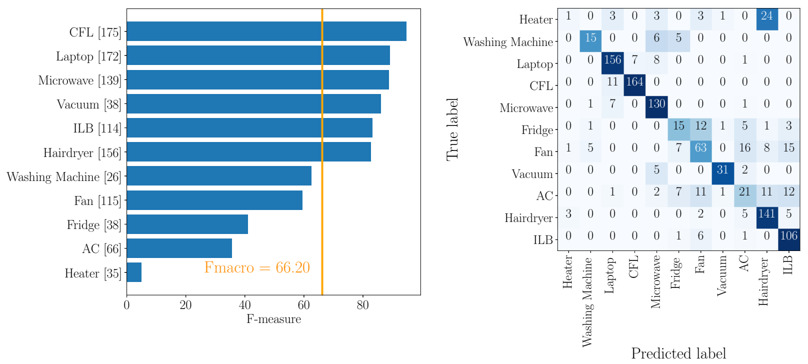
The results for a random forest are comparable to those reported in [6], where 16×16 pixelated VI images were used as plain input for a random forest, see Figure 4.10. Originally, only the confusion matrix was reported. We added the F_{macro} for each appliance. The reported accuracy is 81.75%, and $F_{\text{macro}} = 70.41\%$. The obtained accuracy by the random forest using 3 EFDs is 78.49% and $F_{\text{macro}} = 66.20\%$. For both classifiers, the same appliances cause confusion. A key advantage of the reported approach over [6] is that the required storage for the features is far less, albeit at a small additional cost for computing the contours. Only 12 values per sample need to be stored, compared to 256 values for the original 16×16 pixelated image.

4.3.5 Results on aggregated data

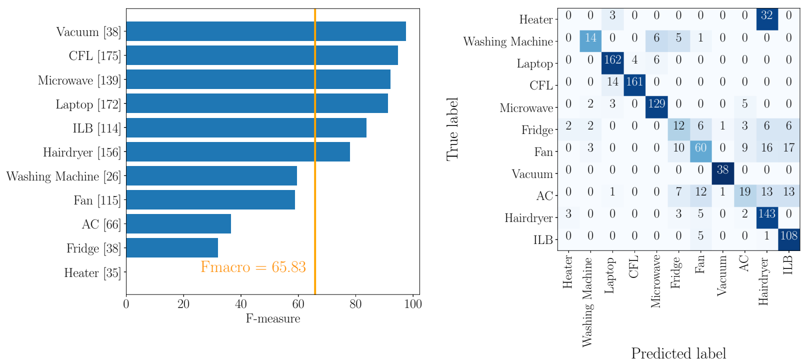
In the previous subsection, the proposed method was benchmarked on high frequency submetered data. The obtained appliance classification results were a first step towards a more realistic NILM setting starting from the aggregated power measurements. It was a very meaningful step, as typically appliances are turned on/off one at a time, and the single appliance current (and thus VI trajectory) can



(a)



(b)



(c)

Figure 4.9: The F-measure per appliance and confusion matrix for the 2014 version of PLAID when using 3 EFDs and applying (a) logistic regression, (b) random forest, and (c) neural networks. The number of samples per appliance type is mentioned between the brackets. AC = air conditioning, CFL = compact fluorescent lamp, ILB = incandescent light bulb

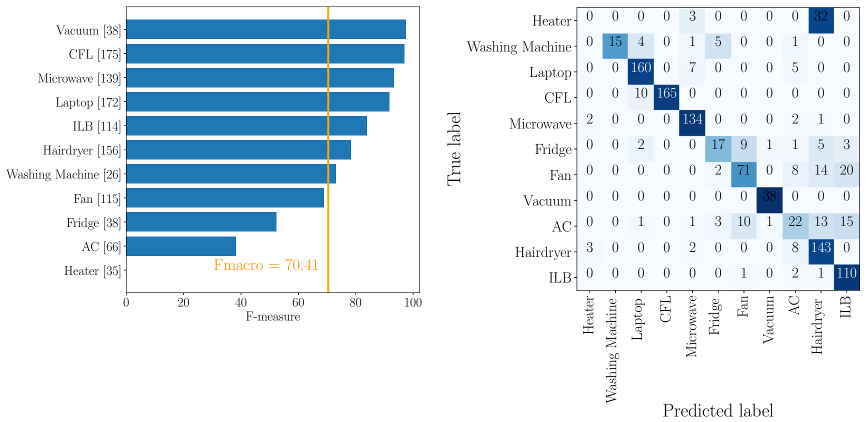


Figure 4.10: The F-measure per appliance and confusion matrix for the PLAID dataset when the random forest trained on the 16×16 *pixelated VI images, is used [11]. The number of samples per appliance type is mentioned between the brackets.*

be extracted from the aggregated measurements by considering the difference in current before and after the event. This subsection discusses that this was indeed the correct assumption to take. First, it is explained how submetered current and voltage signals can be obtained from the aggregated data. Then, the results of the method on aggregated data are presented. It is investigated if submetered data is necessary for training the algorithms or if aggregated data is sufficient. In addition, it is also investigated what the importance of the image size is.

Obtaining submetered current and voltage signal In order to obtain the pixelated VI images from aggregated data, the current and voltage before and after all events are selected. If the event is caused by the activation of an appliance and only one appliance is activated, then the current i and voltage v of the activated appliance is obtained by:

$$i = i_{\text{after}} - i_{\text{before}} \tag{4.9}$$

$$v = v_{\text{after}} \tag{4.10}$$

where i_{after} is one current cycle when the appliance reaches steady state behavior after the event (here, this is set to occur 1 second after the event), i_{after} is one current cycle before the event, and v_{after} is one voltage cycle after the event. The current cycles i_{after} and i_{before} are aligned in such a way that the first point of the corresponding voltage cycles v_{after} and v_{before} is zero, see Figure 4.11. If the event is caused by the deactivation of an appliance and only one appliance is deactivated, then the current i and voltage v of the deactivated appliance is obtained by:

$$i = i_{\text{before}} - i_{\text{after}} \tag{4.11}$$

$$v = v_{\text{before}} \tag{4.12}$$

From the obtained per-appliance/submetered i and v signals, the pixelated VI image is created.

Benchmark datasets To benchmark the proposed method on aggregated data, the data added to the 2018 version of PLAID is used, see Chapter 2. For this dataset, it holds that only one appliance is turned on/off at a time. In addition, the events are labelled making it straightforward to extract the per-appliance v and i.

Evaluation criteria Evaluating the results is done using the per-appliance F-measure, F_{macro}-measure, and the confusion matrix. To investigate if submetered data is necessary for training the algorithm, two scenarios are considered: the training of the classification algorithms uses (1) the corresponding submetered data, and (2) a random part of the aggregated data. For the first scenario, all the aggregated data is used for testing and for the second one, 4-fold cross validation is performed, ensuring us that every sample was used once for testing. This approach guarantees a fair comparison between the two approaches. The accuracy is not mentioned as the same conclusion as when using the F-measure can be made concerning the best method, the number of EFDs to use, and the most informative image size.

Classification results In the obtained single appliance pixelated VI images with varying image sizes, the outer contour is detected from which the EFDs are calculated using a varying number of harmonics e. These components are used as input for three classification algorithms: logistic regression, random forests, and neural networks (as before, see Figure 4.8). Figure 4.12 show the F_{macro} of the three classifiers trained on submetered and aggregated data for different image sizes.

It is clear that training on the aggregated data leads to higher F_{macro} than training on the submetered data. This can be explained intuitively by the fact that when training uses aggregated data, the training data contains the same noise (caused by other active appliances) as present in the test data. This is important for practical reasons, as in a household, the users will try to avoid the labor of submetering different appliances. Using three or more EFDs does not significanty impact the accuracy for logistic regression and random forest in terms of the F_{macro}-measure.

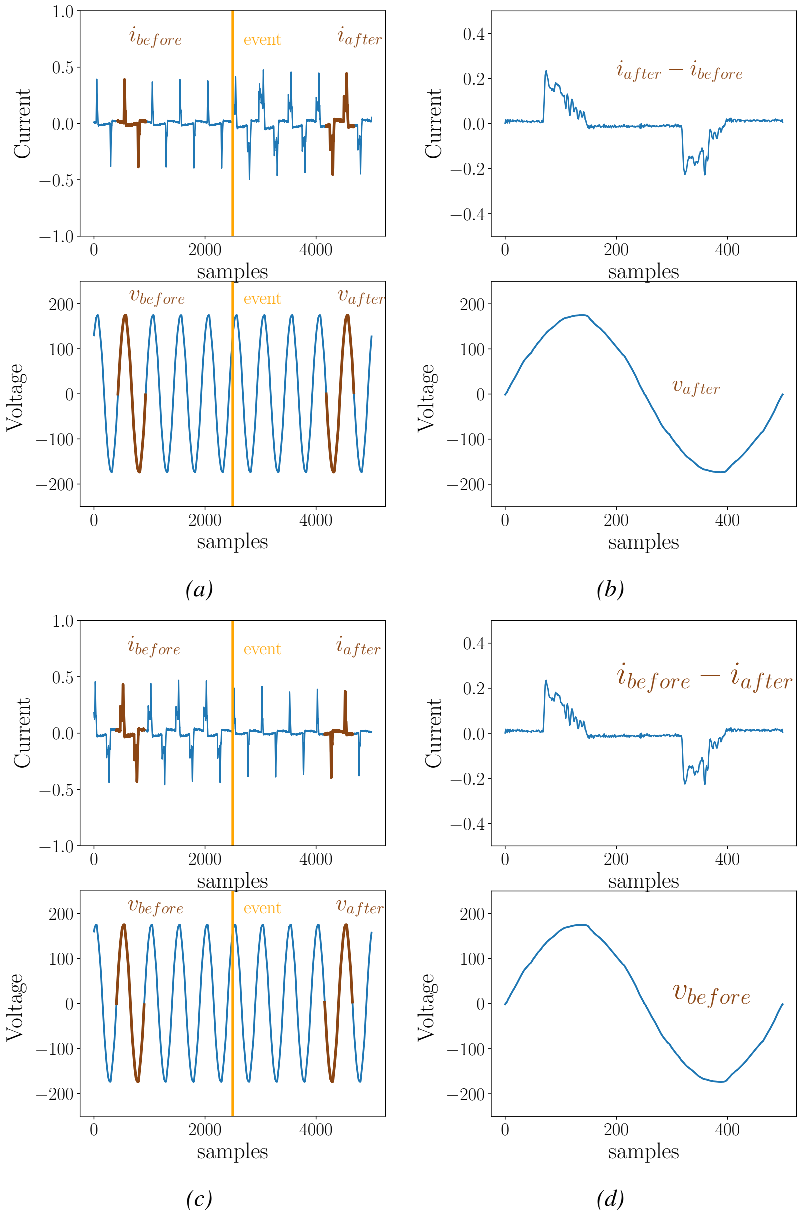


Figure 4.11: The aggregated current and voltage when an appliance is (a) activated and (c) deactivated, together with the current and voltage of the appliance causing the event (b) and (d).

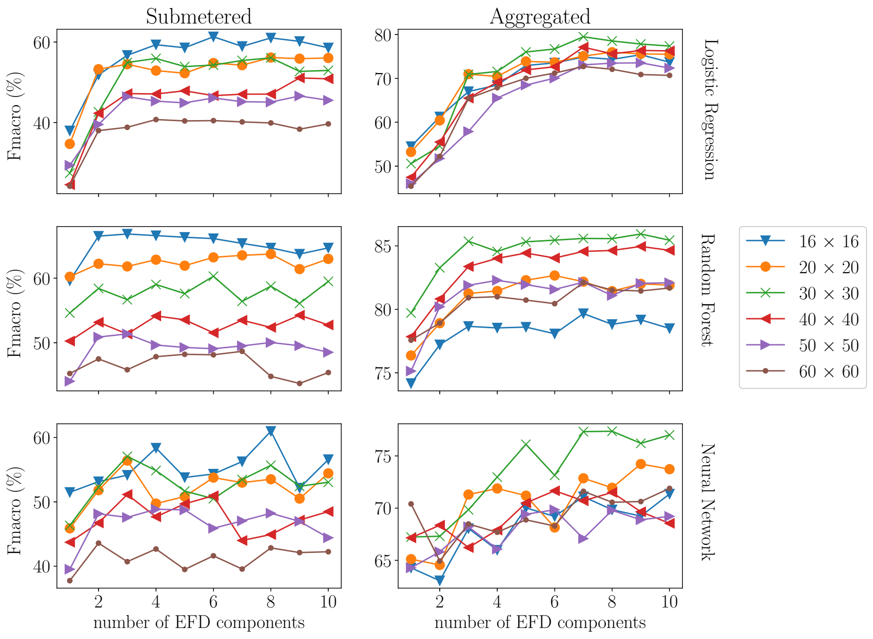
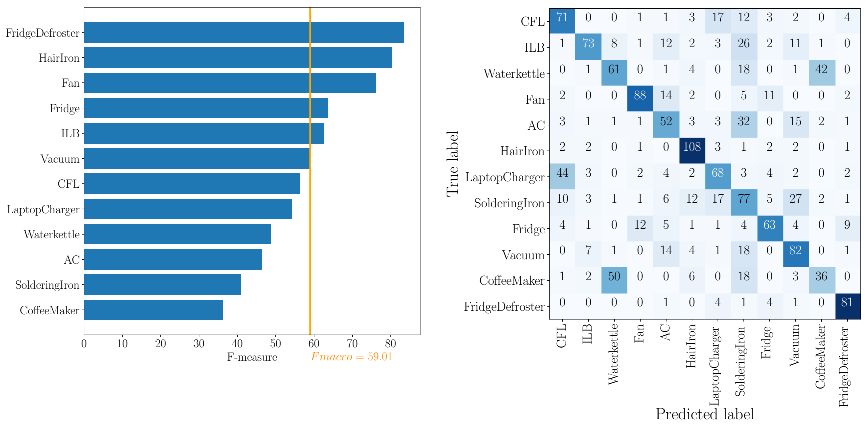


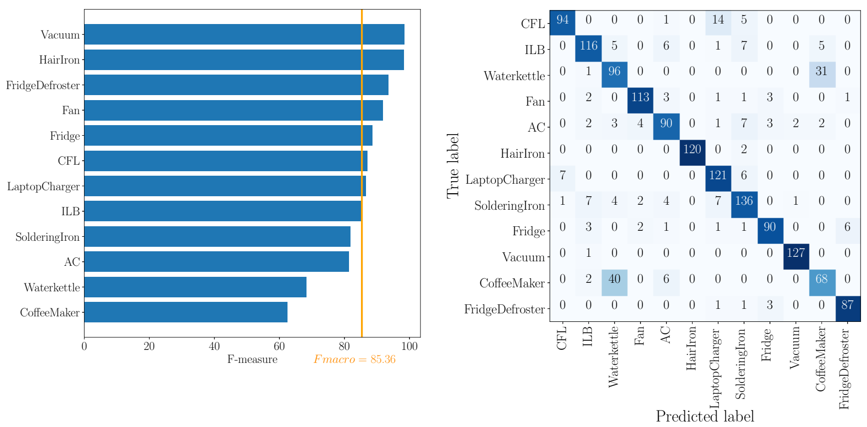
Figure 4.12: The F_{macro} of the three classifiers using an increasing number of EFDs e for different image sizes and when trained on submetered data (left column) or aggregated data (right column).

In the case of neural networks, no clear improvement can be seen in the F_{macro}-measure when using more EFDs, except when trained on aggregated data and using an image of size 30 × 30 to calculate the contours and corresponding EFDs. The pixelated image size is altered between $[16, 20, 30, 40, 50, 60]$. As shown, increasing the image size does not lead to an improvement in the F_{macro} when using EFDs as input for the classifiers. When trained on submetered data, the EFDs calculated from the contours from the smallest image (16 × 16) lead to the best F_{macro}, and those from the largest (60×60) to the worst. The best F_{macro} is obtained when the random forest is trained on aggregated data and the 3 EFDs are calculated form images of the size 30×30. Further research is needed to explain these results.

The F-measure per appliance and the confusion matrix when the number of EFDs is 3, the image size is 30 × 30 and a random forest is trained using submetered and aggregated data are shown in Figure 4.13. When using submetered data for training (Figure 4.13 a)), the water kettle and coffeemaker are confused with each other (both resistive heaters). Additionally, some other confusion exists: the CFL is confused with the laptop charger (both non-linear loads) and the AC with the soldering iron. When training uses aggregated data (Figure 4.13 b)), a lot of



(a)



(b)

Figure 4.13: The F-measure per appliance and confusion matrix for the aggregated data in the 2018 version of PLAID with $e = 3$ EFD components, the image size is 30×30 and the random forest is trained using (a) submetered, and (b) aggregated data. AC = air conditioning, CFL = compact fluorescent lamp, ILB = incandescent light bulb

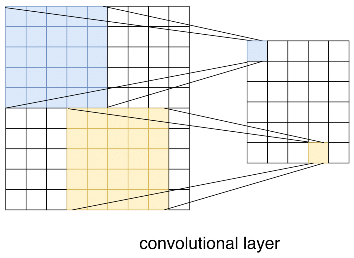


Figure 4.14: A convolutional layer.

confusion is resolved. Now only the water kettle and the coffeemaker are confused with each other, and the CFL with the laptop charger.

4.4 Convolutional neural networks

Instead of converting the VI trajectory into a pixelated image, it can also be converted into a weighted pixelated image. A CNN can then be applied in order to classify the samples. CNNs are a type of neural network (NN) that are often used in computer vision because they are highly suitable for classifying images [10]. The (C)NN takes training samples as input and classifies them by automatically extracting informative features from the data. To this end, an architecture and training procedure is needed, described in Subsection 4.4.1 and 4.4.2 respectively. Subsection 4.4.3 benchmarks this method on the 2014 version of PLAID and WHITED, and Subsection 4.4.4 benchmarks it on the aggregated data added to the 2018 version of PLAID.

4.4.1 Architecture

In Section 4.3.3, the architecture of a NN is explained. When using a $n \times n$ weighted pixelated image as input, the input layer contains n^2 nodes. The output layer has k nodes with k equal to the number of classes.

To create a CNN from a NN, convolutional layers are added. These are placed between the input and output layers as desired and are consequently hidden. The main difference between a convolutional and fully connected layer is that each node in a convolutional layer is connected to a small region of the input matrix exploiting local correlation, see Figure 4.14. In each node, a convolution is performed by adding each element of the input image to its local neighbours, weighted by a matrix called a filter. After the convolutional layer, it is common to implement a pooling layer to downsample the convolved matrix. This reduces the spatial size

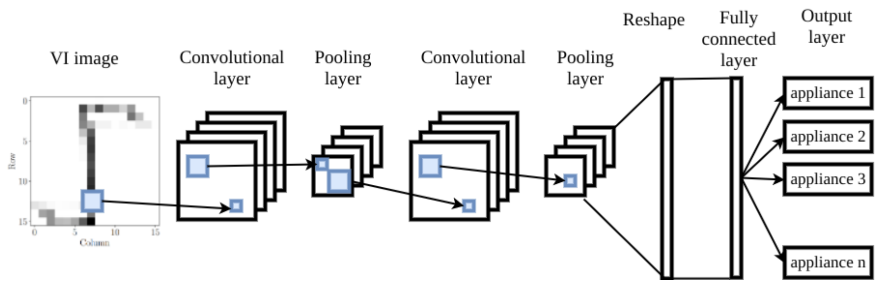


Figure 4.15: The architecture of the implemented CNN taking as input the VI image.

of the representation and the number of parameters, and hence also manages overfitting. This downsampling is achieved by sliding a $d \times d$ window over the input (here, with $d = 2$) and each time outputting the largest element of the window. Different flavor exist for pooling layers, e.g., a max pooling layer will maintain the maximal value of the original window, and the min pooling layer the minimal value.

The CNN implemented in this chapter has the following structure, see Figure 4.15: it takes as input the weighted pixelated VI image (a $n \times n$ matrix, with $n = 50$), and has the following hidden layers: a convolutional layer with f filters of size 5, a max pooling layer, another convolutional layer with f filters of size 5, another max pooling layer, a fully connected layer with n^2 nodes and an output layer with k nodes. The number of filters f is set to 50. The number of output nodes k is determined by the number of different appliances present in the dataset (i.e., the number of classes). An analysis of alternative parameter settings for n and f showcased no significant changes in the results. The activation function of the hidden layers and output layer are respectively the rectified linear, and softmax function.

4.4.2 Model training

Once the architecture is specified, a training procedure is initiated so that the CNN learns to classify the different classes. To this end, multiple training examples are needed. These are images $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ labelled with their corresponding class $\mathbf{t} = (\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n)$ where $\mathbf{t}_i$ is a 1-of-k coding of the classes. The aim of the training is to find weights and biases such that a cost function is minimized. Since the class labels are categorical, the cost function is defined as the cross-entropy function [23]:

$$L = -\sum_{i=1}^{n}\sum_{j=1}^{k} t_{i,j} log(y_{i,j}) \tag{4.13}$$

where the predicted outputs y_i depend on all the weights and biases in the CNN. The cost function L is minimized using gradient descent, and decreases as the predicted output y_i approximates the real output t_i for all n training samples. As such, the whole CNN learns its weights and biases in such a way that the filters are able to represent spatial connections and features in the data. The reader is referred to the online book 'Neural Networks and Deep Learning' [23] to learn more about (C)NNs and their training.

4.4.3 Results on submetered data

Benchmark dataset The proposed method is benchmarked on the 2014 version of PLAID (see Chapter 2, and [11]) and WHITED [12]. WHITED [12] is a public dataset including submetered v and i measurements sampled at 44kHz for 46 different appliance types. For each appliance type, 1 to 9 different appliances are available. For each appliance, 10 start-up events are measured, resulting in a total of 1100 measurements.

Evaluation criteria As done in Section 4.3.4, the generalization properties of the classifier are validated using leave-one-house-out cross-validation, as recommended in [6]. For the PLAID dataset, this can be done straightforwardly as the data is divided per house (55 houses in total). In the WHITED dataset, the annotation of measurement locations is not available. Houses are created artificially by assigning each appliance of each appliance type randomly to one house. The total number of houses is set at 9, which corresponds to the maximum number of appliances per appliance type. As a consequence, appliance types with only one appliance are left out. The final dataset for the experiment contains 22 appliance types.

Again, the F_{macro}, accuracy, and confusion matrix are used to express the performance of the classifier.

Classification results on PLAID In order to obtain the weighted pixelated VI images for the 2014 version of PLAID, the voltage and current are measured over 20 cycles of the voltage signal, resulting in 10000 samples. In PLAID, there are $K = 11$ appliance types. The F-measure per appliance and the confusion matrix for PLAID is shown in Figure 4.16. The $F_{\text{macro}} = 77.60\%$. For all appliances except the washing machine, fan, fridge and air conditioner (AC), the F-measure

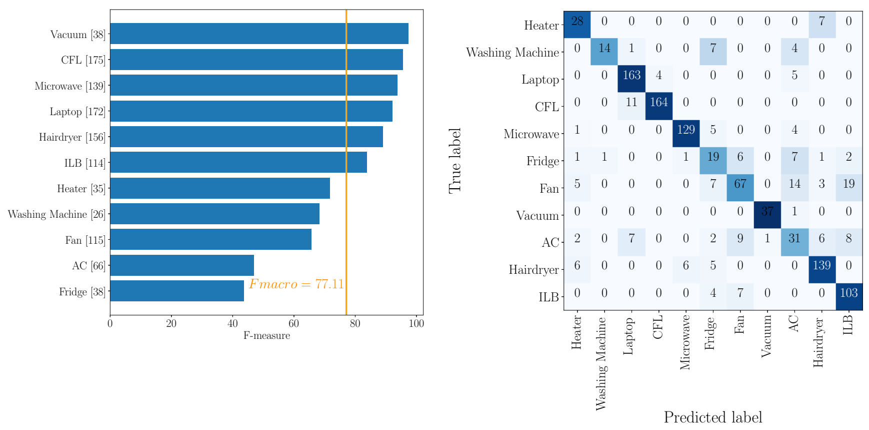


Figure 4.16: The F-measure per appliance, and confusion matrix for the 2014 version of PLAID when the CNN for $n = 50$ and $f = 50$ is used. The number of samples per appliance type is mentioned between the brackets. AC = air conditioning, CFL = compact fluorescent lamp, ILB = incandescent light bulb

is higher than F_{macro}. When investigating the confusion matrix, it is clear that the lower F-measure obtained for the washing machine, fan, fridge and AC is caused by confusion among them. Common electrical components can explain this phenomenon: the AC also has a fan, and both the washing machine and fridge have a motor. The accuracy for the 2014 version of PLAID is 83.24%.

In [24], some F-measure results on the 2014 version of PLAID are given, however, the train-test split approach is not done in a leave-one-house-out manner, making comparison pointless. In Figure 4.10, the results reported in [6] are shown where 16×16 pixelated VI images are used as plain input for a random forest. Originally, only the confusion matrix was reported. We added the F_{macro} for each appliance. The reported accuracy (81.75%) and F_{macro} (70.41%) are lower than those achieved by using the method presented in this section which are respectively 83.24% and 77.60%. Also the accuracy (78.49%) and F_{macro} (66.20%) obtained by the previous presented method, are lower. This gain in performance can be explained by the fact that the new classifier is more capable of classifying heaters, as can be seen by comparing their respective confusion matrices.

Classification results on WHITED In order to obtain the weighted pixelated VI images for the WHITED dataset, the voltage and current are measured over a time interval of 20 cycles of the voltage signal, resulting in 17600 samples. In WHITED, there are $K = 22$ appliance types. The F-measure per appliance for the

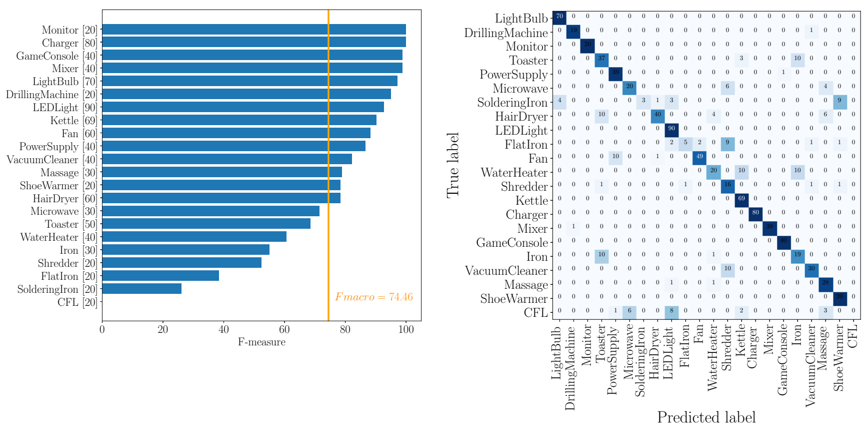


Figure 4.17: The F-measure per appliance, and confusion matrix for WHITED when the CNN for $n = 50$ and $f = 50$ is used. The number of samples per appliance type is mentioned between the brackets.

WHITED dataset is shown in Figure 4.17. The F_{macro} equals 75.46%. For the soldering iron, flat iron, and compact fluorescent lamp (CFL), the method performs poorly. When investigating the confusion matrix, it is clear that the soldering iron is confused with the light bulb and the shoe warmers (all having resistive heating and more measurements of the light bulbs are present), the flat iron with the shredder (no common electrical components but both only have 20 examples) and the CFL mainly with the charger (both having an element limiting the voltage and more measurements of the chargers are present). This confusion caused by the FNs explains the low F-measure. The FPs for these classes are quite low.

In [24], some F-measure results on the WHITED dataset are given, however the train-test split approach is not done in a leave-one-house-out manner, making comparison pointless.

4.4.4 Results on aggregated data

In the previous subsection, the proposed method was benchmarked on high frequency submetered data. Just like the results of Subsection 4.3.4, the obtained appliance classification results were a first, meaningful step towards a more realistic NILM setting starting from the aggregated power measurements. This subsection, like Subsection 4.3.5, discusses the results obtained in this realistic setting. It is investigated if submetered data is necessary for training the algorithms or if aggregated data is sufficient. In addition, it is also investigated what the importance of the image size is.

Benchmark datasets To benchmark the proposed method on aggregated data, the data added to the 2018 version of PLAID is used, see Chapter 2. For this dataset, it holds that only one appliance is turned on/off at a time. In addition, the events are labelled making it straightforward to extract the per-appliance v and i, as explained in Subsection 4.3.5.

Evaluation criteria Evaluating the results is done using the per-appliance F-measure, F_{macro}-measure, and the confusion matrix. As for the approach in Subsection 4.3.5, two scenarios are considered to investigate if submetered data is necessary for training: the training of the classification algorithms uses (1) the corresponding submetered data, and (2) a random part of the aggregated data. For the first scenario, all the aggregated data is used for testing and for the second one, 4-fold cross validation is performed, ensuring us that every sample was used once for testing. This approach guarantees a fair comparison between the two approaches. This approach guarantees a fair comparison between the two approaches. The accuracy is not mentioned as the same conclusion as when using the F-measure can be made.

Classification results The pixelated VI images from the aggregated data added to the 2018 version of PLAID are obtained in the same wasy as explained in Subsection 4.3.5. The F_{macro} for varying image sizes, and using aggregated and submetered data for training is shown in Figure 4.18. Training on the aggregated data leads to higher performance than when training is done on submetered data, just like was the case when EFDs are used as input (see Section 4.3). This can be explained intuitively by the fact that when training uses aggregated data, the training data contains the same noise (caused by other active appliances) as present in the test data. This is important for practical reasons, as in a household, the users will try to avoid the labor of submetering different appliances. Using image sizes larger than 30 × 30 does not considerably improve the F_{macro}-measure.

The F-measure per appliance and the confusion matrix when the image size is 30 × 30, and using submetered and aggregated data for training, are shown in Figure 4.19. When using aggregated data for training (Figure 4.19b), only the the water kettle and the coffeemaker are confused with each other (both resistive heaters) in respectively 45.31% and 31.90% of the samples. When using submetered data (Figure 4.19a)), 46.55% of the coffeemaker samples are confused with the water kettle and 13.28% the other way around. Additionally, also the ILB and AC are confused sometimes with the coffeemaker (respectively 15.71% and 9.65%). Further research is necessary to explain why there is an asymmetry in the

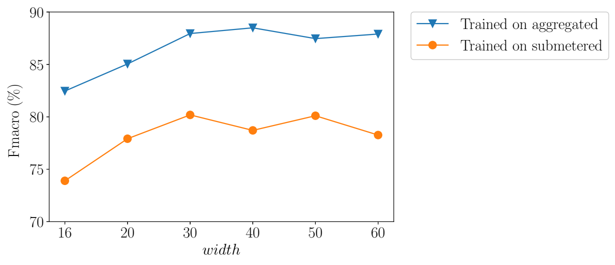


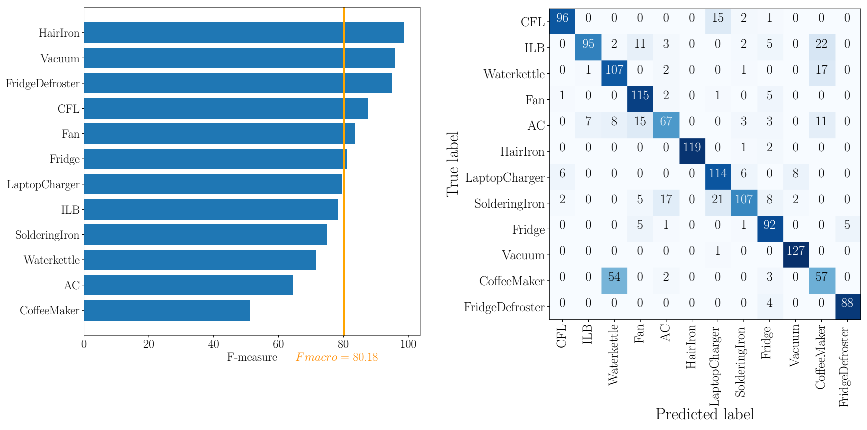
Figure 4.18: The F_{macro} for the aggregated data of the 2018 version of PLAID when CNN for $f = 50$ and varying image size n is used. The training is done using submetered or aggregated data.

confusion and why the confusion is less present when using aggregated data for training than when using submetered data.

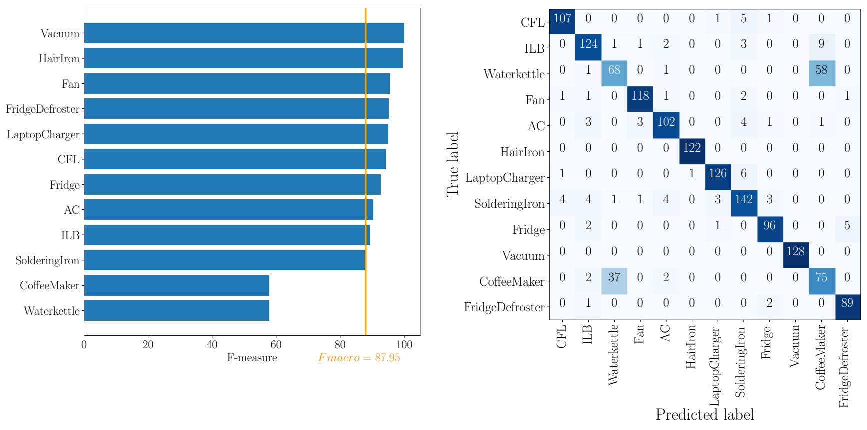
When using submetered data for training, the F_{macro} of this method using CNNs (80.38%) is significantly higher than the one obtained by the previous method based on EFDs (72.47%). The method using CNNs is better in classifying the AC and there is less confusion between the water kettle and the coffee maker. When using aggregated data for training, the F_{macro} of this method using CNNs (87.95%) is slightly higher than the one obtained by the previous method based on EFDs (85.31%). Both the method using CNNs and based on EFDs, confuse the water kettle and coffee maker with each other, but the method using CNNs is better in classifying the CFL.

4.5 Current as feature

The two methods presented before do not include any information concerning the current peaks as both the binary and pixelated VI image are normalized. Consequently, two appliance types having the same electrical components will be classified as the same nevertheless there is a difference in the absolute value of the current peak or absolute power consumption. The approach in this section adds this information as an additional input feature for the previous methods, see Subsection 4.5.1. Another possibility is presented in Subsection 4.5.2, where the current is used as only feature. The results on aggregated data are presented in Subsection 4.5.3. No results are shown on submetered data as the submetered data exhibited calibration problems, comprising the maximal (peak) values of the measurements, see Chapter 2.



(a)



(b)

Figure 4.19: The F-measure per appliance and confusion matrix for the aggregated data in the 2018 version of PLAID when the CNN for $n = 30$ and $f = 50$ is used, and is trained using (a) submetered, and (b) aggregated data . The number of samples per appliance type is mentioned between the brackets. AC = air conditioning, CFL = compact fluorescent lamp, ILB = incandescent light bulb

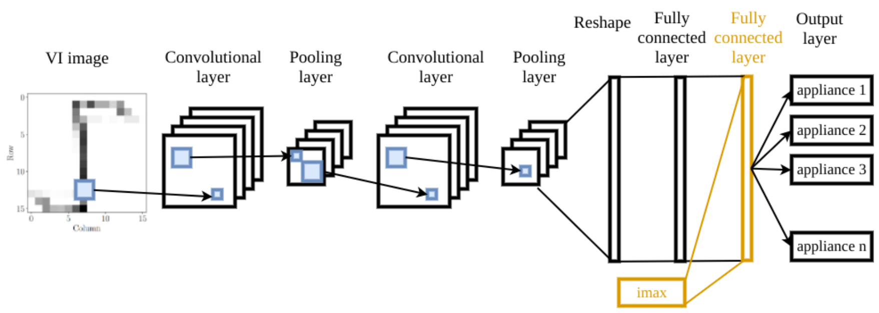


Figure 4.20: The architecture of the implemented CNN taking as input the VI image and the current consumption.

4.5.1 Current peak as extra feature

The current peak is the maximal value in a cycle of the current signal once the appliance reaches steady state behaviour, i_{max}.

Object recognition For the object recognition method presented in Section 4.3, four features per EFD coefficient are used. Adding the current consumption as an extra feature, can be done as simple as adding i_{max} to the vector of $4 \cdot e$ features (for e EFD components).

Convolutional neural networks For the convolutional neural network, the input is represented as an $n \times n$ image. Adding a single value (i_{max}) to this input is not meaningful as the CNN assumes that neighboring values are correlated. Instead, an extra layer is added between the last hidden and output layer taking as input the output of the last hidden layer and i_{max}, see Figure 4.20.

4.5.2 Current as only feature

Instead of using the current peak values as an extra feature, it can also be used as only feature. Then one cycle of the current signal when the appliance reaches steady state behaviour, is used as input. Note, that this current cycle is aligned with the voltage cycle in such a way that the first point of voltage cycle is zero. This is necessary as otherwise the information concerning a possible phase shift is lost. Every sample of the cycle is seen as a separate feature. So when normalization is performed, every feature is scaled separately. If the sampling frequency is 30 kHz and the power line frequency is 50Hz, then the number of samples/features in a cycle is 500. The cycles are used as input for the same classifiers presented in Section 4.3: logistic regression, random forest, and neural network with two fully connected layers and an output layer having as non-linearity function respectively the rectified linear unit and softmax function. Additional parameters are optimized, see Table 4.2.

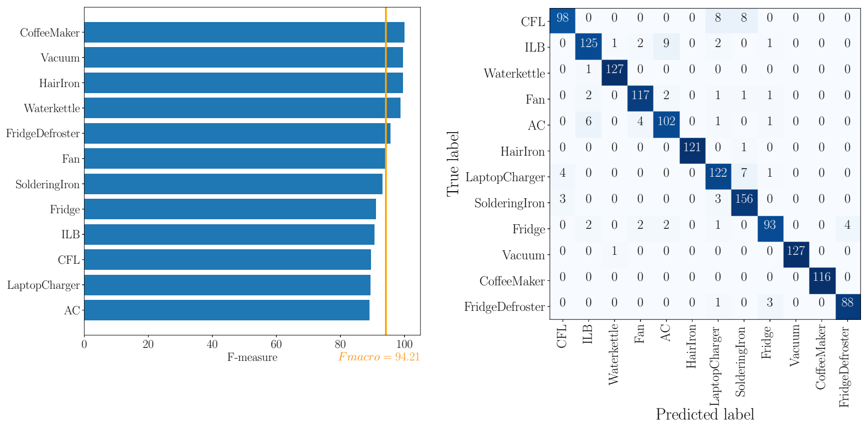
4.5.3 Results on aggregated data

Benchmark dataset To benchmark the proposed method on aggregated data, the data added to the 2018 version of PLAID is used, see Chapter 2.

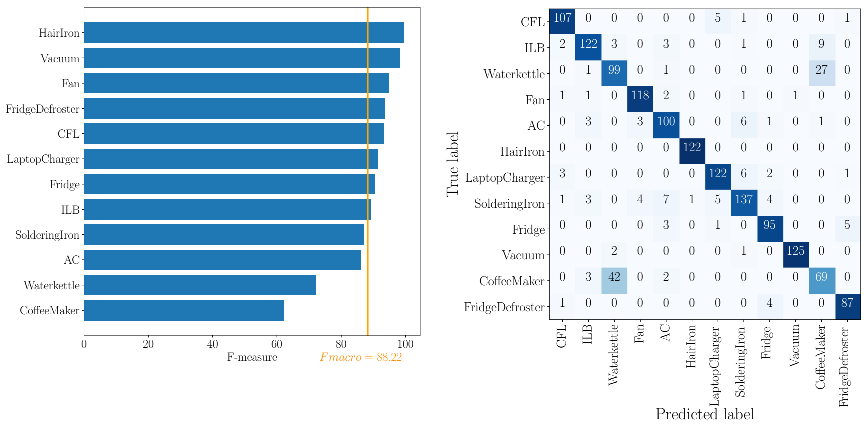
Evaluation metric Evaluating the results is done using the per-appliance F-measure, F_{macro}-measure, and the confusion matrix. In contrast to previous sections, only one scenario is considered: the training of the classification algorithms uses a random part of the aggregated data. 4-fold cross validation is performed, ensuring us that every sample was used once for testing. As previous sections (Subsection 4.3.5 and 4.4.4) made clear that training using aggregated data is better, the scenario where the training uses the corresponding submetered data is omitted.

Classification results The results when the current consumption is used as extra feature is shown in Figure 4.21. Only the result for an image size of 30×30 is presented as previous sections shows that this image size leads to the best result. Also, for the object recognition method using the EFDs as features, only the result for the random forest is presented as Section 4.3 shows this method achieves the best result. The F_{macro} obtained by when using the current as an extra feature (94.21% for the object recognition method and 88.22% for the CNN) is higher when the current is not used as extra feature (respectively 85.31% and 87.95%). The increase in the F_{macro}-measure is much higher for the object recognition method than for the CNN: the random forest is more capable than the CNN to extract extra information from i_{max}. For the object recognition almost no confusion exists. For the CNN, it is again the case that the water kettle and the coffeemaker are confused with each other.

The result when the current consumption is used as only feature is shown in Figure 4.22. An F_{macro} of 96.69% is obtained. This is the highest F_{macro} obtained when compared to the previous presented method. Only the AC is confused with the fan, which is stems from the fact that the AC contains a fan.

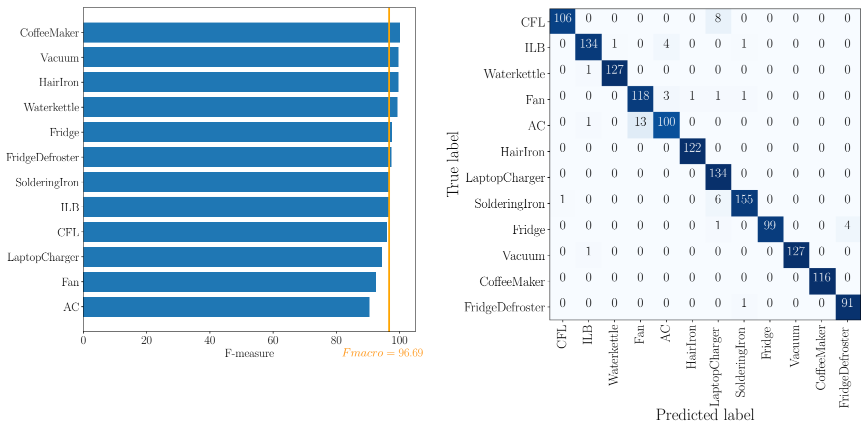


(a)



(b)

Figure 4.21: The results for the aggregated data of the 2018 version of PLAID for (a) the random forest using 3 EFDs (calculated from an image of size 30×30*) and* i_{max} *as input, and (b) the CNN using* $f = 50$ *and as input an image of size* 30×30 *and* i_{max}*.*



(a)

Figure 4.22: The results for the aggregated data of the 2018 version of PLAID for when current cycles of steady state behaviour are used as input for a random forest.

4.6 Handling imbalance

When a dataset is imbalanced, there are minority and majority appliance types, with the first represented by more samples. Due to the class imbalance, it is possible that the classifier focuses too strongly on the majority types during training, thereby ignoring the minority types. Subsection 4.6.1 gives an overview of relevant previous research. Next two types of methods to deal with an imbalanced dataset are examined: resampling the dataset (Subsection 4.6.2) and reweighing the error function (Subsection 4.6.3). The results on the 2017 version of PLAID are presented in Subsection 4.6.4 when using the pixelated VI image as input for multiple classifiers. It is important to note that these methods influence the train phase of the machine learning methods, while leaving the test phase intact, i.e., the approaches respectively modify the train set and the error function used for training the classifier's parameters.

4.6.1 Related work

In the NILM literature, no previous work can be found for handling the appliance type imbalance. However, research exists that deals with the imbalance caused by the difference in active and idle time of appliances. This occurs in NILM datasets containing consumption patterns over time. In [25], which determines how much energy a specific appliance consumes at any given moment using regression, the imbalance caused by the difference in activation and idle time of appliances is present. To handle it, they propose the usage of the target-weighted root mean squared error as an alternative error metric for optimizing the regression. In [26] where temporal sequence classification algorithms are researched, the same imbalance is counteracted with under-sampling: reducing the number of majority samples (the idle samples) so that it equals the number of minority samples (the active samples). It is qualitatively mentioned that this approach is preferred to over-sampling (increasing the number of samples of the minority till it equals the number of the majority) or leaving the data as-is, however no quantitative comparison of the results is shown.

Although no literature can be found where these methods are applied on NILM data to solve the appliance type imbalance, these methods are well investigated for classical machine learning methods. These methods can change the distribution in the dataset by resampling the classes. Some methods oversample the dataset, like in [27] where they want to predict age and gender from images. Simple oversampling (increasing the number of samples in the minority classes by duplicating samples) is effective, but one should be aware of overfitting [28]. To avoid overfitting, more advanced oversampling can be used, like SMOTE [28] where new samples are synthesized. For this latter, one could also use a smart home simulator like SmartSim [29] or AMBAL [30]. However, it must be mentioned that these are two frameworks for low frequency data concerning consumption patterns over time and not for high frequency data concerning the activation of appliances. Therefore, these simulators can not be used to generate PLAID-like data. Another possibility to change the distribution in the dataset is to undersample the dataset as in [31] where samples of the majority class are randomly deleted. In the case of [32] where a decision tree learner is used, undersampling is preferred over oversampling. But in [33] where convolutional neural networks are used for classifying imbalanced classes, it is found that oversampling performs better than undersampling. In [34], the undersampling is done multiple times and an ensemble of classifiers is trained upon them. Another way to handle imbalance is to change the classifier so that different misclassification errors incur different penalties [35]. In this chapter over- and under-sampling, synthesizing samples, balanced bagging, and altering the weight function will be researched in a NILM context.

4.6.2 Modifying the dataset

Changing the dataset so that it becomes balanced can be done in several ways:

- **Over-sampling**: adjust the distribution of the dataset by replicating samples of the minority types till the number of the majority types is reached [36]. For PLAID, this results in a train set of 2673 samples on average. This is a significant increase when compared to the size of the normal train set of 1769 samples.

- **Under-sampling**: adjust the distribution of the dataset by reducing the number of samples in the majority types to the number of the minority types [36]. For PLAID, this results in a train set of 803 samples on average. This is a significant decrease when compared to the size of the normal train set of 1769 samples.

- **Synthesizing samples**: instead of replicating samples from the minority types, artificial samples are created. In this chapter, the synthetic minority oversampling technique (SMOTE) [28] is used, where the artificial samples are formed by interpolating two neighbouring samples of a minority type. For example, having samples A and B with n features, then the new interpolated sample C is constructed by:

$$\begin{aligned}
&C[i] = A[i] + \text{gap} \times \text{diff}, \forall i \in [1, n] \\
&\text{diff} = B[i] - A[i] \\
&\text{gap} = \text{random number between } 0 \text{ and } 1
\end{aligned}$$

The formula can be easily extended for multi-dimensional features by applying the formula in each dimension. For PLAID, this procedure results in a train set of 2673 samples on average, just as in the case of over-sampling.

- **Balanced bootstrapping** (BB): is based on a probabilistic approach allowing for identification of dataset characteristics (such as dimensionality, sparsity, etc.) that exacerbate the problem [34]. It goes as follows:

  1. randomly select instances from the train set with replacement (bootstrap the dataset). Do this multiple times, each resulting dataset is called a bootstrap.
  2. under-sample each bootstrap, like explained above.
  3. train a classifier on each bootstrap.
  4. when classifying the test instances, the majority vote of the classification of all separate classifiers is taken as outcome.

  For this dataset 10 bootstraps are created, each with size equal to the original train set, but now with the difference that some samples can be present more than once.

4.6.3 Adaptation of the classifier

The error function of the classifier can be changed such that misclassification of the minority types is penalized more strongly than misclassification of the majority types. One way to achieve this, is by assigning a weight to each instance. In this chapter, weights w_i are defined per appliance type i corresponding to the imbalance (minority types will get a higher weight) using the following equation:

$$w_i = \frac{n}{a \times \# \text{ samples of type } i}, \quad \forall i \in [1, \ldots, a]$$

where n is the number of samples, and a the number of appliance types. This definition is standard in the Python's sklearn library and is based on [37]. It must be noted that this approach is only valid for classifiers whose error function is dependent on hyperparameters that can be tuned in order to minimize the error. For example, if the k-nearest-neighbors classifier is used with $k = 5$, giving weights to the instances does not impact the outcome as the error function is not dependent on any hyperparameter.

4.6.4 Results

Benchmark dataset For the discussion concerning methods handling imbalanced datasets, the 2017 version of PLAID is used, see Chapter 2 and [11]. The

16×16 pixelated VI image is used as a feature because this was the most promising feature extracted from the study performed on PLAID [6]. Multiple classifiers are used: k-nearest-neighbors (kNN), Gaussian naive Bayes (GNB), logistic regression classifier (LR), support vector machines (SVM), linear discriminant analysis (LDA), decision tree (dTree), random forest (RF), and adaptive boosting (adaBoost). These classifiers were also applied to PLAID in the previous study on PLAID [6].

Evaluation criteria Like done in Section 4.3.3 and 4.4.3, the generalization properties of the classifier are validated using leave-one-house-out cross-validation, as recommended in [6]. Also, the F_{macro} is used to express the performance of the classifier and not the accuracy, as the accuracy should be used with caution in the scenario where some appliances are rarely used [21].

Handling imbalance Table 4.3 shows the relative gain/loss for the different methods when comparing them to the standard approach, when the imbalance is not counteracted. Three things can be noted from the results. First, applying over-, and under-sampling, smote or adapting the error function, does not lead to an improved F_{macro}-measure when applied to PLAID if the pixelated VI image is used as input for the previously mentioned classifiers. Second, the improvement is significant for balanced bootstrapping when used with adaBoost. However, in this case, the standard performance is well below the F_{macro} result of a RF (F_{macro} = 68.62%). Even when handling data imbalance, none of these classifiers outperform the RF method when the pixelated VI image is used as input. Third, when considering the best classifiers kNN, RF, and SVM, none of the methods handling the data imbalance lead to a significant improvement when the pixelated VI image is used as input. This shows that kNN, RF, and SVM are quite robust in learning the appliances types, even when the data is imbalanced. The confusion matrix constructed from the results when RF is applied on the pixelated VI image of PLAID is shown in Figure 4.23. The values in the matrix represent the absolute number of appliance instances detected. The color represents per appliance (per row) the relative number of detected appliances with respect to the total number of that appliance. The air conditioner (AC), fan, and fridge get confused with each other, as well as the heater and the hairdryer. This is due to the fact that the intermixed appliances contain similar electrical components: the AC and fridge are mostly compressors, and both the heater and hairdryer consist of a heating element.

Reduce redundancy From the results in Table 4.3, one can also conclude that the dataset contains redundancy and more measurements were performed than necessary as synthesizing and over-sampling do not offer a performance increase

method	standard	over	under	smote	BB	weighted
kNN	66.39	-1.00	-5.09	-0.47	-2.54	$+0.00$
GNB	46.65	$+0.82$	$+1.64$	$+0.69$	-5.96	$+0.00$
LR	59.27	-2.54	-4.37	-2.92	$+5.84$	$+0.35$
LDA	63.23	$+0.30$	-3.46	-0.16	-2.47	$+0.00$
dTree	58.89	$+2.50$	-1.16	-0.37	$+4.82$	$+0.14$
RF	68.62	-2.88	-2.73	-1.1	-0.44	-1.26
SVM	66.89	-3.51	-2.78	-4.04	-4.43	-3.39
adaBoost	27.97	-2.83	-1.62	-3.32	$+36.04$	$+0.00$

Table 4.3: The gain/loss in the F_{macro}-measure (in %) when handling data imbalance compared to the case when using the standard classifier.

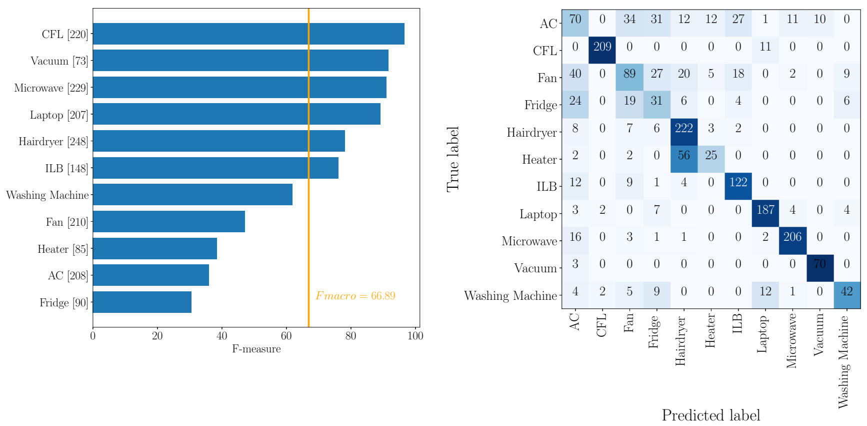


Figure 4.23: The F-measure per appliance and confusion matrix for the 2014 version of PLAID when the RF is used. The number of samples per appliance type is mentioned between the brackets.

method	standard	over	under	smote	BB	weighted
kNN	63.35	$+0.30$	-9.11	$+1.44$	-5.22	$+0.00$
GNB	52.42	-1.03	$+0.23$	-0.71	-6.39	$+0.00$
LR	59.26	-0.11	-3.35	-0.80	$+1.65$	$+0.43$
LDA	60.32	-0.51	-3.92	-0.85	-6.40	$+0.00$
dTree	56.52	-3.66	-10.06	-3.55	$+1.85$	-0.63
RF	63.08	$+0.08$	-7.68	-1.4	-0.77	$+0.14$
SVM	63.01	-0.59	-5.20	-2.10	-1.73	-2.87
adaBoost	19.62	$+2.96$	$+2.19$	$+8.96$	$+25.95$	$+0.00$

Table 4.4: The gain/loss in the F_{macro}-measure (in %) when handling data imbalance compared to the case when using the standard classifier using less training data.

when the pixelated VI image is used as input. To reinforce this statement, the experiments are repeated in the same manner but each training set is reduced such that each appliance is measured only once in each house (so some data remained unused). It is important to note that the test set remained the same. Based on the results in Table 4.4, similar conclusions can be made as above, and the standard F_{macro}-measure for each classifier is about the same as when trained with more data. This corroborates the statement that for appliance identification, more measurements than necessary are present in the PLAID dataset when the pixelated VI image is used as input.

4.7 Conclusion

In this chapter, the steady state behaviour of appliances after the events were characterized using pixelated or weighted pixelated VI images. These were in turn used for classifying the appliance responsible for the event.

The first proposed method uses the contours of the pixelated VI image to characterize appliances in NILM. From these contours, the elliptic Fourier descriptors are calculated and used as input for logistic regression, random forests, and neural networks. The results show that twelve components per sample (i.e., three times four harmonic coefficients) as input for a random forest are sufficient to obtain a prediction accuracy of 78.49% and $F_{\text{macro}} = 66.20\%$ for the 2014 version of PLAID, which is comparable to the results reported in [6]. This leads to significant ($> 20\times$) storage savings when compared to the original VI image comprising $16 \times 16 = 256$ values. If testing is performed on the aggregated data of the 2018 version of PLAID, the $F_{\text{macro}} = 72.47\%$ and $F_{\text{macro}} = 85.31\%$ when training uses respectively submetered and aggregated data. High frequency submetered data is thus not necessary for training.

The second proposed method uses the weighted pixelated images as input for a deep learning method: a CNN that can automatically extract relevant spatial features from the VI trajectories. The method is applied on the 2014 version of PLAID and WHITED resulting in a F_{macro}-measure of respectively 77.60% and 75.46%. The F-measure per appliance shows that the method gives good results for a large number of appliances. The confusion between appliances can be explained by common electrical components or the small number of appliance instances. This method outperforms the previous method and [6] for PLAID, the gain in performance is obtained mainly by the ability to accurately classify the heaters. If testing is performed on the aggregated data of the 2018 version of PLAID, the $F_{\text{macro}} = 80.38\%$ and $F_{\text{macro}} = 87.95\%$ when training uses respectively submetered and aggregated data. Again, this method outperforms the previous method and the same conclusion concerning the redundancy of high frequency submetered data can be made.

Adding the current peak values as an extra feature to the input of the previous methods increases the F_{macro}. However, the highest F_{macro} (96.69% for the 2018 version of PLAID) is obtained when current cycles of steady state behaviour are used as input for a random forest.

Furthermore, it was shown that applying methods handling imbalance on PLAID like over-, and under-sampling, synthesizing samples, balanced bootstrapping and adjusting the error function do not lead to any improvements in terms of the F_{macro}-measure in the scenario where the right classifier is used and when the pixelated VI image is used as input. If a sub-optimal choice of classifier is made, balanced bootstrapping can increase performance. The results also indicate that for appliance identification purposes, more measurements than necessary are present in PLAID. This was confirmed by the fact that results when training with less data (only one measurement per appliance type in a house) are comparable when all the pixelated VI images are used as input.

References

[1] Nilson Henao, Kodjo Agbossou, Sousso Kelouwani, Yves Dubé, and Michaël Fournier. *Approach in nonintrusive type i load monitoring using subtractive clustering*. IEEE Transactions on Smart Grid, 2015.

[2] Warit Wichakool, Zachary Remscrim, Uzoma A Orji, and Steven B Leeb. *Smart metering of variable power loads*. IEEE Transactions on Smart Grid, 6(1):189–198, 2015.

[3] Hsueh-Hsien Chang, Kun-Long Chen, Yuan-Pin Tsai, and Wei-Jen Lee. *A new measurement method for power signatures of nonintrusive demand monitoring and load identification*. IEEE Transactions on Industry Applications, 48(2):764–771, 2012.

[4] Taha Hassan, Fahad Javed, and Naveed Arshad. *An empirical investigation of VI trajectory based load signatures for non-intrusive load monitoring*. IEEE Transactions on Smart Grid, 5(2):870–878, 2014.

[5] Liang Du, Dawei He, Ronald G Harley, and Thomas G Habetler. *Electric load classification by binary voltage–current trajectory mapping*. IEEE Transactions on Smart Grid, 7(1):358–365, 2016.

[6] Jingkun Gao, Emre Can Kara, Suman Giri, and Mario Bergés. *A feasibility study of automated plug-load identification from high-frequency measurements*. In Signal and Information Processing (GlobalSIP), 2015 IEEE Global Conference on, pages 220–224. IEEE, 2015.

[7] Hana Altrabalsi, Vladimir Stankovic, Jing Liao, and Lina Stankovic. *Low-complexity energy disaggregation using appliance load modelling*. AIMS Energy, 4(1):884–905, 2016.

[8] M Nguyen, Sami Alshareef, A Gilani, and Walid G Morsi. *A novel feature extraction and classification algorithm based on power components using single-point monitoring for NILM*. In Electrical and Computer Engineering (CCECE), 2015 IEEE 28th Canadian Conference on, pages 37–40. IEEE, 2015.

[9] Kaustav Basu, Ahmad Hably, Vincent Debusschere, Seddik Bacha, Geert Jan Driven, and Andres Ovalle. *A comparative study of low sampling non intrusive load dis-aggregation*. In Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE, pages 5137–5142. IEEE, 2016.

[10] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. *Imagenet large scale visual recognition challenge*. International Journal of Computer Vision, 115(3):211–252, 2015.

[11] Jingkun Gao, Suman Giri, Emre Can Kara, and Mario Bergés. *PLAID: a public dataset of high-resoultion electrical appliance measurements for load identification research: demo abstract*. In proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings, pages 198–199. ACM, 2014.

[12] Matthias Kahl, Anwar Ul Haq, Thomas Kriechbaumer, and Hans-Arno Jacobsen. *Whited-a worldwide household and industry transient energy data set*. In 3rd International Workshop on Non-Intrusive Load Monitoring, 2016.

[13] A Longjun Wang, B Xiaomin Chen, C Gang Wang, and DD Hua. *Non-intrusive load monitoring algorithm based on features of V–I trajectory*. Electric Power Systems Research, 157:134–144, 2018.

[14] Joseph M Carlo, Marcos S Barbeitos, and Howard R Lasker. *Quantifying complex shapes: elliptical fourier analysis of octocoral sclerites*. The Biological Bulletin, 220(3):224–237, 2011.

[15] Øivind Due Trier, Anil K Jain, and Torfinn Taxt. *Feature extraction methods for character recognition-a survey*. Pattern recognition, 29(4):641–662, 1996.

[16] Mrs Manjiri Deshmukh and Mukta Dhopeshwarkar. *3D face recognition using Contour, Fourier Descriptors*.

[17] William Lorensen and Harvey E. Cline. *Marching Cubes: A High Resolution 3D Surface Construction Algorithm*. Computer Graphics (SIGGRAPH 87 Proceedings), pages 163–170, 1987.

[18] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[19] Frank P Kuhl and Charles R Giardina. *Elliptic Fourier features of a closed contour*. Computer graphics and image processing, 18(3):236–258, 1982.

[20] Jodi Caple, John Byrd, and Carl N Stephan. *Elliptical Fourier analysis: fundamentals, applications, and value for forensic anthropology*. International Journal of Legal Medicine, pages 1–16, 2017.

[21] Stephen Makonin and Fred Popowich. *Nonintrusive load monitoring (NILM) performance evaluation*. Energy Efficiency, 8(4):809–814, 2015.

[22] Nathalie Japkowicz. *Assessment metrics for imbalanced learning*. Wiley Online Library, 2013.

[23] *Nielsen, M., 'Neural Networks and Deep Learning'*. `http://neuralnetworksanddeeplearning.com/`. Accessed: 2017-06-17.

[24] Matthias Kahl, Anwar Ul Haq, Thomas Kriechbaumer, and Hans-Arno Jacobsen. *A Comprehensive Feature Study for Appliance Recognition on High Frequency Energy Data*. In Proceedings of the Eighth International Conference on Future Energy Systems, pages 121–131. ACM, 2017.

[25] Michael Mayo and Sara Omranian. *Towards a new evolutionary subsampling technique for heuristic optimisation of load disaggregators*. In Proc. Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 3–14. Springer, 2016.

[26] Kaustav Basu, Vincent Debusschere, Ahlame Douzal-Chouakria, and Seddik Bacha. *Time series distance-based methods for non-intrusive load monitoring in residential buildings*. Energy and Buildings, 96:109–117, 2015.

[27] Gil Levi and Tal Hassner. *Age and gender classification using convolutional neural networks*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 34–42, 2015.

[28] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. *SMOTE: synthetic minority over-sampling technique*. Journal of artificial intelligence research, 16:321–357, 2002.

[29] Dong Chen, David Irwin, and Prashant Shenoy. *SmartSim: A device-accurate smart home simulator for energy analytics*. In Smart Grid Communications (SmartGridComm), 2016 IEEE International Conference on, pages 686–692. IEEE, 2016.

[30] Nadezda Buneeva and Andreas Reinhardt. *AMBAL: Realistic Load Signature Generation for Load Disaggregation Performance Evaluation*. In Smart Grid Communications (SmartGridComm), 2017 IEEE International Conference on, pages 1–9. IEEE, 2017.

[31] Muhammad Tahir, Josef Kittler, Krystian Mikolajczyk, and Fei Yan. *A multiple expert approach to the class imbalance problem using inverse random under sampling*. Multiple Classifier Systems, pages 82–91, 2009.

[32] Chris Drummond, Robert C Holte, et al. *C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling*. In Workshop on learning from imbalanced datasets II, volume 11. Citeseer Washington DC, 2003.

[33] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. *A systematic study of the class imbalance problem in convolutional neural networks*. arXiv preprint arXiv:1710.05381, 2017.

[34] Byron C Wallace, Kevin Small, Carla E Brodley, and Thomas A Trikalinos. *Class imbalance, redux*. In Proc. 11th International Conference on Data Mining (ICDM), pages 754–763. IEEE, 2011.

[35] Charles Elkan. *The foundations of cost-sensitive learning*. In International joint conference on artificial intelligence, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[36] NV Chawla, N Japkowicz, and A Kotcz. *Editorial: special issue on learning from imbalanced data sets, SIGKDD Explor. Newsl. 6 (1)(2004) 1–6*.

[37] Gary King and Langche Zeng. *Logistic regression in rare events data*. Political analysis, 9(2):137–163, 2001.

5 Unidentified Appliance Detection

"How often have I said to you that when you have eliminated the impossible, whatever remains, however improbable, must be the truth?"
– Sherlock Holmes, *The Adventure of the Copper Beeches*

De Baets, L., Develder, C., Dhaene, T., & Deschrijver, D. "Detection of unidentified appliances in non-intrusive load monitoring using siamese neural networks". submitted to International Journal of Electrical Power & Energy Systems.

5.1 Introduction

The previous chapter proposed supervised methods to classify appliances. Several other supervised and unsupervised methods have been developed to recognise the appliances and to compute the total power consumption [1–3]. However, to our knowledge, nearly all classification algorithms described in the literature are unable to handle unidentified appliances. These will be assigned a label and power consumption that corresponds to the appliance having the most similar features. This chapter suggest a method that is capable of classifying and detecting unidentified appliances, which are labeled as 'unidentified', see Figure 5.1. When such an appliance is detected, the user can be queried for information about the appliance (i.e., the class label). In this chapter, appliances are characterised by their pixelated VI image [4, 5], although other representations can also be considered.

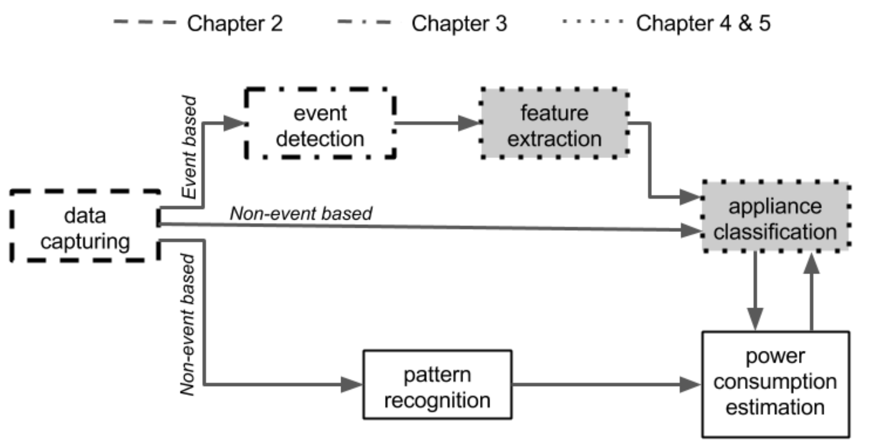


Figure 5.1: A schematic overview of the steps in NILM, situating this chapter.

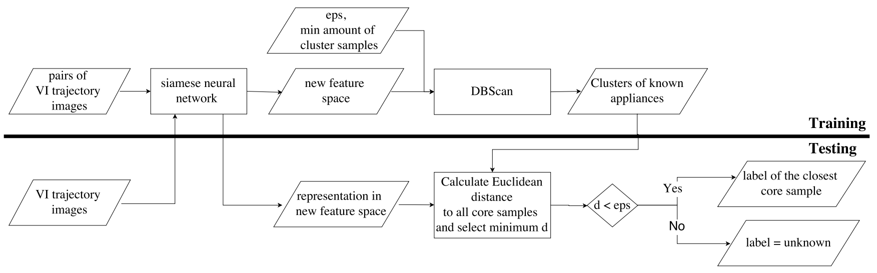

eps, min amount of cluster samples
pairs of VI trajectory images
siamese neural network
new feature space
DBScan
Clusters of known appliances
Training
Testing
VI trajectory images
representation in new feature space
Calculate Euclidean distance to all core samples and select minimum d
d < eps
Yes
No
label of the closest core sample
label = unknown

Figure 5.2: The work flow of the proposed method that is able to detect unidentified appliances.

The proposed method has a training and a test phase, as shown in Figure 5.2. In the training phase, a new, lower dimensional feature space is computed from the pixelated VI images by training a siamese neural network. The pixelated VI images must be paired and labelled respectively as must- or cannot-links, depending on if the images belong to the same appliance type or not. The exact appliance type does not matter. On the transformed input, DBSCAN is performed to group samples with similar feature vectors in the new space. DBSCAN is a state-of-the-art clustering method that does not require prior knowledge about the number of clusters and that is capable of detecting outliers. In the test phase, a pixelated VI image is transformed to the new feature space. If this point does not belong to a cluster, it is labelled as 'unidentified'. If it belongs to a clusters, and labels are available, then it gets the label of the cluster.

The outline of the chapter is as follows: Section 5.2 describes the related work concerning NILM classification algorithms. Section 5.3 explains the concept of siamese neural networks and how they can be used to learn a new feature space. Section 5.4 explains the DBSCAN clustering algorithm. Section 5.5 benchmarks the quality of the clustering, the capability of detecting unidentified appliances and the generalization property of the method. Finally, Section 5.6 concludes this chapter.

5.2 Related work

After measuring the aggregated power consumption and detecting the events using a robust statistical test [6] (see Chapter 3), the events must be characterized, see Figure 5.1. The proposed method describes the transitions of the signals using pixelated VI images that are explained in Section 4.2 of Chapter 4. The next paragraph describes how the state-of-the-art methods that recognize appliances are incapable to cope with unseen appliances. The proposed method poses a solution to detect these unidentified appliances. This solution includes clustering. The last paragraph discusses how clustering is already used in NILM, their low accuracy and their lack of exploiting their ability of detecting unidentified appliances.

Recognizing appliances and monitoring power consumption Once the features are extracted, they can be fed into different classification methods, like support vector machines (SVM) [7, 8], neural networks [9], decision trees [10], or nearest neighbors [11], see Chapter 4. For these methods, labelled training data is necessary. In literature, three groups of unsupervised methods are proposed to circumvent this issue [12]:

- In the first group, unlabelled aggregated training data from the same house is required to build appliance models. Like in [13], where individual appliances are iteratively separated from an aggregate load by tuning prior models of general appliance instances to specific appliance instances using only signatures extracted from the aggregate load. The more training data available, the better the results. However, this approach has not yet been shown to be generalizable across different houses.

- In the second group, submetered data is collected in some houses to build models for appliance classification, which are validated in other (unknown/previously unseen) houses. Like in [14], where a neural network (NN) is constructed for each submetered appliance and used to disaggregate the total power consumption of an unseen house or like in [15], where classifiers are trained on the submetered data of 55 houses and used for classifying data from another house. These methods do not work for unidentified appliances, and require a large number of submetered houses.

- In the third group, no training at all is necessary. In [12], power disaggregation is done using graph signal processing performing adaptive thresholding, signal clustering, and pattern matching. In [16], non-parametric factorial hidden Markov models are used to disaggregate per-load power consumptions from the aggregated signal with minimum prerequisite. The methods in this third group only perform power disaggregation, no appliance names are present.

The majority of the NILM approaches, supervised or unsupervised, are sensitive to appliance changes in the house, and thus require regular re-training. In this chapter, the focus lies on creating a classification algorithm that is able to detect unidentified appliances and is thus resilient against appliance changes in the house. If an unidentified appliance is detected, labeling and retraining is requested. The proposed method fits partly in the second group of the unsupervised methods, as training relies on known houses where submetering is done for building a new appliance feature space and where this space is used to classify appliances in other houses. Moreover, for the proposed method it suffices to know if two appliances belong to the same class or not, independently of their exact label.

Clustering In order to detect unidentified appliances, clustering must be performed. The idea is that samples originating from the same appliances will appear as clusters in the feature space and samples originating from unidentified appliances will appear as outliers indicating the need to create a new cluster. The use of clustering methods has previously been explored in NILM. When Hart [17] described the different steps of the NILM process, a simple clustering algorithm was mentioned where the appliances are grouped using the active - reactive power (P-Q) plane as feature representation space. When an event occurs, the change in the electricity signal is mapped to the plane and a cluster algorithm determines if this two dimensional vector belongs to an existing cluster. If the clusters are labeled, assigning a sample to a cluster leads to classification. Despite its simplicity, this method is incapable of recognizing appliances with overlapping P and Q consumption. In [18], the P-Q plane is also used for genetic k-means and agglomerative clustering. This approach clusters steady-state changes and employs a matching pursuit algorithm to reconstruct the original power signals. This is done using the detected clusters as the sources in a linear blind source separation strategy. However, this method has problems in distinguishing appliances with small P and Q consumptions as their steady-state changes tend to cluster together. In [19], mean-shift clustering is proposed on features that are extracted from the power signal. The resulting clusters are classified into different appliance classes. None of these clustering algorithms exploit their capability to detect unidentified appliances, and none are capable of clustering on appliance-level with high accuracy. The proposed method uses a novel clustering work flow to cope with these two shortcomings. Section 5.3 explains how a higher accuracy can be obtained by learning a new feature space using siamese neural networks. Section 5.4 explains how unidentified appliances can be detected using DBSCAN.

5.3 Siamese neural network

The ability of clustering algorithms to detect small power consuming appliances can be improved by adding more features. However, clustering is sensitive to the curse of dimensionality as it relies on the computation of a distance function like the Euclidean distance. In a high-dimensional case, the differences in distance become less apparent, making the clustering method unusable. For clustering to work, it is thus key to find a low dimensional feature space where the clusters are well separated. To this end, a special kind of NNs called siamese neural networks can be used. Subsection 4.3.3 in Chapter 4 explains NN.

A siamese network consists of two identical NN, meaning that each of them has the same architecture, parameter values and weights. When updating the parameters and weights in one network, the same updates are applied to the other twin network. As input, two feature vectors must be given and as label, a binary value indicating whether or not the feature vectors belong to the same class. The output of the siamese network are two vectors, forming a lower-dimensional representation of the two input vectors. The idea is to learn the representation in such a way, that the distance between these two vectors will be smaller than a given threshold if the two belong to the same class and larger if not. This leads to the use of the so-called contrastive loss function:

$$L(y, d) = \frac{1}{2}\Big(y \times d + (1 - y) \times \max\{m - d, 0\}\Big) \tag{5.1}$$

where y is the binary output, d is the distance between the two input feature vectors, and m is the margin determining when samples are dissimilar: dissimilar input vectors only contribute to the loss function if their distance is smaller than the margin.

Siamese networks are ideally suited to find a relationship between two comparable samples. This is the case in one-shot learning [20], where classification needs to be done with only one example of each class or signature verification [21], where the authenticity of a signature is checked. In this chapter, the siamese neural network is used for dimensionality reduction, like in [22]. This method of dimension reduction is different from classical approaches, such as local linear embedding (LLE) and principal component analysis (PCA), as the siamese neural network learns a function that is capable of consistently mapping unseen samples

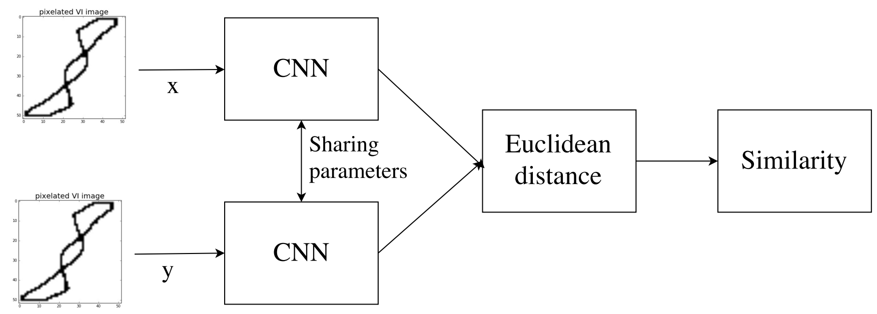


Figure 5.3: The architecture of the siamese network.

to the learned feature space and as the siamese neural network is not constrained by a simple distance function like the Euclidean distance to create the new feature space. This function is one of the trained NN. The resulting representation has the following properties [22]:

1. Simple distance functions, like the Euclidean distance, in the output feature space represent the neighborhood relationships from the input.
2. The learned mapping function needs to be capable to learn invariances to complex transformations.
3. The output for an unseen sample (whose neigborhood relations are unknown) must be reliable.

In this work, the input of the siamese networks consists of pixelated VI images. The architecture of the siamese network is shown in Figure 5.3. For the siamese neural network, the proposed method uses two convolutional neural network (CNN). CNNs are a type of neural networks (NNs) that are often used in computer vision because they are highly suitable to classify images [23]. For an explanation, see Section 4.4 in Chapter 4. The used CNN is depicted in Figure 5.4 and takes as input an $n \times n$ pixelated VI image. This is transformed by a convolutional layer which uses 20 filters each considering regions of 5×5 pixels. After the convolutional layer, there is a pooling layer with a sliding window of 2×2. This combination of a convolutional layer followed by a pooling layer is repeated, and finally, a dense layer is added with n_{out} nodes. The margin m used in the loss function is set to 50. Changing this value does not significantly influence the results.

After the training phase, the siamese neural network can be used to calculate a n_{out}-dimensional representation of new VI binary images. These are obtained by using the output of just one trained CNN.

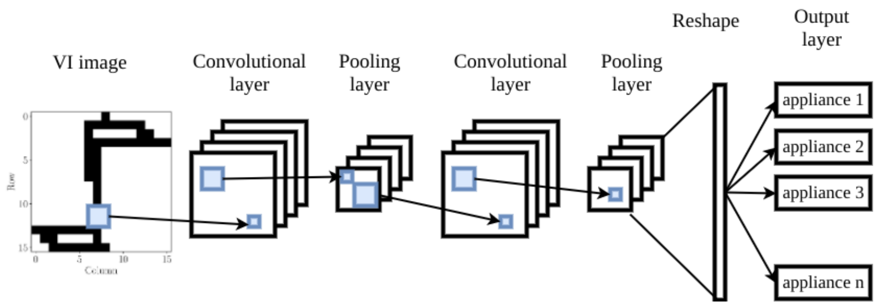


Figure 5.4: The architecture of the cNN that is used in the siamese network.

5.4 DBSCAN

After learning the feature space, unidentified appliances can be detected by performing clustering. Namely, if a new sample is too distant from present clusters (representing known appliances), then it is considered as unidentified.

Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm [24] that will partition a given set of points into non-overlapping clusters as well as outliers. It is a density-based clustering algorithm: points forming a cluster will be close together, whereas the outliers will only have relatively far away neighbors. The algorithm starts with picking one random sample. If not enough close by neighbours are present, the point will be labeled as an outlier and the process continues by selecting a new sample. If there is a sufficient number of close by neighbours, they are all added to the same cluster. The algorithm continues by iterating over all new added points, if these have sufficient close by neighbours, these are also added to the same cluster. This continues until no more samples are added to this cluster. Then a new unvisited random sample is selected and the process is iterated until all points belong to a cluster or are labeled as outliers (noise). Three types of samples can be distinguished:

1. core samples, which are sufficiently close to other samples,
2. non-core samples, which are close by to some core samples but are insufficiently close to other samples (these form border),
3. noisy samples, which have not sufficient close by samples.

Three elements needs to be defined for DBSCAN: (1) the number of sufficient close by points, mintPts, (2) the distance function, and (3) the maximal distance to a close by sample, ϵ. These last two points define if a sample is close by or not.

The advantages of DBSCAN are that the number of clusters does not need to be specified by the user (unlike, e.g., for K-means clustering), clusters can be of any shape (not just circular ones), and outliers are not forced to belong to a cluster but are identified as such. The algorithm is also robust against an imbalance in the occurrence of samples from different clusters. DBSCAN is one of the most common clustering algorithms and was awarded the test of time award at the leading data mining conference, KDD [25].

In this chapter, the transformed input samples are clustered with DBSCAN. As the learned feature space has the property that simple distance functions represent the neighborhood relationships from the original input, the Euclidean distance is used. The parameters mintPts and ϵ are not trained but heuristically set to respectively 5 and 0.2.

To determine which cluster a new sample belongs to (if any), its feature vector is first transformed to the calculated lower-dimensional space. Next, the Euclidean distance is calculated to all core samples and the minimal distance is selected. If this distance is smaller than threshold ϵ, then the sample belongs to the same cluster as the closest core sample. Otherwise, it will be assigned the label 'unidentified', see Figure 5.2.

5.5 Results

To benchmark the described method, several checks must be performed. First, it must be examined if the learned feature space separates the different classes well. Second, the capability of detecting unidentified appliances is tested by using data of an unidentified appliance as test data. Lastly, the generalization property of the method is checked by using data from other (unseen) houses as test data.

Benchmark dataset The performance of the proposed algorithm is validated on the 2014 version of PLAID, see Chapter 2 and [15]. PLAID is a public dataset including sub-metered current and voltage measurements sampled at 30 kHz for 11 different appliance types. More than 200 individual appliances are available, captured in 55 households. For each appliance, at least 5 start-up events are measured, resulting in a total of 1074 measurements.

Scenario 1 To determine if the learned feature space from the siamese neural network separates the classes well, the rand index (RI) of the clusters found by the DBSCAN algorithm is calculated. The RI is a measure of similarity between two

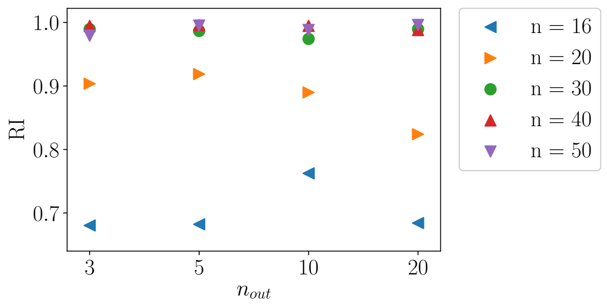


Figure 5.5: The rand index when all data is used to learn the feature space with dimension n_{out}

data clusterings X and Y:

$$R = \frac{a+b}{a+b+c+d} \tag{5.2}$$

where a and b are respectively the number of pairs of elements that are in the same/different cluster(s) in both clusterings X and Y, and c and d are respectively the number of pairs that are in the same cluster for X/Y, but in a different one for Y/X. Higher values of R (max. value 1) indicate a better match of clusterings.

Figure 5.5 shows the RI for different parameter configurations of the siamese neural network that learns the representation from pixelated VI images. The input size of the pixelated VI image ($n \times n$) and the dimension n_{out} of the learned representation are altered. For this, all data samples of the PLAID dataset are fed into the siamese neural network to calculate the mapping function. Increasing n_{out} (for fixed values of n) has little impact on the RI values. In contrast, changes in the value of n (for fixed values of n_{out}) have a strong impact. The best RI values are obtained for $n \geq 30$ with a maximum of 0.996. The high RI values confirm the capability of the siamese neural network to learn a feature space where the clusters are well separated and confirm the ability of the DBSCAN algorithm to find these clusters. Figure 5.6 shows an example of the learned three dimensional feature space (for the ease of visualization) and the corresponding clustering.

Scenario 2 To define how well the method can identify unidentified appliances, training is done on 10 appliances and testing on 1 hold out appliance. It is validated whether (1) the 10 selected appliances are properly separated in individual clusters, and (2) the 11th appliance has its data points classified as ‘unidentified’. The first criterion is validated by calculating the RI, like done above, and the second

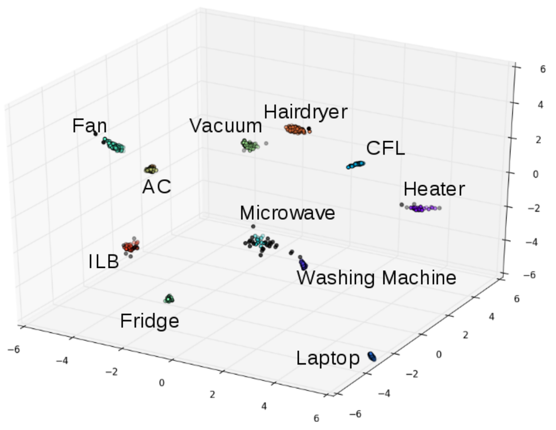


Figure 5.6: The clusters found by DBSCAN in the learned feature space.

criterion by calculating the accuracy of the 'unidentified' labels:

$$\text{Accuracy} = \frac{n_{11}^{un}}{n_{11}} \tag{5.3}$$

with n_{11} the number of samples from the 11th appliance, and n_{11}^{un} the number thereof that is labeled as 'unidentified'. This procedure is performed in a leave-one-appliance-out cross-validation. As the PLAID dataset contains 11 appliances, the training and testing is repeated 11 times, resulting in 11 accuracy values. To obtain the final test result, these values are averaged.

Figure 5.7a and 5.7b display the RI values and accuracy for different n_{out} and n values. Increasing n_{out} and keeping n fixed does not change the values much. This in contrast to changing n and keeping n_{out} fixed, which has a bigger influence. The best RI values and accuracy are obtained for $n \geq 30$, the maximum is respectively 0.994 and 86.67%. The high RI values confirm the capability of the siamese neural network to learn a feature space where the 10 clusters are well separated and the high accuracies confirm the capability to detect new (unidentified) appliances.

Figure 5.8 shows an example of the learned three dimensional feature space when using ten appliances ($n_{out} = 3$ for the sake of easy plotting), the corresponding ten clusters, and the mapping of the unseen samples of the test set. In this example, the samples in the test set originate from a microwave. To know which appliances are mixed up, a confusion matrix is created: Figure 5.9 reports for each appliance type (row index) the number of labels that were correctly predicted as 'unidentified' or confused with other appliances (column index). The

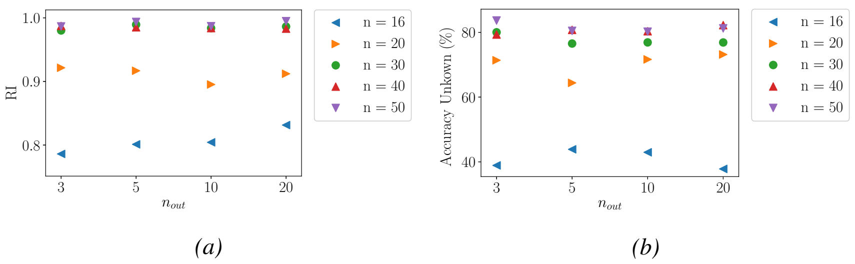


Figure 5.7: The RI values and accuracy for different n_{out} and n values when leave-one-appliance-out cross-validation is used.

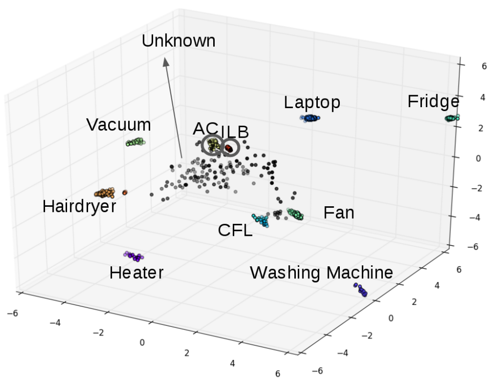
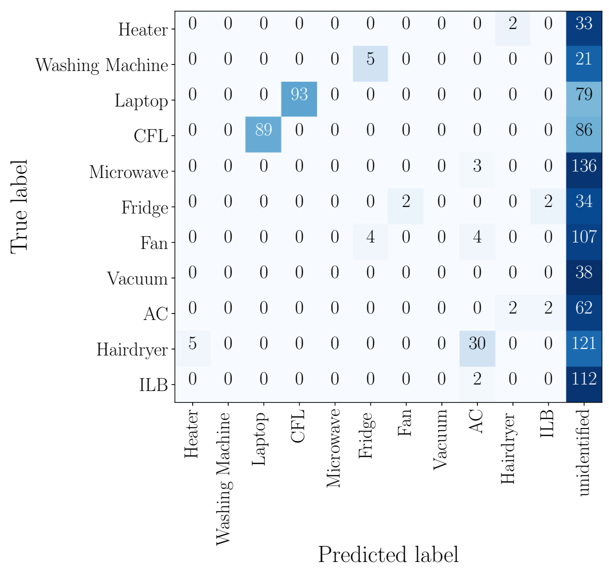


Figure 5.8: The 10 *clusters and unidentified points found by DBSCAN in the learned feature space representing respectively the* 10 *seen appliances and the* 1 *hold out appliance.*

values in the matrix are absolute and the colors represent the relative value per row (thus per appliance). It can be seen that if the laptop is not used for training it is put in the cluster containing the compact fluorescent lamp (CFL) examples and the other way around.

Scenario 3 To test whether the method generalizes well, training and testing is performed on different houses. As recommended in [4], leave-one-house-out cross-validation is used. For the PLAID dataset, this means that 54 houses are used for the training set and 1 for test set. The method works if (1) the learned feature space using the 54 houses separates the appliance clusters sufficiently, and (2) the appliances of the test set are projected on the correct cluster. The first criterion is validated by calculating the RI and the second by counting the number of test samples that are assigned to the correct cluster, to a wrong cluster or get the label unidentified. Three accuracy measures are defined: (1) the positive rate defined as the percentage of samples assigned to the correct cluster, (2) the negative rate



True label \ Predicted label	Heater	Washing Machine	Laptop	CFL	Microwave	Fridge	Fan	Vacuum	AC	Hairdryer	ILB	unidentified
Heater	0	0	0	0	0	0	0	0	0	2	0	33
Washing Machine	0	0	0	0	0	5	0	0	0	0	0	21
Laptop	0	0	0	93	0	0	0	0	0	0	0	79
CFL	0	0	89	0	0	0	0	0	0	0	0	86
Microwave	0	0	0	0	0	0	0	0	3	0	0	136
Fridge	0	0	0	0	0	0	2	0	0	0	2	34
Fan	0	0	0	0	0	4	0	0	4	0	0	107
Vacuum	0	0	0	0	0	0	0	0	0	0	0	38
AC	0	0	0	0	0	0	0	0	0	2	2	62
Hairdryer	5	0	0	0	0	0	0	0	30	0	0	121
ILB	0	0	0	0	0	0	0	0	2	0	0	112

Figure 5.9: The confusion matrix when leave-one-appliance-out cross-validation is used.

defined as the percentage of samples assigned to the wrong cluster, and (3) the unidentified rate defined as the percentage of samples labelled as 'unidentified'. These quantities add up to 100%.

To be able to calculate the first and second rates, the appliances labeling provided in the PLAID dataset is used. It must be noted that these labels are only used to validate the model predictions: they are not used for training the method. As leave-one-house-out cross-validation is performed on 55 houses, there are 55 test scores (one for each house). To obtain the final test result, these values are averaged.

Figure 5.10a shows the RI values for different n_{out} and n values. It can be concluded that the clusters are separated sufficiently for higher n values. Changing n_{out} does not influence the result. Figure 5.10b, 5.10c and 5.10d show the three accuracies defining how much test samples are respectively assigned to the correct, incorrect or no cluster. Again, changing n_{out} does not change the results significantly. When using $n = 16$, a large part of the test samples ($> 50\%$) are assigned to the correct cluster, but also a significant part of them ($\backsim 22.5\%$) are classified incorrectly. When using a larger $n = 50$, the number of correctly assigned test samples is much smaller, namely around 38.7%, but so are the incorrectly assigned samples ($\sim 1.3\%$). As a consequence, the number of samples

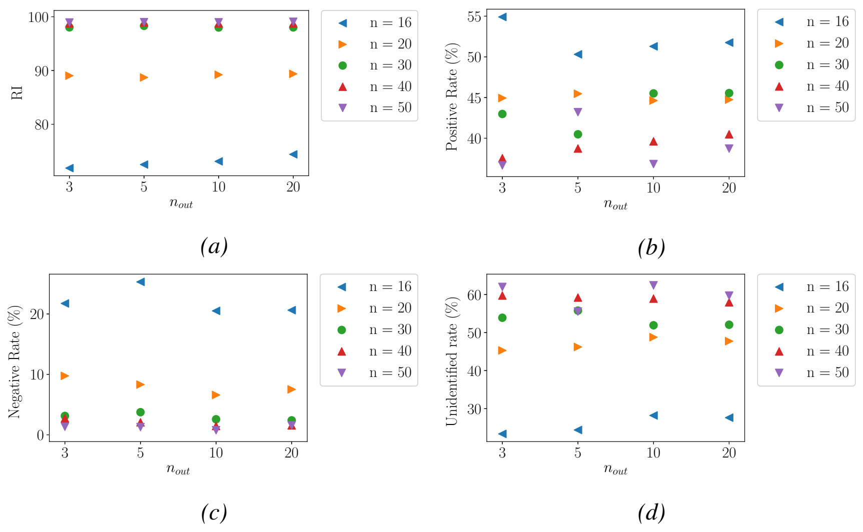
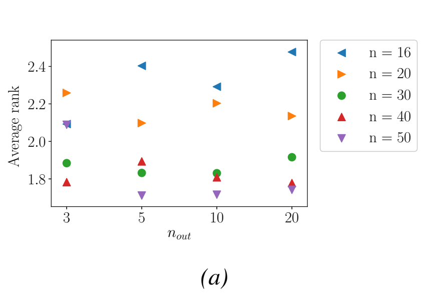


Figure 5.10: The RI values, and the positive, negative and unidentified rate for different n_{out} and n values when leave-one-house-out cross-validation is used.

labeled as unidentified is larger (around 60%). The method avoids the risk of being too certain. Most likely, this is caused by test samples lying just outside the cluster, but still in its proximity. This can be checked by calculating the rank of the correct cluster for each test sample labeled as unidentified. First, all clusters are ranked by calculating the distance d_{ij} from the given sample i to each of the clusters j, and sorting these distances in ascending order. Ideally, the test sample's appliance cluster should be on the first position. Figure 5.11a shows the rank that is averaged over the 55 folds. For all the different parameter combinations, this result shows that the correct cluster is the closest or second closest cluster. Figure 5.11b shows the box plot of the rank of the correct cluster for all the test samples labeled as 'unidentified' in all the test houses (1074 ranks) when $n = 50$. As these box plots show, their is very little variance on the rank of the correct cluster, making the average a valid measure. This result is important because it implies that if the method labels a sample as unidentified, it can query the user to confirm if it is a new appliances or not, while immediately suggesting a label (e.g., listing the top-3 of the ranked list). If the sample originates from an existing appliance, the correct appliance (cluster) will be in this top.

5.6 Conclusion

This chapter presents a novel method for appliance classification and detection of unidentified appliances in non-intrusive load monitoring. Both rely on a learned



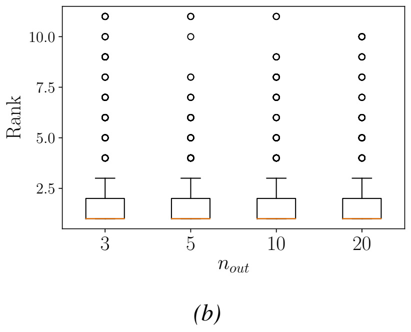


Figure 5.11: (a) The average rank of the correct cluster for the samples labeled as 'unidentified' for different n_{out} and n values when using leave-one-house-out cross-validation. (b) The box plots of the rank of the correct cluster for all samples labeled as 'unidentified' for $n = 50$ and different n_{out} values.

vector representation function (a trained CNN), which takes as input a pixelated VI image. To learn this representation, training data in the form of such VI images, in labeled pairs of the same/different appliance types is needed. A siamese neural network is trained on these pairs outputting a pair of lower-dimensional vector representations, such that the distance between these two vectors is lower or higher than a threshold for respectively same/different appliance. In this newly learned feature space, DBSCAN is performed, allowing us to assign test samples to clusters or label them as unidentified. Benchmarking on the PLAID dataset shows that 87% of the unknown appliances are labelled as unidentified. Furthermore, if unseen instances of known appliance types are given as input 39% is classified correctly, 1% incorrectly and 60% as unidentified. However for the appliances classified as unidentified, good suggestions can be made concerning the cluster it belongs to as the correct cluster is on average the 1.75 closest cluster.

Future work includes the incorporation of domain knowledge into the error function. For example, appliances belonging to the same type (resistive, reactive or capacitive) may correspond to clusters that are nearby in the feature space.

References

[1] Michael Zeifman and Kurt Roth. *Nonintrusive appliance load monitoring: Review and outlook*. IEEE transactions on Consumer Electronics, 57(1), 2011.

[2] Ahmed Zoha, Alexander Gluhak, Muhammad Ali Imran, and Sutharshan Rajasegarar. *Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey*. Sensors, 12(12):16838–16866, 2012.

[3] Anthony Faustine, Nerey Henry Mvungi, Shubi Kaijage, and Kisangiri Michael. *A Survey on Non-Intrusive Load Monitoring Methodies and Techniques for Energy Disaggregation Problem*. arXiv preprint arXiv:1703.00785, 2017.

[4] Jingkun Gao, Emre Can Kara, Suman Giri, and Mario Bergés. *A feasibility study of automated plug-load identification from high-frequency measurements*. In Signal and Information Processing (GlobalSIP), 2015 IEEE Global Conference on, pages 220–224. IEEE, 2015.

[5] Liang Du, Dawei He, Ronald G Harley, and Thomas G Habetler. *Electric load classification by binary voltage–current trajectory mapping*. IEEE Transactions on Smart Grid, 7(1):358–365, 2016.

[6] Leen De Baets, Joeri Ruyssinck, Chris Develder, Tom Dhaene, and Dirk Deschrijver. *On the Bayesian optimization and robustness of event detection methods in NILM*. Energy and Buildings, 145:57–66, 2017.

[7] Hana Altrabalsi, Vladimir Stankovic, Jing Liao, and Lina Stankovic. *Low-complexity energy disaggregation using appliance load modelling*. AIMS Energy, 4(1):884–905, 2016.

[8] Hana Altrabalsi, Jing Liao, Lina Stankovic, and Vladimir Stankovic. *A low-complexity energy disaggregation method: Performance and robustness*. In Computational Intelligence Applications in Smart Grid (CIASG), 2014 IEEE Symposium on, pages 1–8. IEEE, 2014.

[9] Chaoyun Zhang, Mingjun Zhong, Zongzuo Wang, Nigel Goddard, and Charles Sutton. *Sequence-to-point learning with neural networks for non-intrusive load monitoring*. arXiv preprint arXiv:1612.09106, 2016.

[10] M Nguyen, Sami Alshareef, A Gilani, and Walid G Morsi. *A novel feature extraction and classification algorithm based on power components using single-point monitoring for NILM*. In Electrical and Computer Engineering (CCECE), 2015 IEEE 28th Canadian Conference on, pages 37–40. IEEE, 2015.

[11] Kaustav Basu, Ahmad Hably, Vincent Debusschere, Seddik Bacha, Geert Jan Driven, and Andres Ovalle. *A comparative study of low sampling non intrusive load dis-aggregation*. In Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE, pages 5137–5142. IEEE, 2016.

[12] Bochao Zhao, Lina Stankovic, and Vladimir Stankovic. *On a training-less solution for non-intrusive appliance load monitoring using graph signal processing*. IEEE Access, 4:1784–1799, 2016.

[13] Oliver Parson, Siddhartha Ghosh, Mark Weal, and Alex Rogers. *Non-Intrusive Load Monitoring Using Prior Models of General Appliance Types.* In AAAi, 2012.

[14] Jack Kelly and William 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, pages 55–64. ACM, 2015.

[15] Jingkun Gao, Suman Giri, Emre Can Kara, and Mario Bergés. *PLAID: a public dataset of high-resoultion electrical appliance measurements for load identification research: demo abstract*. In proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings, pages 198–199. ACM, 2014.

[16] Ruoxi Jia, Yang Gao, and Costas J Spanos. *A fully unsupervised non-intrusive load monitoring framework*. In Smart Grid Communications (SmartGridComm), 2015 IEEE International Conference on, pages 872–878. IEEE, 2015.

[17] George William Hart. *Nonintrusive appliance load monitoring*. Proceedings of the IEEE, 80(12):1870–1891, 1992.

[18] Hugo Goncalves, Adrian Ocneanu, Mario Berges, and RH Fan. *Unsupervised disaggregation of appliances using aggregated consumption data*. In The 1st KDD Workshop on Data Mining Applications in Sustainability (SustKDD), 2011.

[19] Zhenyu Wang and Guilin Zheng. *Residential appliances identification and monitoring by a nonintrusive method*. IEEE transactions on Smart Grid, 3(1):80–92, 2012.

[20] Gregory Koch. *Siamese neural networks for one-shot image recognition*. PhD thesis, University of Toronto, 2015.

[21] Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. *Signature Verification Using A 'Siamese' Time Delay Neural Network*. IJPRAI, 7(4):669–688, 1993.

[22] Raia Hadsell, Sumit Chopra, and Yann LeCun. *Dimensionality reduction by learning an invariant mapping*. In Computer vision and pattern recognition, 2006 IEEE computer society conference on, volume 2, pages 1735–1742. IEEE, 2006.

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. *Imagenet large scale visual recognition challenge*. International Journal of Computer Vision, 115(3):211–252, 2015.

[24] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. *A density-based algorithm for discovering clusters in large spatial databases with noise.* In KDD, volume 96, pages 226–231, 1996.

[25] *2014 SIGKDD TEST OF TIME AWARD*. `http://www.kdd.org/News/view/2014-sigkdd-test-of-time-award`. Accessed: 2017-04-10.

6 Conclusion

"Education never ends, Watson. It is a series of lessons, with the greatest for the last."
– Sherlock Holmes, *His Last Bow: 8 Stories*

6.1 Summary

In this thesis, we cover the different steps comprising event-based NILM, see Figure 6.1, answering the different research questions posed in Chapter 1:

- **Can data be collected in a consistent way, and in addition be easily extendable?**

  To answer this question, we extended PLAID with submetered and aggregated voltage and current measurements from different household appliances sampled at 30 kHz, making it the first dataset to contain both submetered and aggregated data at high frequency. Furthermore, it is collected in a consistent way: there are no calibration errors and no labels are missing. A systematic description of the measurement set-up and dataset is given, providing the means to consistently extend this dataset is presented.

- **How can we develop event detection methods that are robust against parameter changes?**

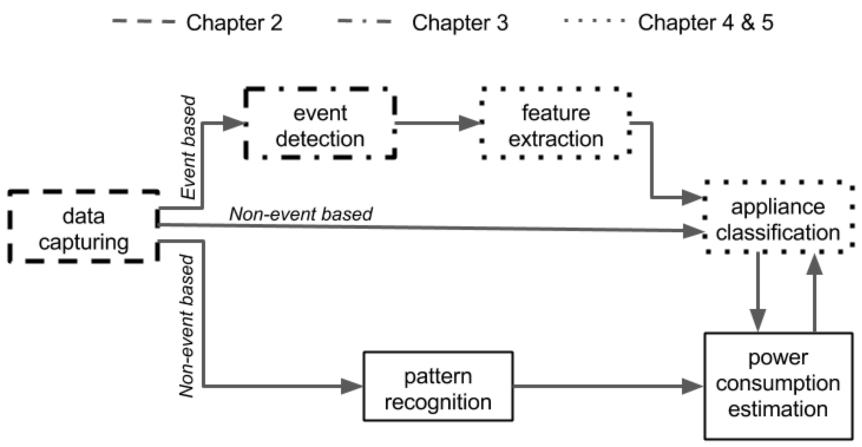


Figure 6.1: A schematic overview of the steps in NILM.

Two event detection methods, the voting χ^2 GOF and cepstrum method, are presented in Chapter 3 that are robust against base load differences. This in contrast to the standard χ^2 GOF method. For example, when a base load of 3000 W (which corresponds to the power consumption of two typical electrical heaters) is added to the power signal, compared to the standard χ^2 GOF method, the voting χ^2 GOF method leads to a performance increase of $7 - 12\%$ in terms of F-measure, while Cepstrum reaches $7 - 15\%$ larger F-measure values.

Other parameters were optimized in an efficient way by presenting surrogate-based optimization, resulting in a speed up in computation time with a factor up to 100000 when compared to the standard brute force method.

- **Which features are best to characterize the appliances?**

  We tried three different features. First, the contours of the pixelated VI image to characterize appliances in NILM are used as input for logistic regression, random forests, and neural networks. The results show that twelve components per sample (i.e., three times four harmonic coefficients) are sufficient to obtain a prediction accuracy of 78.49% and $F_{\text{macro}} = 66.20\%$ for the 2014 version of PLAID, which is comparable to the results reported in [1]. This leads to significant ($> 20\times$) storage savings when compared to the original VI image comprising $16 \times 16 = 256$ values.

  Second, weighted pixelated images are used as input for a CNN. The method is applied on the 2014 version of PLAID and WHITED dataset resulting in a F_{macro}-measure of respectively 77.60% and 75.46%. The F-measure per appliance shows that the method gives good results for a large number of appliances. The confusion between appliances can be explained by common electrical components or the small number of appliance instances. This method outperforms the previous method and [1] for PLAID, the gain in performance is obtained mainly by the ability to accurately classify the heaters.

Third, the current peak values are used as a feature. When added as an extra feature to previous methods, the F_{macro} increases. However, the highest F_{macro} (96.69% for the 2018 version of PLAID) is obtained when current cycles of steady state behaviour are used as input for a random forest.

However, as the performance is not perfect, the question remains which features are the best to characterize the appliances.

- **How does the appliance imbalance influence the performance of the classifier and do approaches that address the imbalance improve the performance?**

  In chapter 4, it is demonstrated that applying methods to deal with this imbalance on PLAID does not lead to any improvements in terms of the F_{macro}-measure in the scenario where the right classifier is used and the pixelated VI image is used as input. For the 2017 version of PLAID, the right classifier is a random forest, kNN or SVM. If a suboptimal choice of classifier is made, like adaBoost, dTree, or logistic regression, balanced bootstrapping can increase performance.

  An additional study on other datasets is necesarry before general conclustions can be made.

- **How can a feature space be learned in such a way that different appliance types form different clusters?**

  In order to create clearly separated clusters, we learn a new feature space using siamese neural networks. For this we need training data in the form of weighted pixelated images, in labelled pairs of same/different appliance types. Training makes sure that instances of same/different appliance types lie closer/farther from each other.

  In this newly learned feature space, the density based clustering method DBSCAN is performed, allowing us to assign test samples to clusters or label them as unidentified. Benchmarking on the 2014 version of PLAID shows that 87% of the unknown appliances are labelled as unidentified. Furthermore, if unseen instances of known appliance types are given as input 39% is classified correctly, 1% incorrectly and 60% as unidentified. For the appliances classified as unidentified, good suggestions can be made concerning the correct label, because when ranking the clusters on their closeness to the sample, the cluster having the correct label has on average rank 1.75.

In the course of the research, additional questions were answered:

- **Is high frequency submetered data necessary?**

  If the EFD's of the countours of pixelated VI images are used as input for random forest and testing is performed on the aggregated data of the 2018 version of PLAID, the $F_{\text{macro}} = 72.47\%$ and $F_{\text{macro}} = 85.31\%$ when training uses respectively submetered and aggregated data. If weighted VI images are used as input for a CNN and testing is performed on the aggregated data of the 2018 version of PLAID, the $F_{\text{macro}} = 80.38\%$ and $F_{\text{macro}} = 87.95\%$ when training uses respectively submetered and aggregated data. For the aggregated data of the 2018 version of PLAID, it holds that high frequency submetered data is not necessary for training.

- **Are there sufficient different samples available in public datastes?**

  We found that there is an overabundance of submetered measurements within the 2014 version of PLAID for the task of identifying appliances. It would be interesting to investigating if this the case for all datasets, and thus if classifiers can perform as good if there are less measurements available.

In summary, the work presented in this thesis comprises robust and efficient methods for each of the steps involved in event-based NILM.

6.2 Future work

Future work can develop along two lines: the accuracy of the NILM can be improved and/or new use cases using NILM data can be developed.

6.2.1 Improving the accuracy

The accuracy of appliance classification or power consumption estimation is still not perfect. One possible explanation for this is that event based methods excel in detecting single state appliances, think a lamp, but get confused when detecting program-based appliances such as washing machine, where several events belong to one appliance. For non-event based methods, it is the other way around, they excel at detecting program based appliances but get confused by the unpredictable active time of single state appliances. Running these methods in parallel would be beneficial. The biggest unsolved difficulty for this approach will be to decide which method to believe at what moment.

Improving the accuracy of NILM for industrial buildings remains mainly unadressed. Most NILM algorithms have been developed for residential households. This is mostly due to the high availability of NILM data sets pertaining to these households. However, energy consumption of commercial, public, and industrial buildings (think schools, hospital, stores, offices, etc.) is also significant making it an interesting use case. However, these settings prove more challenging, as assumptions with regard to the temporal dependencies between appliance usage, and the activation of only one appliance at at time, no longer hold [2]. Some preliminary results have been reported in the literature [3].

Another reason for the imperfect accuracy is the inability to detect variable power draw appliances correctly. Variable power appliances (VPA) prove more difficult to monitor as they have a continuous range of power consumption values while operating [4]. Since VPA appliances do not exhibit a clear event-based signature, many methods for load disaggregate become inadequate due to their reliance on the identification of steady states and/or switching events of individual appliances. Nevertheless, VPA appliances are abundant within households (e.g., a TV, desktop computer, HVAC installations). If these devices cannot be properly disambiguated from other appliances, a portion of the power remains unassigned, lowering the obtained accuracy [5]. If ultimately a perfect accuracy is desired, research on how to detect these VPA appliances is required.

6.2.2 Developing new use cases

Using the data obtained from NILM, new use cases can be developed. For example from the active appliances, one could determine the occupancy of a house and several energy-efficiency optimizations could be performed, such as automatically turning appliances on/off, gaining actionable insights about slumber power consumption or burglary detection. Determining the occupancy of a house has been analyzed in [6]. However, only two houses were taken into account and events were not always available (for example when only 1 Hz or lower frequency data is available). The authors also explicitly assumed the home owners were asleep between 1 and 4 am. Research needs to be conducted to check whether the house occupancy can be determined by the active appliances and if the energy-efficiency optimizations are practical and meaningful.

Using the active appliance, one could also construct behavioural profiles in terms of energy consumption. This can be useful to compare different users in order to detect abnormal or illegal activity. This kind of information would not only benefit the end user, but could also be valuable for third parties, e.g., insurance companies could charge a flexible premium based on your profile. Research must be performed to validate the idea that illegal or strange behaviour can be detected in energy consumption having as challenge to obtain the relevant data.

Using the active time and estimated energy consumption of a fridge or freezer obtained from NILM, we can analyze the performance and age of these appliances. This knowledge can result in several actionable insights, such as defrosting is necessary, or the device is no longer energy efficient and needs to be replaced. Knowing that a large proportion of total energy within households is consumed by fridges and freezers, this can lead to large savings. The same approach can also be adopted for less frequently used appliances such as water and heat pumps, or battery chargers for electric cars. The biggest research challenge for this use case is the gathering of data of older/newer, badly/well maintained fridges or freezers.

Combining NILM data with various internet-of-things (IoT) sensors such as gas, water, and temperature meters, or all round smart thermostats creates a wealth of new use cases, e.g.:

- besides reporting energy consumption of wet appliances, we can also obtain the corresponding water consumption,
- detecting the presence in the home by using the gas consumption,
- generation of alarm messages, for example when high water use in the absence of active wet appliances indicate a leak.

Research needs to be conducted to find out, a.o., if the addition of water consumption is really informative and if high water use consumption can not be caused by other usages like opening a kitchen faucet.

References

[1] Jingkun Gao, Emre Can Kara, Suman Giri, and Mario Bergés. *A feasibility study of automated plug-load identification from high-frequency measurements*. In Signal and Information Processing (GlobalSIP), 2015 IEEE Global Conference on, pages 220–224. IEEE, 2015.

[2] Nipun Batra, Oliver Parson, Mario Berges, Amarjeet Singh, and Alex Rogers. *A comparison of non-intrusive load monitoring methods for commercial and residential buildings*. arXiv preprint arXiv:1408.6595, 2014.

[3] Emil Holmegaard and Mikkel Baun Kjærgaard. *NILM in an Industrial Setting: A Load Characterization and Algorithm Evaluation*. In Proceedings of the 2nd IEEE International Conference on Smart Computing (SMARTCOMP), pages 1–8, 2016.

[4] Hung-Yuan Chen, Yao-Chung Fan, Chien-Liang Lai, and Huan Chen. *Identifying Variable-Power Appliances in Non-intrusive Load Monitoring Systems*. In Proceedings of the 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pages 452–457, 2016.

[5] Warit Wichakool, Zachary Remscrim, Uzoma A Orji, and Steven B Leeb. *Smart metering of variable power loads*. IEEE Transactions on Smart Grid, 6(1):189–198, 2015.

[6] Dong Chen, Sean Barker, Adarsh Subbaswamy, David Irwin, and Prashant Shenoy. *Non-intrusive occupancy monitoring using smart meters*. In Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings, pages 1–8, 2013.