Demand Response Management Using Machine Learning Methods

Lastverschiebung mit Hilfe von maschinellen Lernverfahren

Der Technischen Fakultät
der Friedrich-Alexander-Universität
Erlangen-Nürnberg

zur

Erlangung des Doktorgrades Dr.-Ing.

vorgelegt von

Marc Wenninger
aus
Stuttgart

Als Dissertation genehmigt
von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen Prüfung:	16.02.2024
Gutachter:	Prof. Dr.-Ing. habil. Andreas Maier
	Prof. Dr.-Ing. Jochen Schmidt
	Prof. Dr.-Ing. Marco Pruckner

Abstract

The worldwide transformation of electricity production from fossil and nuclear energy sources to renewable energy sources is accompanied by many challenges. One of those challenges is finding an equilibrium of supply and demand – an important balance for the stability of electric grids. Production and consumption are kept in balance by adapting electricity production to consumption. Most renewable energy sources do not produce energy when demanded, but when natural conditions are suitable. As energy cannot yet be stored efficiently, over-production is as much of a problem as underproduction. Demand Response (DR) is the means for end-users to contribute to the balancing challenge. Providing the end-users with an incentive such as time-based pricing that changes according to the supply will encourage users to contribute to the equilibrium. Users' contribution usually has implications for their daily habits and can be associated with discomfort, thus it requires a high level of involvement. Lowering the required involvement is therefore seen as an important step toward an acceptance of time-based pricing. Since the 1980s, machine learning has been seen as a solution to lower the barrier for private households to participate. The idea is to provide households with information about their electricity consumption, make recommendations on behavior changes or take automated actions. Such information can be retrieved from monitoring a household's electricity consumption.

This thesis contributes to the process of extracting information and knowledge from monitored electricity consumption in private households using machine learning. An overview of data sources and general approaches is provided. Based on this research, the Machine Learning Demand Response Model (MLDR) is introduced, defining the relation between data, knowledge, and actions. This model enhances the understanding of the individual steps required to transform monitored electricity consumption data into individual recommendations or automated actions. These steps are: data monitoring, appliance identification, appliance usage segmentation, and appliance usage prediction. For each of these steps, this thesis provides an overview of the current research state and introduces new approaches.

A new monitoring system for both individual appliances and household mains is introduced. The system was used to collect a scientific dataset called Domestic Energy Demand Dataset of Individual Appliances in Germany (DEDDIAG). It contains measurements of 50 individual appliances located in 15 homes, recorded with a sample rate of 1 Hz over a period of up to 3.5 years. The dataset has been enriched with manual appliance usage annotations as well as demographic data describing the household. The system, as well as the dataset, has been published under an open-source license.

Based on this dataset, an appliance category identification algorithm is introduced. The algorithm extracts features using a wavelet transformation and classifies data using the k-Nearest-Neighbor (kNN) classifier. It was evaluated and published as a challenge baseline for DEDDIAG. Next to this approach that relies on low sample rates, a high sample rate algorithm is introduced. The algorithm is based on transforming one voltage-current cycle, known as the voltage-current (V-I) trajectory, into two separate Recurrence Plots (RPs) which are then classified using a Convolutional Neural Network (CNN) in combination with Spacial Pyramid Pooling (SPP). The algorithm is evaluated on three different datasets and compared to previously proposed algorithms.

Finding the start and stop of an appliance is the basis for deriving usage patterns. This appliance event segmentation has received little attention from other researchers, and the most commonly used algorithm, a lower-bound thresholding approach, has never been evaluated. Using the manual annotations created for DEDDIAG, this approach is evaluated using a newly introduced performance metrics called Jaccard-Time-Span-Event-Score (JTES). Together with this, a new segmentation algorithm using Support Vector Machine (SVM) is presented.

Finally, based on the usage events that were determined, a combined statistical model for appliance usage prediction is introduced. It predicts future appliance usage based on the preferred time of day and the elapsed time since it was used last. It is evaluated on the GREEND dataset as well as the DEDDIAG. The thesis concludes with an outlook of potential future work.

Zusammenfassung

Das Ziel, den weltweiten Strombedarf durch erneuerbare Energien wie Wind, Solar und Wasser zu decken birgt viele Herausforderungen. Eine dieser Herausforderungen ist das Gleichgewicht aus Angebot und Nachfrage, welche für die Stabilität des Stromnetzes unabdingbar ist und bisher ausschließlich über die Produktion geregelt wird. Bei einem Großteil der erneuerbaren Energien lässt sich die Stromproduktion jedoch nicht beliebig steuern, da sie von veränderlichen Umweltgegebenheiten abhängig ist. Angesichts dessen, dass Strom nicht in großen Mengen effizient gespeichert werden kann, stellt die Überproduktion ein ebenso großes Problem dar wie die Unterproduktion. Eine Lösungsansatz dafür ist die Lastverschiebung, wobei der private Verbraucher seinen Bedarf an die Produktion anpasst. Diesem wird hierfür ein finanzieller Anreiz in Form eines dynamischen Strompreises angeboten. Allerdings stellt diese Anpassung sehr hohe Anforderungen an den Verbraucher. Bereits in den 1980er Jahren entstand deshalb die Idee, zur Reduzierung des Aufwands für den Verbraucher maschinelles Lernen einzusetzen. Durch die Analyse von Stromdaten können damit dem Verbraucher Vorschläge für Nutzungsanpassungen gemacht oder auch Geräte automatisiert gesteuert werden. Ziel dieser Arbeit ist die Entwicklung neuer Methoden zur Informations- und Erkenntnisgewinnung aus Strommessungen privater Haushalte mittels maschineller Lernverfahren.

Zunächst werden die Grundlagen zu Datenquellen und zum generellen Vorgehen vorgestellt. Hieraus wird ein neues Modell abgeleitet, Machine Learning Demand Response Model (MLDR) genannt, welches die Zusammenhänge der notwendigen Daten und Schritte modelliert. Diese Schritte sind das Aufzeichnen von Daten, die Erkennung einzelner Geräte, die Segmentierung von Gerätenutzungen, sowie deren Vorhersage. Zu jedem dieser Schritte wird der aktuelle Stand der Forschung betrachtet und neue Verfahren vorgestellt.

Des Weiteren wird ein neues Messsystem für die Aufzeichnung von Stromverbrauchsdaten einzelner Geräte sowie eines gesamten Haushaltes beschrieben, mit dem der Forschungsdatensatz Domestic Energy Demand Dataset of Individual Appliances in Germany (DEDDIAG) aufgezeichnet wurde. Der Datensatz umfasst Messungen von 50 Geräten aus 15 Haushalten, die über einen Zeitraum von bis zu 3,5 Jahren mit einer Frequenz von 1 Hz aufgezeichnet wurden. Neben den reinen Messdaten enthält dieser Datensatz auch manuelle Annotationen von Gerätenutzung und demographische Daten des Haushalts. Sowohl das Messsystem als auch der Datensatz sind unter einer freien Lizenz veröffentlicht.

Auf Basis dieses Datensatzes wurde eine Gerätekategorieerkennung entwickelt. Diese basiert auf Wavelet-Merkmalen und dem k-Nearest-Neighbor (kNN) Klassifikator und wurde als Referenzwert für den Datensatz veröffentlicht. Neben diesem Ansatz wird ein weiteres Verfahren zur Geräteerkennung vorgestellt, welches jedoch deutlich höher auflösende Messungen benötigt. Dieses erkennt Geräte anhand von Strom-/Spannungsschwingungen, voltage-current (V-I) trajectory genannt. Strom und Spannung werden hierbei in zwei Recurrence Plots (RPs) umgewandelt und mit Hilfe eines Convolutional Neural Network (CNN) und Spacial Pyramid Pooling (SPP) klassifiziert. Der Algorithmus wird anhand von drei Datensätzen evaluiert und mit einem ähnlichen Verfahren verglichen.

Die unterschiedlichen Zeitpunkte der Gerätenutzung sind die Grundlage um Verbraucherverhalten zu verstehen. Für gewöhnlich wird diese Segmentierung von Messdaten mit einem Schwellwertverfahren durchgeführt. Da dieses Verfahren nur wenig erforscht und validiert ist, erfolgt eine Evaluierung unter Zuhilfenahme des DEDDIAG Datensatzes. Des Weiteren wird ein auf Support Vector Maschinen (SVM) basierender Segmentierungsalgorithmus vorgestellt.

Unter Verwendung der so gefundenen Gerätenutzungen wird ein statistisches Verfahren zur Vorhersage zukünftiger Gerätenutzungen eingeführt. Die Vorhersage basiert auf den gewöhnlichen Nutzungszeiten und dem gewöhnlichen Abstand zwischen Nutzungen. Das Verfahren wird unter Verwendung des GREEND sowie des DEDDIAG Datensatzes evaluiert. Abschließend wird ein Ausblick für zukünftige Forschungsbedarfe gegeben.

Acknowledgments

One's scientific journey is full of obstacles that feel insurmountable. My scientific journey was made possible by Prof. Dr.-Ing. Jochen Schmidt, who has always inspired me with his great knowledge and passion for science. He has been my supervisor for both my B. Sc. and M. Sc., and has given me the opportunity, support, and courage for my scientific path. The knowledge and wisdom he shared were outstanding and gave me the guidance I needed over the past few years. Thank you!

I am also grateful for the support of Prof. Dr.-Ing. habil. Andreas Maier, who has been a source of great ideas, knowledge and wisdom. He has, without hesitation, supported the cooperative work between the Technische Hochschule Rosenheim and Friedrich-Alexander-Universität Erlangen-Nürnberg and invited me into the Pattern Recognition Lab. Thank you!

Many others have been of great support. Thank you to Sebastian P. Bayerl, who has been a great sparing partner, critic, and most importantly a good friend. Thank you to Dr. Toni Göller and Thomas Eilks, at MINcom Smart Solutions GmbH, who have been a great partner throughout the project that enabled this research. The project was made possible through a funding by the German Federal Ministry of Education and Research (BMBF), grant 01LY1506. Only through the support of the faculty of Computer Science at the Technische Hochschule Rosenheim as well as the Bayerisches Wissenschaftsforum (BayWISS) was I able to complete this thesis. My gratitude goes particularly to those who contributed to building the monitoring system and to collecting the DEDDIAG dataset.

The love and care I received from family and friends, was without doubt the most vital part on this endeavor. My deepest gratitude goes to those who provided guidance, structure, care, and love - always and unconditionally.

So long and thanks for all the fish [Adam 84].

Marc Wenninger

Contents

CHAPTER 1

Introduction

1.1 Motivation

The modern world's largest threat remains rapid climate change with its most prominent indicator, the increasing global temperature. Electricity consumption is a major contributor to global warming and in 2019 63% of the world-wide electricity is still generated by burning fossil fuels [bp p 20]. Our technophile society maintains high hopes that modern technologies will solve the majority of these problems amid the ever-present rejection of behavioral change. In reality, these new technologies come with side effects such as a complexity increase. In order to handle this complexity, again new technologies are required. This complexity dilemma can also be found in energy generation from renewable sources.

A major attribute of most renewable energy sources, such as wind turbines or photo-voltaic systems, is the highly variable power production volume. Most renewable energy sources do not produce energy when demanded, but when natural conditions are suitable. A wind turbine generates energy in windy weather conditions, a photo-voltaic system when the sun is shining, and during daylight when it is less overcast. As energy cannot yet be stored very efficiently, over-production is as much of a problem as underproduction. Additionally, these systems are designed in a distributed way and with less scale compared to a single large power plant. According to the EN50160 regulation, the grid operator and energy suppliers are responsible for the grid's stability, which is highly affected by fluctuating energy demand and supply. The key factor for energy grid quality is an equilibrium of demand and supply. A deviation from the equilibrium will result in frequency fluctuations: An undersupply results in a reduced grid frequency and an oversupply in an increased frequency. The reason for these frequency fluctuations can be explained in a simplified scenario with a single generator. An increase of the active power drawn from the generator will increase the torque on the generator, thus without any countermeasures slowing down the rotational speed and lowering the utility frequency [Schw 09].

While grid operators always had to adjust to fluctuations on the demand side, the supply was a fairly stable part of the system since a few large power plants can be easily regulated to adjust to the demand. The general rule for energy production is: if the grid frequency $f > 50.2\,\mathrm{Hz}$, the effective power has to be reduced until $f \leq 50.05\,\mathrm{Hz}$. If this rule is applied independently at all plants, the grid might become even more unstable which has led to more sophisticated regulation [Scha 17]. In today's regulation model, only the supply side is responsible to maintain the grid quality, although all participants, including the demand side (the customer), are introducing changes to the system that lead to frequency changes. One step towards engaging the demand side would be to ask customers to adapt their demand to the supply. In order to encourage a customer to act, an incentive would need to be introduced [Gell 85]. This process is known as Demand Response (DR). There are clear differences when comparing the DR potential of different countries [Gils 14]. The contribution each consumer sector has in the overall DR potential depends on the flexibility of energy-intensive industries as well as the residential equipment, such as electric heating, air conditioning and washing appliances. DR creates a decision-making challenge for residents in a complex system, where for every usage of an appliance, its electrical load and the electricity price need to be estimated to make a cost versus need decision. Altering our electric usage is accompanied by discomfort, created by adding constraints on every day's habits. Incentives such as lower electricity prices have a limit in their effect. A study on the effectiveness of financial incentives to enable DR has shown that the participants were willing to adapt their practices to fluctuations in productions, as long as it does not rule their lives [Ozak 18]. Reducing the complexity of the task is necessary to increase the acceptance by the end-user [Quan 05]. Analyzing one's daily electricity habits and deriving automated recommendations is seen as the solution to lower the implication on daily habits. This analysis can be performed on monitored electricity consumption, either by monitoring each appliance directly or through a single metering point, also known as a smart meter. A wide range of machine learning practices have been developed and applied over the years to identify appliances, find usage patterns, and electricity profiles, and predict appliance usage. Despite the many years of research, the techniques are still subject of research and have not yet found their way into our lives. This thesis aims to model the relation between the required task by providing an extensive literature review on developed practices. Based on this review, a model of how to extract the relevant knowledge to lower the complexity of DR from household electricity data was developed. For each of the steps in the proposed model, a good scientific practice is proposed and applied to newly developed algorithms. All in the aim to work towards applicable computer aided DR.

1.2 Contributions to the Progress of Research

This work contributes to the overall systematic understanding of machine learning in DR. An introduction into the field of DR and it required infrastructure is given.

[Goel 18] T. Goeller, M. Wenninger, and J. Schmidt. “Towards Cost-Effective Utility Business Models - Selecting a Communication Architecture for the Rollout of New Smart Energy Services”. In: *Proceedings of the 7th International Conference on Smart Cities and Green ICT Systems - Volume 1: SMARTGREENS,*, pp. 231–237, INSTICC, SciTePress, 2018

Further, a new Machine Learning Demand Response Model (MLDR) is introduced, which shows the relations between data, knowledge, and actions associated with assisting with the DR process. It describes a detailed knowledge extraction model for approaches relying on electricity consumption monitored within a household. For each of these steps, a contribution with a focus on good scientific practice has been made. Established terminology has been gathered, potential terminology conflicts have been identified, and new terminology has been proposed.

1.2.1 Data

A data source for DR is everything that helps to understand the demand, behavior, and needs of a household. The primary source is electricity consumption measurements monitored at each appliance in a household. Next to the individual appliance readings, the total electricity consumption may be of interest. For this purpose, researchers have recorded and published a number of datasets using different sample rates and focusing on different aspects. Datasets with recording lasting over several months or years are sparse. Further, datasets recorded in Europe, especially Germany, are sparse or do not exist. Also, no monitoring software and hardware has been published that can be used to collect such datasets. Commonly the software was not published and the hardware was custom-built.

Thus, an open-source monitoring system was developed and the software as well as the used hardware was published. Unlike previously described monitoring systems for research purposes, the system uses hardware that is readily available for purchase at a very low cost. This system was used to collect a dataset containing 50 appliances from 15 private households at a sample rate of 1 Hz. The dataset contains manual usage annotations, which are required to develop and evaluate appliance usage segmentation algorithms. Such manual annotations are not available for any of the other existing datasets.

[Wenn 21b] M. Wenninger, A. Maier, and J. Schmidt. “DEDDIAG, a domestic electricity demand dataset of individual appliances in Germany”. *Scientific Data*, Vol. 8, No. 176, July 2021

1.2.2 Appliance Identification

One of the most researched topics in computer-aided DR is the identification of appliances. In machine learning terms, the identification of an appliance based on the monitored electricity consumption is a time series classification problem. The established approaches can be divided into low and high sample rates. A contribution has been made to both, low and high-sample rate approaches. Based on the collected Domestic Energy Demand Dataset of Individual Appliances in Germany (DEDDIAG) dataset, a new appliance identification algorithm has been proposed and published. The evaluation acts as a challenge baseline for future identification tasks performed on the published dataset.

[Wenn 21b] M. Wenninger, A. Maier, and J. Schmidt. "DEDDIAG, a domestic electricity demand dataset of individual appliances in Germany". *Scientific Data*, Vol. 8, No. 176, July 2021

A high sample rate approach has been developed based on Recurrence Plot (RP) and Convolutional Neural Network (CNN). The idea to incorporate RP first led to the development of a robust time series classification pipeline. This generic time series approach was evaluated on the UCR Time Series Classification Archive [Dau 18]. Based on this work, a high sample rate appliance identification approach was developed which converts the voltage-current (V-I) trajectory into a RP. The approach was evaluated on three different datasets. A similar approach was been proposed before, but relying on manually tuned hyperparameters [Faus 20, Faus 21]. The author's results could not be reproduced and as part of the publication of the new approach, our reevaluated results have been published.

[Wenn 19a] M. Wenninger, S. P. Bayerl, J. Schmidt, and K. Riedhammer. "Timage – A Robust Time Series Classification Pipeline". In: *Artificial Neural Networks and Machine Learning - ICANN 2019: Text and Time Series*, pp. 450–461, Springer International Publishing, Cham, 2019

[Wenn 21a] M. Wenninger, S. P. Bayerl, A. Maier, and J. Schmidt. "Recurrence Plot Spacial Pyramid Pooling Network for Appliance Identification in Non-Intrusive Load Monitoring". In: *20th IEEE International Conference on Machine Learning and Applications - ICMLA 2021*, 2021

1.2.3 Appliance Segmentation

Any appliance usage statistics require knowing an appliance's start and stop in order to derive usage patterns and load profiles. Finding these appliances' starts and stops means segmenting the monitored electricity consumption. The appliance segmentation only received little attention from other researchers. Based on the DEDDIAG dataset, the most commonly used lower bound thresholding algorithm was been evaluated. This thresholding algorithm is commonly used but has never been evaluated. Using the manual usage annotations from the DEDDIAG dataset showed the limitations of this approach. Second, a new Support Vector Machine (SVM) based algorithm has been proposed to overcome the limitations of the thresholding approach. To improve the real-world relation of the performed evaluations, a new performance metric called Jaccard-Time-Span-Event-Score (JTES) was developed. The importance of segmentation algorithms is underestimated and their evaluation has received little research attention. Important steps towards understanding the importance of the segmentation task and its performance evaluation have been contributed.

[Wenn 21b]	M. Wenninger, A. Maier, and J. Schmidt. “DEDDIAG, a domestic electricity demand dataset of individual appliances in Germany”. *Scientific Data*, Vol. 8, No. 176, July 2021
[Wenn 19b]	M. Wenninger, D. Stecher, and J. Schmidt. “SVM-Based Segmentation of Home Appliance Energy Measurements”. In: *18th IEEE International Conference on Machine Learning and Applications - ICMLA 2019*, pp. 1666–1670, 2019

1.2.4 Usage Prediction

Appliance identification and segmentation are performed in order to derive appliance-specific usage and load patterns. Historic usage patterns can be utilized to predict future appliances usages. Short-term prediction commonly only describes the next 1—24 h, while long-term predictions may describe up to several months ahead. The short-term predictions are the main focus in terms of automated operation of appliances or recommendation systems. A combined statistical method is proposed and evaluated on two different datasets, GREEND and DEDDIAG. For the latter, the manual annotations are used to train the thresholding algorithm. Commonly the used thresholds are simply estimated by researchers and their performance is unknown. Therefore, the evaluation of usage predictions commonly suffers from an unknown amount of errors made in the required segmentation step.

[Wenn 17]	M. Wenninger, J. Schmidt, and T. Goeller. “Appliance Usage Prediction for the Smart Home with an Application to Energy Demand Side Management - And Why Accuracy is not a Good Performance Metric for this Problem”. In: *Proceedings of the 6th International Conference on Smart Cities and Green ICT Systems - Volume 1: SMARTGREENS,*, pp. 143–150, INSTICC, SciTePress, 2017

1.3 Outline

The structure of this thesis is as follows:

Chapter 2 briefly describes the concept of load balancing and the origin of Demand Side Management (DSM). First, the practical problems that arise as a result of transforming electricity production from fossil and nuclear to renewable energy sources are laid out. The concept of DR and its relation to smart grids is explained. Different models of dynamic electricity pricing are discussed. The chapter ends with a brief overview of customer acceptance studies on DR and dynamic prices.

Chapter 4 lays out the principles of machine learning in DR. The DR process is modeled as an optimal control system as a basis to understand the complex system. Possible data sources and how to extract knowledge from those are discussed and research datasets are reviewed. Commonly used derived values (features) of the raw data and commonly used performance metrics are presented. The chapter finalizes with the introduction of a new model for machine learning in DR.

In Chapter 5 an electricity data collection system as well as the resulting dataset called DEDDIAG are presented. An overview of the hard- and software requirements, the design decisions, and an explanation of the system components is given. This is followed by a detailed description of the collected dataset, its validation, and publication is presented. The collected dataset was a big enabler for the research in the upcoming chapters.

Chapter 6 is about the identification of appliances based on low and high sample rate electricity data. The low sample rate approach is developed, trained, and evaluated on the DEDDIAG dataset, acting as a dataset baseline. A new high sample rate appliance identification method is compared to similar approaches and evaluated on three different datasets. Corrected results for one of the compared algorithms are presented.

In Chapter 7 the concept of event segmentation is introduced. The importance of event segmentation and its correct evaluation is explained. A new performance metric is introduced, overcoming the shortcomings of evaluating events based on single-time steps instead of as a whole. The commonly used thresholding algorithm is evaluated for the first time and results on the DEDDIAG dataset are presented. Additionally, a new segmentation algorithm is presented and compared to the thresholding method.

Chapter 8 presents work on the last step toward computer-aided DR, the prediction of appliance usage. A statistical model is presented and evaluated on two datasets, including a discussion on how to evaluate the predictions and the shortcomings of previous publications. Its concept is based on the preferred time of the day and the elapsed time between usages.

The thesis concludes with an outlook in Chapter 9 and a summary in Chapter 10.

CHAPTER 2

Demand Side Management

Generally, an equilibrium is a state of balance and in mathematical terms, it is a solution to a set of equations. In economics, the equilibrium is a method to describe a state of the market where "the agent (firms and households) maximizes some objective function (utility, profit) subject to some constraints (budget, technology)" [Cree 90]. In the electricity market, the equilibrium represents the balance of electricity supply and demand, which is of special interest, as it is a key factor in electric power quality. Electric power quality is the degree to which the sinusoidal waveform of voltage or current, as well as the frequency of an electric power supply system, conform to established norms. The demand on an electric grid is the physical quantity of power demanded at any given time, and the unit of power is *watt* (W). This is not to be confused with the consumed electrical energy per unit of time which has the SI-unit *watt-hours* (Wh). Satisfying this demand is the core challenge and main cost factor for grid providers [Meie 06].

Unlike traditional markets, the energy sector has treated the demand as an uncontrollable value instead of an adjustable variable. As a result, the sector is forced to create a highly flexible supply to guarantee grid stability [Gell 85]. In power engineering terms, the demand side is usually referred to as *load*, a fixed value that cannot be controlled by the supplier [Meie 06]. This means that the customer may, without any constraints, decide at any given time how much power is demanded and the supplier will have to meet the demand at any cost.

2.1 Load Balancing

The indicator for an imbalanced system is a deviation from the typical utility frequency of 50 Hz or in some parts of America and Asia 60 Hz. When more electricity is fed into the power grid than is demanded, the power grid frequency increases. If not enough electricity is fed into the power grid in relation to the consumed electricity, the frequency decreases. The reason for this is that the rotational speed of a generator is directly linked to the frequency of the generated electricity, meaning that the rotational speed needs to be kept constant in order to keep the frequency constant.

In a simplified scenario with a single generator, an increase in the active power drawn from the generator will increase the torque on the generator, thus without any countermeasures slowing down the rotational speed and lowering the utility frequency. In order to keep the rotational speed, the generator's primary energy supply (gas, water, steam) has to be increased in order to prevent a change in rotational speed and consequently the frequency of the generated electricity. The same logic applies if less power is drawn: The torque on the generator will decrease and therefore the generator's rotational speed increases resulting in an increase in the frequency of the generated electricity. [Schw 09]

Controlling the frequency in a network is much more complex compared to a single generator, and it is the transmission system operator's obligation to monitor and control the network frequency. They do this by planning and coordinating the power plant's activities in order to keep up with the demand. It requires daily planning of the electricity demand with the goal to predict the required electricity and adjust the generation accordingly, as well as putting in place an incident response plan to react to rapid changes and unplanned incidents.

2.1.1 Daily Demand Planning

The daily electricity demand planning is done on a grid level using a single Standard Load Profile (SLP). The SLP for German households shown in Fig. 2.1 was defined in 1999 and is known as *H0 SLP* [Bitt 99]. It is organized by workdays, Saturdays, and Sundays as well as by seasons of summer, winter, and transitional seasons. Next to H0, other SLPs have been defined for other consumer groups such as industrial (G0-G6) and agricultural (L0-L2).

2.1.2 Incidents Response

Next to the day-to-day demand planning, load balancing is done by responding to demand fluctuations using a control reserve, which are power plants on standby that are used to compensate for unforeseen demand increases. This control reserve is organized into three levels: the primary, secondary, and tertiary control reserve. A multi-stage plan is implemented to group different countermeasures [Foru 12]. When the grid frequency falls by >200 mHz, the control reserve is activated in order to generate the demanded electricity. On deviations >800 mHz, the steps to reduce the load are starting to take place, by deactivating storage pumps. In case the frequency falls below 49.0 Hz, multi-step load-shedding activities are activated that will take large consumers off the grid to further reduce the load. These load-balancing measures can be seen as an emergency response to rapid frequency drops that can be caused by a power plant or substation failure.

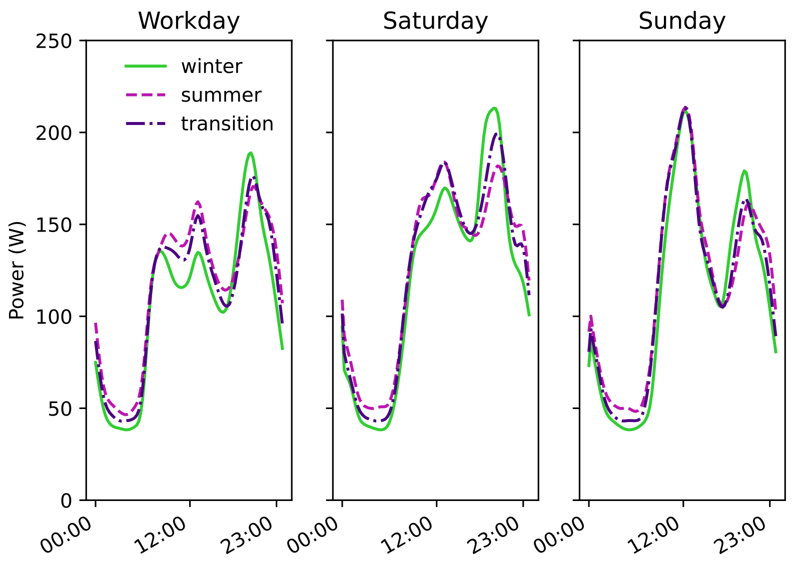


Figure 2.1: Standard load profile H0 used in Germany to forecast demand of households on a grid level. The load profile is organized in workdays, Saturdays, and Sundays as well as three seasons winter, summer, and transitional seasons. The profile is based on an annual consumption of 1000 kWh.

2.2 Origin of Demand Side Management

As an early solution for breaking this one-sided social contract between supply and demand, in 1985 Gellings described the term Demand Side Management (DSM). Gellings described DSM as the task of planning, implementing, and monitoring activities that influence the customers' electricity demand to provide desired changes in the load shape [Gell 85]. While Gellings very clearly defines DSM to only include load-shaping actions that are a deliberate incentive-based intervention into the market and do not include, e.g., the purchase of more energy-efficient utilities in order to reduce the general energy demand, the term DSM was redefined over time. In 1996, Gellings revisits the term DSM and concludes that it is mostly used in the context where consumers are encouraged to "conserve electricity, usually by purchasing super-efficient appliances and devices" [Gell 96]. Nowadays, DSM is usually understood as a combination of Demand Response (DR) and Energy Efficiency (EE) which can be understood as short-term and long-term load-shaping techniques. The United States Federal Energy Regulatory Commission defines DR as: "changes in electric use by end-use customers in response to changes in the price of electricity over time, or to give incentive payments designed to induce lower electricity use at times of high market prices or when grid reliability is jeopardized." [US D 06]. Figure 2.2 shows the current understanding of DSM, as the main term for all load saving (EE) and shaping (DR) efforts. DR actions can be seen as having a short-term load impact, as intended by Gellings, where the energy demand is shifted in time. EE actions have a long-term impact by reducing the overall load which does not provide any assistance in short-term demand management but contributes mainly to overall

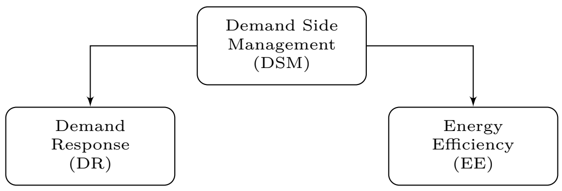


Figure 2.2: Demand Side Management term definition.

capacity reduction and long-term environmental goals to reduce the human impact on the environment.

2.3 Renewable Energy

The International Energy Agency (IEA) expects the worldwide electricity generation to increase to over 45,000 TWh, a growth of over 70% [IEA 19]. Wind and solar are expected to be the two main sources, supplying about half of the electricity generated by 2050.

The transformation of the electrical grid toward renewable energy sources has to be financed, which in many countries is reflected in the electricity price. In China, electricity prices are very low, mostly driven by very cheap domestic coal resources and cross-subsidies of the industry sector. The IEA expects the Chinese electricity prices for households to increase due to the introduction of CO_2 prices. Residential electricity prices are relatively stable in the United States and the European Union.

As of June 2023, Germany had one of the highest end-user electricity price in the world, rating at $0.399 per kWh [glob 23]. This is more than double the price of electricity in the US rating at $0.166 per kWh and over four times of the Chinese, rating at $0.078 per kWh. The major difference originates in a renewable energy levy, called EEG-Umlage, which was passed on to German residential consumers. The renewable energy levy has been eliminated in July 2022 [Germ 22]. Despite the fact that this should have significantly reduced electricity prices, as of June 2022, the price had risen to $0.520 per kWh. The Ukraine conflict has resulted in sharp price increases and significant volatility in energy markets [Adol 22]. Oil, coal, and gas prices skyrocketed in the aftermath of Russia's invasion of Ukraine and have remained volatile ever since. Increased gas prices, in particular, drove up wholesale electricity prices in the eurozone.

In Germany, the Climate Action Plan 2050 lays out the long-term transformation to reduce emissions in many sectors, including the electricity sector. The goal is to reduce greenhouse gas (GHG) emissions 40% by 2020, 55% by 2030, 70% by 2040, and 80-95% by 2050, which would mean Germany will be mostly GHG-free by 2050 [IEA 20]. For the electric sector, the plan is to increase the share of electricity gained from renewable sources to 80% by 2050. The energy sources have already changed significantly from a 15% share in 2008 to 35% in 2018, where particularly wind is the fastest-growing source. In 2017, the contribution by wind power exceeded both nuclear and natural gas and is now the second largest source after coal. In the year 2023, renewable energy sources had already contributed 56.0% to public electricity generation (see Fig. 2.3) [SMAR 24]. This high share of renewable energy

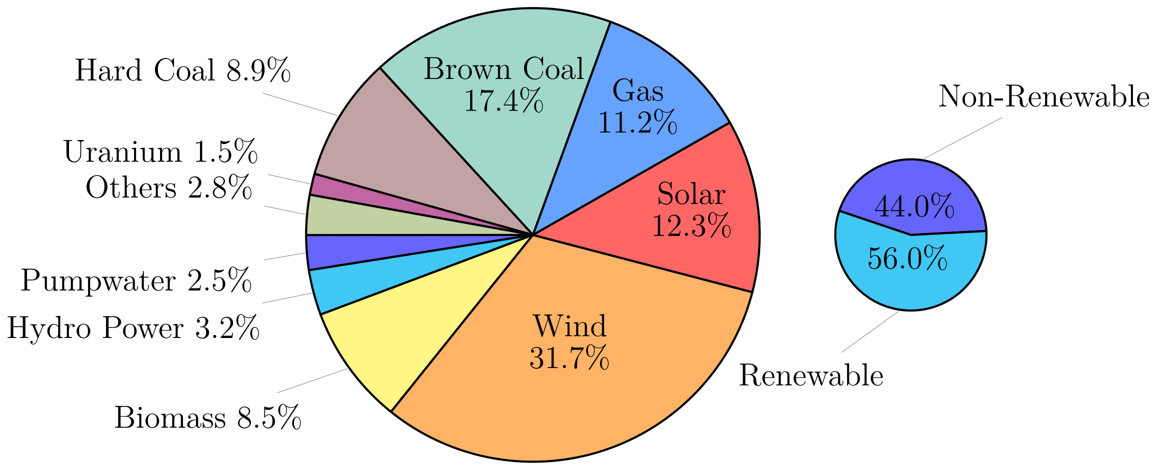


Figure 2.3: Net public electricity generation in Germany in 2023 [SMAR 24].

is attributed to favorable wind and sun conditions as well as a decline in industrial electricity demand as a result of the COVID-19 pandemic.

Increasing renewable energy introduces a new challenge on the supply side as with some sources, such as wind and solar, the supply side is less controllable. Solar energy is only available if there is sun; wind turbines will only produce energy if there is wind. The introduction of these less controllable variables in an already partially controllable equation exacerbates the unconditional supply-demand equilibrium.

Although the electricity demand is higher in the west and south, Germany has invested heavily in wind turbines in the northern parts of the country, neglecting the development of a cross-regional transmission grid to transport the energy to where it is needed [IEA 20]. In windy times this results in surplus capacity in the north during windy times to the point where wind turbines need to be taken off the grid. As an example, the German electricity network operator *Mitnetz* had to limit its wind capacity 357 times in 2019 [MITN 20].

With the goal set for 2050, the German public, government, electricity suppliers, and corporations will all have to be able to mitigate the challenges introduced by renewable energy sources. One of these challenges is adding the demand side to the control equation through Demand Response Management (DRM).

2.4 Demand Response

A very simplistic understanding of DR is the process of flattening the load curve. Therefore, DR describes a set of load-shaping actions that help to reduce the load during high-demand peak times and increase the load during low-demand off-peak times. In the literature, six categories of load-shape objectives are described: peak clipping, valley filling, load shifting, strategic conservation, strategic load growth, and flexible load shape [Gell 85]. The last three objectives, strategic conservation, strategic load growth, and flexible load shape mainly involve sales strategies and define large-scale changes and are not directly related to altering the peak-value

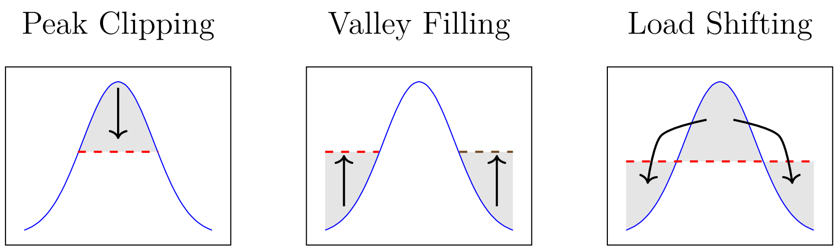


Figure 2.4: The three basic DR objectives as described by [Gell 85].

relation and are not further described here. The three basic DR objectives are shown in Fig. 2.4.

Peak clipping describes the reduction of peak loads by direct load control and represents the most basic form of load management. Direct control means that the operator can directly interrupt a customer's appliance. From a supplier's point of view, this is an efficient way to reduce operating costs by reducing the required peak capacity.

Valley filling means creating load during off-peak times, e. g., by storing energy in a different form. A simple example of creating load in a useful manner is pumped-storage hydroelectricity, where electricity is used to pump water from a lower elevated reservoir to a higher elevated reservoir. This way the electricity is stored as gravitational potential energy and can later be used during high-demand peak times. Next to such big electricity storage facilities, the coordinated charging of electric vehicles (EVs) has the potential for a decentralized valley-filling scenario [Zhan 14]. Studies conducted in Germany and Canada have revealed that controlled charging of electric vehicles has a significant impact on the respective countries' energy systems. [Stro 22, Dolu 20].

Load Shifting combines peak clipping and valley filling by shifting existing loads in time so that high-demand peaks fill low-demand valleys. This demand shift can be accomplished by rescheduling the demand, e. g., using the appliance at a different time or in industrial terms scheduling the process according to a predicted load demand. Traditionally load shifting was applied in the form of night storage heaters or hot water storage.

2.5 Smart Grid

The electric grid is the interconnected network that connects electricity production to consumption including all enabling services such as transmission and storage. Grids are a product of continuous adaption to increasing demand as well as changing forces such as the introduction of renewable energy as well as a wider-distributed generation. Throughout the world, the growth of an electric grid is influenced by multiple factors such as economics, politics, and geographic, but in general, the topology did not undergo any major changes [Farh 10]. The smart grid describes a modernization and automation of the current grid topology by integrating multiple modern and intelligent components [Gutw 09]. The transformation of the current state can be described as a pyramid as shown in Fig. 2.5. The smart grid is a combination of

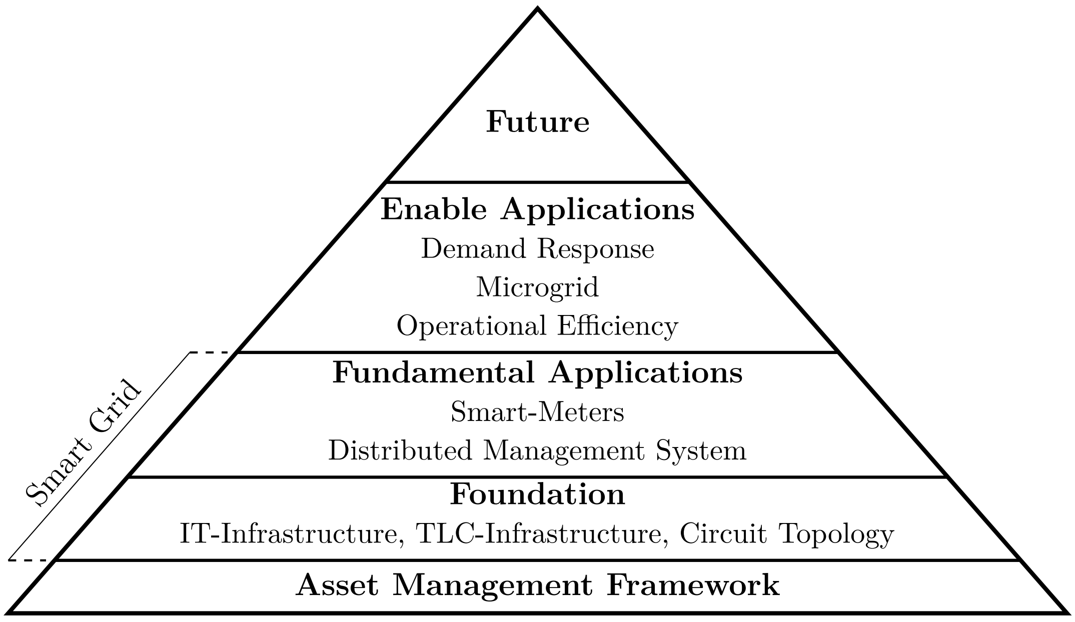


Figure 2.5: Smart grid pyramid showing the Asset Management as the foundation for Smart Grids which enables future applications such as DR [Gutw 09].

basic IT and communication infrastructure, as well as circuit topology and fundamental applications such as smart meters, data management, and various automation [Gutw 09].

The smart grid is the foundation that enables applications such as DR by defining the required components that provide the information and infrastructure for modern Distributed Energy Resources (DER).

The European Union has started a transformation of the European grid as part of the *Third Energy Package* in the directive 2009/72/EC [EU D 19]. It defines numerous goals in order to promote the internal electricity market and energy efficiency by "developing innovative pricing formulas, or introducing intelligent metering systems or smart grid". Regarding smart grid, all EU member states are encouraged to modernize their distribution network through smart grids. The smart grid "should be built in a way that encourages decentralized generation and energy efficient". As a result of this directive, the goal in Germany is to install a country-wide smart-metering infrastructure by 2032 (§29 Abs. 3 S.1 MsbG).

2.6 Dynamic Electricity Pricing

The standard model for retail electricity pricing is a flat fixed price for every kWh consumed. The price is fixed throughout the year and may only be altered annually. This fixed price model manifests the one-sided responsibility of load balancing as it does not provide any incentive to the customer to assist with DSM objectives. An early form of dynamic pricing is storage heating tariffs that were introduced as a means to fill the load valley at night, creating the necessary minimum load for power stations that otherwise need to be shut down. The different loads were metered by installing separate meters in order to attribute the usage. While storage heating using electricity is no longer widespread, the introduction of electric vehicles, which may be charged overnight, might introduce a new valley-filling demand at night.

Dynamic electricity pricing models can be implemented in different forms, where some require modern IT infrastructure in the form of smart meters. In general, there are three types of dynamic tariffs: Real Time Pricing (RTP), Time of Use (TOU), and Critical Peak Pricing (CPP) [Eid 16]. In all scenarios where the price model may change throughout the year, the installation of metering infrastructure capable of monitoring the rate-specific periods is required.

It must be noted that not all components of the retail electricity price are flexible. A retail electricity price is made up of costs of power generation, grid fees, electricity tax, Value added Tax (VAT), concession fees, and, in some countries, an additional levy. Some components might be charged per unit of energy and therefore are not flexible. A report on the German electricity market by the International Energy Agency in 2020 concludes that this limits the pricing model design that creates incentives for customer behavior changes [IEA 20]. In Germany dynamic pricing is, in the year 2021, still very uncommon and only a few providers exist such as Awattar (https://www.awattar.de).

Real Time Pricing (RTP) is the most dynamic price model where the price may change within short time periods such as every 15min or 1h. This price might be dependent on the day-ahead price which in Europe is traded on the European Power Exchange (EPEX) Spot Market (https://www.epexspot.com). In the case that the price is dependent on the day-ahead market, the customer has a price certainty of 24 hours and is notified on a day-ahead basis. RTP comes with high uncertainty for customers and in order to utilize the potential cost-saving aspect the customers need to check rates frequently or are faced with costs that are difficult to plan.

Time of Use (TOU) rates have a fixed electricity price for certain time blocks, usually defined for a 24-hour day. The rate is directly related to the average operational cost during those periods [US D 06]. An example is night storage tariffs where off-peak times are cheaper than high-peak times.

Critical Peak Pricing (CPP) is a hybrid of RTP and TOU. While the basic concept of time periods is similar to TOU, the actual rate is not fixed. It allows the utility to increase the price on short notice for certain days or periods within a year [US D 06]. In case this incentive plays out, the tariff contributes to peak clipping, thus lowering the overall operational cost of the network as well as contributing to network reliability.

2.7 Customer Acceptance

The effectiveness of all domestic DR programs depend heavily on the end-user's willingness to participate. A UK study analyzed the willingness to participate in dynamic electricity tariffs and analyzed their reaction to unpredictable price changes [Ozak 18].

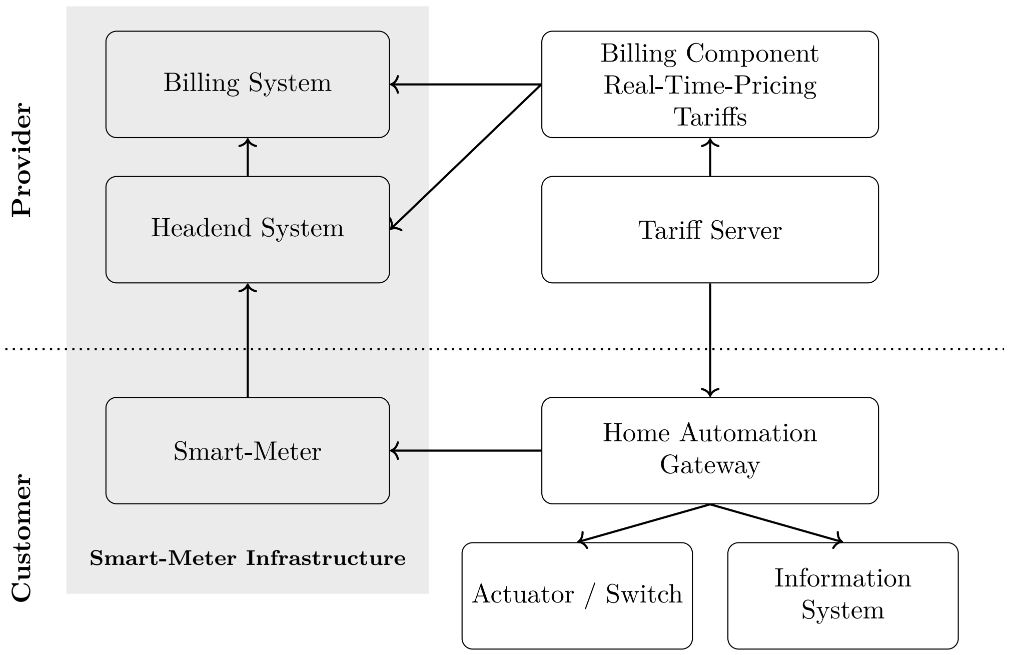


Figure 2.6: Technical infrastructure required to enable applications such as DR.

After the one-year trial period, the study concludes that the "trial participants were willing to adapt their practices to the rate fluctuations as long as the tariff did not rule their lives and ruin their quality of life". Further, the study concludes that real-time information on high energy usage might assist people to modify their own behavior since the study found that during a normal busy day, people need to be reminded of tariff changes through a simple interface. Real-time insights into electricity usage would also greatly assist in identifying high-load appliances including a "visual warning to let people know that what they are doing is consuming large amounts of electricity when the price is high". Other studies also suggest that RTP models, particularly, require a complexity reduction to be accepted by the end-user [Quan 05].

2.8 Demand Response Infrastructure

The technical infrastructure required to enable applications such as DR in a dynamic electricity prices scenario is shown in Fig. 2.6. On the provider side, it contains a headend system responsible for collecting electricity consumption data, relevant for billing purposes. In combination with a billing component and a tariff server, it provides the general capabilities for dynamic electricity price models such as RTP. On the customer side, a smart meter collects and transmits the electricity consumption. A component called the home automation gateway is responsible to combine information from the smart meter and tariff server. It provides information to the customer and may also directly operates actuators and switches. How such functionality can be provided is discussed in Chapter 4.

From a provider's point of view, the infrastructure serves as a platform to provide energy services. Utility services belong to basic needs and competition for physical access is unwanted. Therefore, in many countries including Germany, utility services such as metering, transmission, and remuneration of meter readings are highly regulated. In Germany, the operation of metering points is for privacy reasons separated from the distribution and sale of electric energy. This separation combined with the already many regulations in terms of network charges render offering new services via traditional market roles unprofitable [Goel 18].

Since providing applications beyond pure electricity supply usually requires hardware on the customer, the development of such applications depends on the development of this hardware. While the metering infrastructure is highly regulated, the hardware providing services such as a home automation gateway is not. Since this is unregulated, it allows nontraditional actors to enter the market and offer unregulated services. By using existing communication technology such as the customer's local network and internet, new services could be provided using unregulated hardware and communication. This lowers the entry threshold for Internet of Things (IoT) applications, allowing nontraditional actors to enter the market. This also means that the high data privacy standards introduced for electricity providers do not apply to such actors. For example, the lack of access to high sample rate energy consumption readings could be overcome by installing additional, unregulated meters.

The chosen infrastructure design and regulations heavily influence the range of services offered to customers, market access, and the cost of delivering public services. The currently rolled-out infrastructure has with all its regulations a high entry threshold, while at the same time, new services based on IoT hardware that operates within the customer's local network and have access to the internet have a very low threshold. This increases the risk, that a completely unregulated market is created next to infrastructure rolled out by utility providers.

CHAPTER 3

Machine Learning and Classification

Every meaningful action necessitates making decisions based on its surroundings. Creatures perceive their surroundings by observing, evaluating, and interpreting their sensory experiences [Niem 03]. In order for a computer to take meaningful actions it must possess knowledge of the action domain, thus perceiving its environment. Searching for patterns in the observed using computer algorithms is known as pattern recognition [Bish 06]. Sensory impressions only provide a subset of an environment, thus the actions we take are only based on a subset of possible impressions. Meaningful decisions start with perception, thus choosing the sensors defines the subset of the perceivable available to take an action and therefore is a significant factor defining the subsequent steps [Niem 03].

3.1 Machine Learning

The main goal of machine learning is to model the relationship between input and output. It focuses on developing algorithms and models that allow computers to learn from data rather than being explicitly programmed. This entails automatically detecting patterns between the source and the target. As a result, the terms Machine Learning and Pattern Recognition can be used interchangeably. Figure 3.1 shows a standard pipeline that allows a computer to perceive its environment and search for patterns [Niem 03].

Starting with the data provided by the used sensors, in a pre-processing step, the data are transformed to simplify feature extraction. The main purpose of pre-processing is to increase the data quality aiding the subsequent steps. From the pre-processed data, problem-relevant features are extracted, which results in feature vectors used by the classification algorithm. The classification itself is also called inference. Using the trained algorithm, the best-fitting class is determined. Model training is based on the assumption that recordings with similar characteristics belong

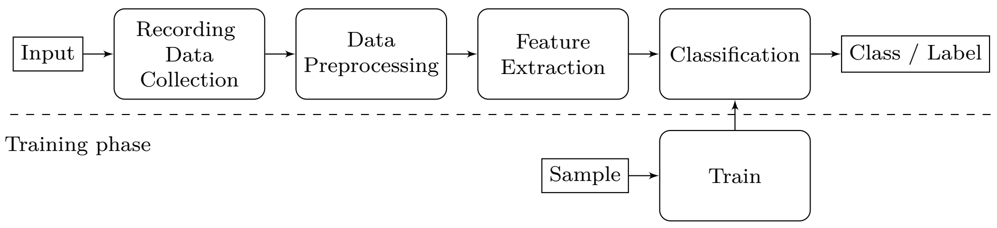


Figure 3.1: Pattern recognition pipeline as described by [Niem 03]. The top part begins with input, followed by a series of processing steps. Classification uses a trained model to predict the most likely output. A class can refer to any type of output, whether it be one-dimensional or multi-dimensional.

to the same class. Section 4.2.2 provides an overview of the features commonly used when working with electricity data. Model performance, and thus output correctness, is determined by feature expressiveness and model accuracy. This machine learning pipeline applies to a wide range of classification problems and goes beyond simple category assignments.

Depending on the application and the availability of input data, either supervised or unsupervised machine learning techniques are used. Supervised machine learning methods require labeled input data to learn the relationship between data and its correct label. The algorithm in supervised learning is trained using a set of example-label pairs known as the training set. The goal is to train the algorithm to predict the label for new, previously unseen data. Unsupervised learning, on the other hand, seeks previously unknown patterns and rules in unlabeled data. The algorithms are tested using separate data, and their accuracy is measured based on their ability to predict the correct labels for these new data [Bish 06]. The performance of machine learning pipelines is measured using various evaluation metrics, which are further described in Section 3.5.

3.2 Classification

Electricity data points and features are recorded chronologically, rendering most Demand Response (DR) related machine learning tasks a time-series classification problem. Time-series data has a numerical and continuous nature, and the temporal relation between the data points is of the essence. A time-series classifier is a function h that maps the sequence $\boldsymbol{x}$ to classes:

$$m = h(\boldsymbol{x}). \tag{3.1}$$

The function h is an element of some search space $\mathcal{H}$ of possible functions called the hypothesis space and $\boldsymbol{x}$ is an element of some feature space $\mathcal{X}$. For multi-class problems like appliance type recognition, we use $m \in \mathcal{M} = \{1, 2, \ldots, M\}$, (with M being the number of classes), possibly extended by class label 0 for rejection (i. e., the classifier is allowed to report "I don't know", as opposed to forcing a decision for one of the appliances). Note that for binary classification, $\mathcal{M}$ is typically chosen as $\{-1, +1\}$ (negative and positive class) or as $\{0, 1\}$, whichever is more suitable (often depending on the classifier used). Determining whether a device is ON or OFF, as described in Chapter 7 can be formulated as a binary classification problem.

In supervised learning, h is learned from a set $\mathcal{S}$ of example-label pairs called the training set:

$$\mathcal{S} = \{(\boldsymbol{x}_1, m_1), \ldots, (\boldsymbol{x}_N, m_N)\} \quad . \tag{3.2}$$

It is assumed that the set of training samples is representative of the problem, providing enough information about the perceived environment and that the elements are independent and identically distributed, i. e., one sample does not affect another one, and all samples are produced by the same underlying process. We, therefore, seek a function $h : \mathcal{X} \rightarrow \mathcal{M}$ that best fits our training set $\mathcal{S}$. This best fit is formalized by defining a loss function L, a single overall measure of loss obtained by choosing a specific classifier h. Thus, the goal of the learning process is to minimize total loss for our training set $\mathcal{S}$. A classification algorithm in general specifies how the function h may look like, how the search space $\mathcal{H}$ is searched, and what loss function L is to be used.

3.3 Classification in Demand Response

A wide range of different algorithms have been established over the years and have also been applied to demand response-related tasks. Figure 3.2 provides an overview of appliance identification (see Section 6) and event segmentation (see Section 7) approaches. The approaches differ in the data acquisition, pre-processing as well as classification method used, and evaluation of the approach. The categorization is not easy, as the established terminology for appliance identification and event segmentation is not well-defined. The application identification has been performed using a wide range of different classification algorithms such as Hidden Markov Model (HMM), Naïve Bayes (NB), Gaussian Mixture Model (GMM), Support Vector Machine (SVM), Neural Network (NN), k-Nearest-Neighbor (kNN), and Dynamic Time Warping (DTW). Event detection, being the less researched topic, was commonly only performed using kNN or SVM. In Chapter 6 two appliance identification approaches are presented based on kNN (see Section 6.3) and NN (see Section 6.4). The switching event segmentation approaches presented in Chapter 7 use a simple lower bound threshold approach (see Section 7.5) as well as a SVM based classification approach (see Section 7.6).

3.4 Classification Algorithms

The following describes the supervised classification algorithms used within this work. In supervised classification, an algorithm is trained on a labeled dataset, in which each sample is connected with a known class or label and is then used to predict the class of new, unseen data.

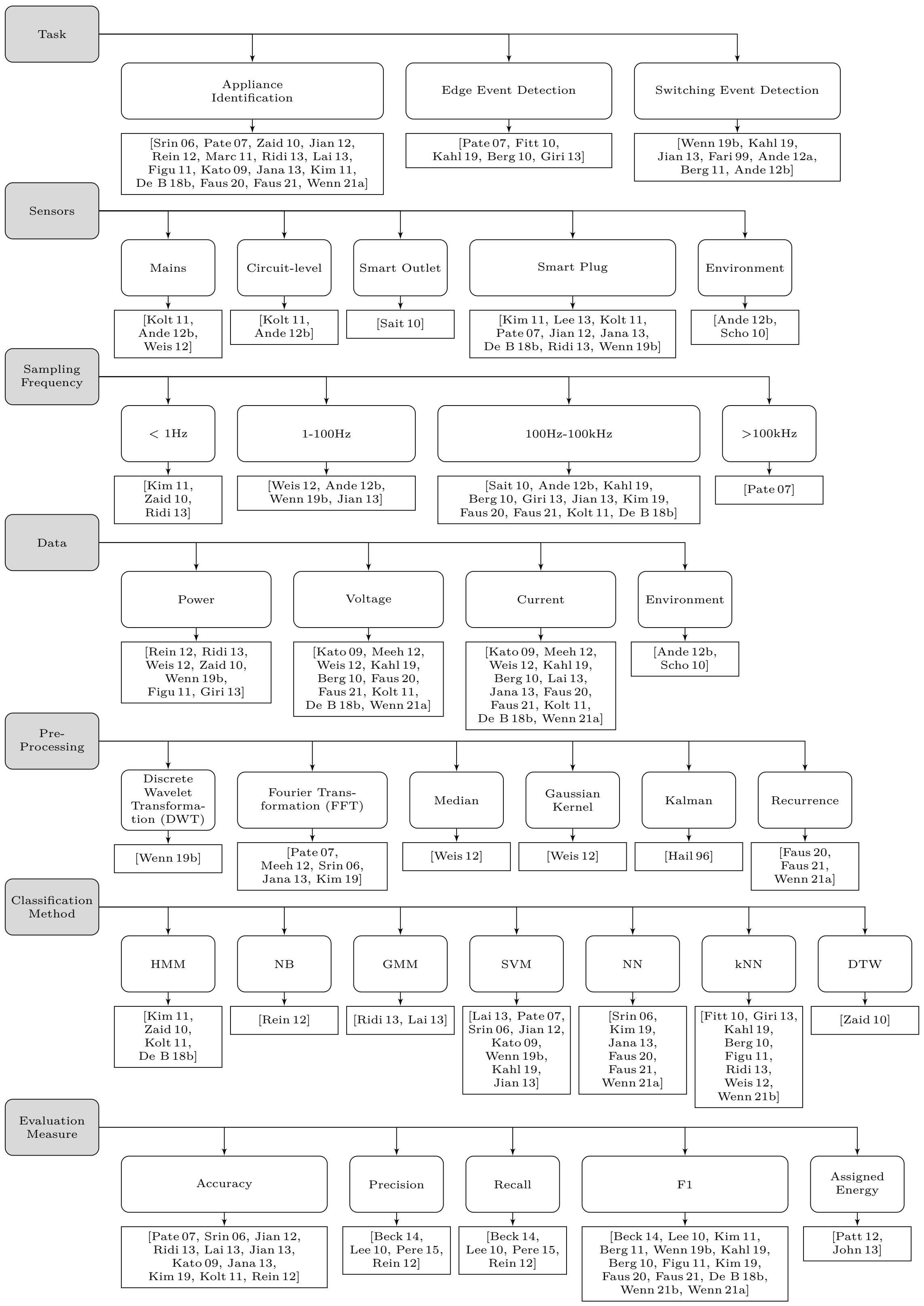

Task
Appliance Identification
[Srin 06, Pate 07, Zaid 10, Jian 12, Rein 12, Marc 11, Ridi 13, Lai 13, Figu 11, Kato 09, Jana 13, Kim 11, De B 18b, Faus 20, Faus 21, Wenn 21a]
Edge Event Detection
[Pate 07, Fitt 10, Kahl 19, Berg 10, Giri 13]
Switching Event Detection
[Wenn 19b, Kahl 19, Jian 13, Fari 99, Ande 12a, Berg 11, Ande 12b]
Sensors
Mains
[Kolt 11, Ande 12b, Weis 12]
Circuit-level
[Kolt 11, Ande 12b]
Smart Outlet
[Sait 10]
Smart Plug
[Kim 11, Lee 13, Kolt 11, Pate 07, Jian 12, Jana 13, De B 18b, Ridi 13, Wenn 19b]
Environment
[Ande 12b, Scho 10]
Sampling Frequency
< 1Hz
[Kim 11, Zaid 10, Ridi 13]
1-100Hz
[Weis 12, Ande 12b, Wenn 19b, Jian 13]
100Hz-100kHz
[Sait 10, Ande 12b, Kahl 19, Berg 10, Giri 13, Jian 13, Kim 19, Faus 20, Faus 21, Kolt 11, De B 18b]
>100kHz
[Pate 07]
Data
Power
[Rein 12, Ridi 13, Weis 12, Zaid 10, Wenn 19b, Figu 11, Giri 13]
Voltage
[Kato 09, Meeh 12, Weis 12, Kahl 19, Berg 10, Faus 20, Faus 21, Kolt 11, De B 18b, Wenn 21a]
Current
[Kato 09, Meeh 12, Weis 12, Kahl 19, Berg 10, Lai 13, Jana 13, Faus 20, Faus 21, Kolt 11, De B 18b, Wenn 21a]
Environment
[Ande 12b, Scho 10]
Pre-Processing
Discrete Wavelet Transformation (DWT)
[Wenn 19b]
Fourier Transformation (FFT)
[Pate 07, Meeh 12, Srin 06, Jana 13, Kim 19]
Median
[Weis 12]
Gaussian Kernel
[Weis 12]
Kalman
[Hail 96]
Recurrence
[Faus 20, Faus 21, Wenn 21a]
Classification Method
HMM
[Kim 11, Zaid 10, Kolt 11, De B 18b]
NB
[Rein 12]
GMM
[Ridi 13, Lai 13]
SVM
[Lai 13, Pate 07, Srin 06, Jian 12, Kato 09, Wenn 19b, Kahl 19, Jian 13]
NN
[Srin 06, Kim 19, Jana 13, Faus 20, Faus 21, Wenn 21a]
kNN
[Fitt 10, Giri 13, Kahl 19, Berg 10, Figu 11, Ridi 13, Weis 12, Wenn 21b]
DTW
[Zaid 10]
Evaluation Measure
Accuracy
[Pate 07, Srin 06, Jian 12, Ridi 13, Lai 13, Jian 13, Kato 09, Jana 13, Kim 19, Kolt 11, Rein 12]
Precision
[Beck 14, Lee 10, Pere 15, Rein 12]
Recall
[Beck 14, Lee 10, Pere 15, Rein 12]
F1
[Beck 14, Lee 10, Kim 11, Berg 11, Wenn 19b, Kahl 19, Berg 10, Figu 11, Kim 19, Faus 20, Faus 21, De B 18b, Wenn 21b, Wenn 21a]
Assigned Energy
[Patt 12, John 13]

Figure 3.2: Overview of appliance identification and event segmentation approaches.

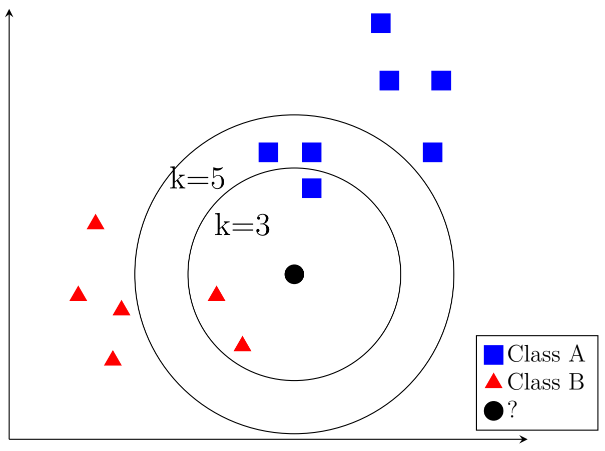


Figure 3.3: Classification example of a new feature vector (black dot) using the k nearest neighbor (kNN) method for different k. If three neighbors are decisive ($k = 3$), the new point is classified as class B. If five neighbors are decisive ($k = 5$), the new point is classified as class A.

3.4.1 k-Nearest-Neighbor Method

The development of the kNN classification method goes back to E. Fix, and J. Hodges [Fix 51]. It is a simple, non-parametric, supervised classification method suited for discrimination problems when reliable parametric estimates of probability densities are unknown or hard to determine. The kNN technique has been widely used in DR tasks [Fitt 10, Giri 13, Kahl 19, Berg 10, Figu 11, Ridi 13, Weis 12, Wenn 21b] (see Figure 3.2). Let's assume a training set of observations/samples $O_t = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ and their corresponding class labels $C_t = \{y_1, y_2, \ldots, y_N\}$. Each observed vector X_n is composed of several observed features. The classifier provides an answer to a discrimination problem by calculating the distance to a specific observation $\boldsymbol{x}_s$, with unknown class label y_s, by measuring the distance of $\boldsymbol{x}_s$ to the observations in O_t. The classification rule is, to assign $\boldsymbol{x}_s$ the majority class label of its k nearest neighbors in O_t.

A general distance metric definition describing the distance between two observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is known as the Minkowski distance and defined as:

$$D(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\sum_{q=1}^{Q} |x_{iq} - x_{jq}|^p \right)^{\frac{1}{p}} \quad , \tag{3.3}$$

where Q is the feature vector size and $p \geq 1$ determines the actual distance metrics used. The typical values for p are 1 and 2, where $p = 1$ corresponds to the Manhattan distance, and $p = 2$ corresponds to the Euclidean distance. The Euclidean distance is the most commonly used. Using (3.3) with $p = 2$, results in:

$$D(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_{q=1}^{Q} |x_{iq} - x_{jq}|^2} \quad . \tag{3.4}$$

Figure 3.3 illustrates the kNN classification based on the Euclidean distance using three ($k = 3$) or five ($k = 5$) neighbors. It is convenient to choose k to be odd to avoid ties. Since the decision process is based on the training samples at prediction time, the training phase includes storing the training observations. Thus, the training set size does not really influence the training time but the prediction time.

3.4.2 Support Vector Machine

SVM is a supervised learning method used for classification and regression analysis; however, only the former is relevant here. It tries to find a boundary in a high-dimensional space that separates data points that belong to different classes [Cort 95]. The border is designed to maximize the margin between the closest points from distinct classes, which are referred to as support vectors. Therefore, memorizing all training set observations is unnecessary as only a limited number of observations are crucial. Even if the training set is limited in size or contains outliers, SVMs still has a decent generalization. This is one of the advantages of using SVMs. Non-linear data can be handled by mapping it into a higher-dimensional space and locating the boundary there. The SVM technique has been widely used in DR tasks [Lai 13, Pate 07, Srin 06, Jian 12, Kato 09, Wenn 19b, Kahl 19, Jian 13] (see Figure 3.2).

Figure 3.4 shows an example of a maximum margin hyperplane or decision boundary that separates two classes. As the example shows, no sample from either class is within the margin on either side of the separation hyperplane. This is known as hard margin SVM. Suppose an optimal solution or error-free separation of the datasets is not attainable. In that case, this restriction can be relaxed to allow samples to lie inside the margin, known as soft margin SVM. The data points on the dashed lines act as support vectors.

A linear model for a binary classification problem can be expressed as:

$$y(\boldsymbol{x}_i) = \boldsymbol{w}^T \boldsymbol{x_i} + b \quad , \tag{3.5}$$

where $\boldsymbol{w}$ is the weight vector that defines the separating hyperplane, $\boldsymbol{x}_i$; the data samples with corresponding class labels $y(\boldsymbol{x}_i) = y_i \in \{-1, 1\}$, and b the bias component. Two parallel hyperplanes can be chosen that maximally divide the two classes:

$$\boldsymbol{w}^T \boldsymbol{x}_i + b \geq 1, \quad \text{for} \quad y_i = 1 \tag{3.6}$$

for the first class denoted as $y_i = 1$, and

$$\boldsymbol{w}^T \boldsymbol{x}_i + b \leq -1, \quad \text{for} \quad y_i = -1 \tag{3.7}$$

for the second class denoted as $y_i = -1$. This can be expressed as a set of inequalities:

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1 \quad i = 1, 2, \ldots, N \tag{3.8}$$

where N is the number of training samples and $\boldsymbol{x}_i$ is the i_{th} training sample. The region encompassed by these two hyperplanes defined in (3.6) and (3.7) is referred to as the margin. The maximum-margin hyperplane is the hyperplane that lies at their midpoint:

$$\boldsymbol{w}^T \boldsymbol{x} + b = 0 \quad . \tag{3.9}$$

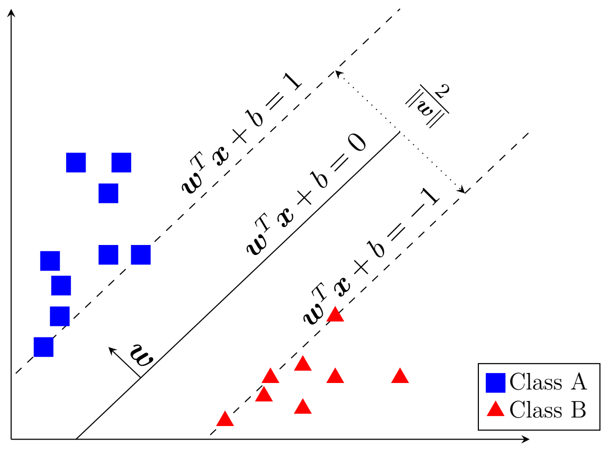


Figure 3.4: Example of data consisting of two classes that have been separated using SVM with the maximum margin separation hyperplane. The points on the dashed lines act as support vectors.

The geometric distance between these two hyperplanes (3.6) and (3.7) is $\frac{2}{||\boldsymbol{w}||}$, hence to maximize the distance between the planes, we must minimize $||\boldsymbol{w}||$. Minimizing $||\boldsymbol{w}||$ within the limitations of (3.8) yields the best normal vector for separating the feature space.

The Lagrange multiplier method can be utilized to solve this optimization problem, which will then result in the dual formulation of the SVM optimization problem. The Lagrangian is constructed as [Bish 06]:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{N} \alpha_i \{y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1\} \quad , \tag{3.10}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)^T$. Setting the derivatives of $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ with respect to $\boldsymbol{w}$ and b to zero, the following two conditions are obtained:

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i \quad , \tag{3.11}$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i \quad . \tag{3.12}$$

Eliminating $\boldsymbol{w}$, b from (3.10) by setting $\boldsymbol{w}$ and b's using these conditions yields the dual representation of the maximum margin problem:

$$\widetilde{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad , \tag{3.13}$$

with respect to the constraints:

$$\alpha_i \geq 0 \quad , \quad i = 1, \ldots, N, \tag{3.14}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad . \tag{3.15}$$

The kernel function k is in the case of a linear kernel given by $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T\boldsymbol{x}'$. A polynomial kernel for example can be formulated by $k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T\boldsymbol{x}' + c)^d$. These kernel functions offer exceptional performance for data points that cannot be separated linearly, which is one of the primary reasons why SVMs have gained such widespread adoption. Since the decision function only depends on the inner product of the vectors in the feature space, it suffices to evaluate the kernel function instead of explicitly projecting it into the feature space.

To use the trained model to categorize additional data points, we evaluate the sign of the function $y(\boldsymbol{x})$ defined in (3.5). This can be represented in terms of the parameters $\boldsymbol{\alpha}$ and the kernel function k by substituting for $\boldsymbol{w}$ using (3.11):

$$y(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b \quad . \tag{3.16}$$

The presented constrained optimization problem meets the Karush-Kuhn-Tucker requirements, which necessitate the occurrence of the following properties:

$$\alpha_i \geq 0 \quad , \tag{3.17}$$

$$y_i(\boldsymbol{w}^T\boldsymbol{x} + b) - 1 \geq 0 \quad , \tag{3.18}$$

$$\alpha_i\{y_i(\boldsymbol{w}^T\boldsymbol{x} + b) - 1\} = 0 \quad . \tag{3.19}$$

It can be deduced that samples either satisfy the condition $\alpha_i = 0$ or $y_i(\boldsymbol{w}^T\boldsymbol{x}+b) = 1$. Every data point for which $\alpha_i = 0$ will not affect the sum in (3.16) and is therefore irrelevant for any prediction. The remaining data points satisfying $y_i(\boldsymbol{w}^T\boldsymbol{x} + b) = 1$ are the support vectors and lie on the maximum margin hyperplane.

3.4.3 Deep Learning

Researchers like Warren McCulloch and Walter Pitts [McCu 43] in the 1940s and 1950s claimed that the human brain's functioning could be described as a simple network of binary neurons, laying the groundwork for the development of modern artificial NNs [Bish 06]. When fast backpropagation algorithms were developed in the late 1980s and early 1990s, it became possible to train multi-layer NNs [Rume 86]. During this period, NNs were put to use in many contexts, including but not limited to pattern recognition, image and speech processing, and control systems. Unfortunately, early NNs' performance was constrained by the size of available computational resources and training datasets. While the topic of deep learning has been around since the 1960s [Widr 90], it wasn't until the late 2000s and early 2010s that breakthrough results were produced on a variety of tasks using deep learning NNs, sparking increased interest in the area. Since then, deep learning has expanded into a major field of study with many practical applications in industry. NNs have been widely used in DR tasks [Srin 06, Kim 19, Jana 13, Faus 20, Faus 21, Wenn 21a] (see Figure 3.2). Unlike other areas where large amounts of data are easily available, utilizing NNs in DR task is sometimes challenging as vast amounts of meaningful training data are not easily available. This is further discussed in Section 4.1.3.

Traditional machine learning methods, such as SVM and kNN are typically based on simple mathematical models and require manual feature engineering to extract useful information from the data. Compared to deep learning models, they often have a shallower architecture and are less capable of handling vast and complicated datasets. In contrast, deep learning models have a more complex architecture with numerous hidden layers that can automatically learn and extract features from raw data. This enables them to manage huge and complicated datasets and model non-linear solutions. In addition, deep learning models can improve continually through training, but older methods have a limited capability for improvement without human involvement.

Feed-forward Neural Networks

The foundation of deep learning models is feed-forward neural networks or Multilayer Perceptrons (MLPs). A computational perceptron, also called a neuron, is the smallest component of an MLP, which is composed of many neurons stacked in layers. As illustrated in Figure 3.5(a), an input layer, one or more hidden layers, and an output layer make up their structure. The input layer takes the input data, while the output layer generates the network's predictions. Complex relationships between input and output data are modeled using the hidden layers. Since all neurons in one layer are typically connected to all neurons in the following layer, these layers are also referred to as fully connected (FC). Each layer provides a function $f(x)$ that processes the output of the previous layer. Together they forms a chain of functions $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$, where $f^{(1)}$ is the input layer, $f^{(2)}$ the hidden layer, and $f^{(3)}$ the output layer [Good 16]. The term deep in deep learning refers to the overall length of the chain, incorporating different aspects of the classification pipeline such as feature extraction. As depicted in Figure 3.5(b), a single perceptron consists of weights $\boldsymbol{w}$ and bias b that process the input features, apply an activation function h, and generate the output a_j. Activation functions introduce non-linearity to the neuron's output. The activation function determines whether or not a neuron should be activated, i.e., whether or not its output should be forwarded to the next network layer. Some of the commonly used activation functions include sigmoid, hyperbolic tangent (tanh), Rectified Linear Unit (ReLU), and variants of ReLU such as leaky ReLU.

An MLP discovers the model parameters of a function f such that the mapping of input $\boldsymbol{x}$ to output y is optimally approximated by $y = f(\boldsymbol{x}, \boldsymbol{\theta})$ [Good 16]. The model $\boldsymbol{\theta}$ contains all trainable parameters. In an MLP, these parameters are the weights $\boldsymbol{w}$ and biases b of all neurons. For a single neuron j, this is described as:

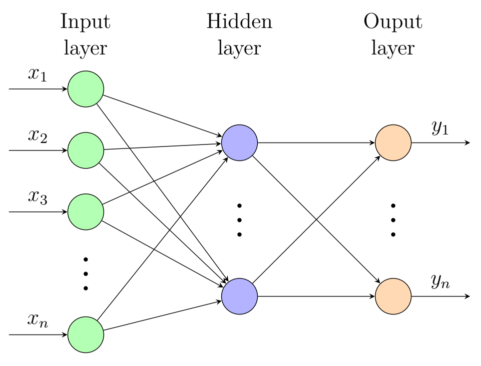
$$z_j = \boldsymbol{w}_j^T \boldsymbol{x} + b_j \quad . \tag{3.20}$$

Applying an activation function h to z_j, the output a_j of a layer's neurons $1, \ldots, M$ can be described as:

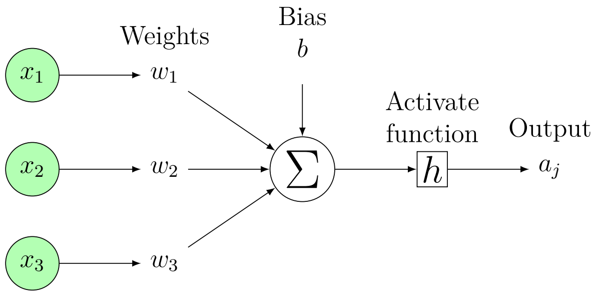
$$a_j = h(\boldsymbol{w}_j^T \boldsymbol{x} + b_j) \quad . \tag{3.21}$$

In classification problems, the last layer must produce an output indicating class membership y. For binary classification tasks, this can be achieved by using the logistic sigmoid activation function:

$$y = \frac{1}{1 + \exp(-z)} \quad , \tag{3.22}$$



(a) Multilayer perceptron



(b) Computational perceptron

Figure 3.5: An MLP as shown in a) comprises an input, hidden, and output layer. The input layer receives the input data and passes it through the network, where each hidden layer processes and passes the data until it reaches the output layer. Each layer's associated perceptrons, as shown in b), employ weights to determine the strength of their connections and apply activation functions to produce the final output predictions.

so that $0 \leq y \leq 1$. For non-binary classification problems with $K > 2$ classes, a softmax activation function is used. A softmax activation function represents the probability distribution over a discrete variable and is defined as [Bish 06]:

$$p(y_k|\boldsymbol{x}) = \frac{exp(z_i)}{\sum_{j=1}^{K} exp(z_j)} \quad , \tag{3.23}$$

It applies the standard exponential function to each element z_i of the input vector $\boldsymbol{z}$ and normalizes these values by dividing by the sum of all these exponentials. This normalization ensures that the sum of the components of the output vector is 1.

Training of Neural Networks

The weights and biases are learned using a supervised learning algorithm such as gradient descent, which updates the parameters based on the difference between the predicted output, and the actual output [Good 16]. This difference between predicted and actual output is expressed as a cost function $\mathcal{C}(\boldsymbol{\theta})$ to find the optimal parameters. Initially, all network parameters $\boldsymbol{\theta}$, such as weights and biases, are usually initialized randomly and optimized by minimizing this cost function. The cost function is employed as an indirect performance metric, expecting to improve the actual performance metric used to evaluate the problem solution. These actual performance metrics are further explained in Section 3.5.

The cost function $\mathcal{C}(\boldsymbol{\theta})$ can be expressed as an average of the per-example loss function L:

$$\mathcal{C}(\boldsymbol{\theta}) = \mathbb{E}_{(x,y)\sim\widehat{p}_{data}} L(f(\boldsymbol{x}, \boldsymbol{\theta}), y) \quad , \tag{3.24}$$

where the input is $\boldsymbol{x}$, $\widehat{p}_{data}$ is the empirical distribution, $f(\boldsymbol{x}, \boldsymbol{\theta})$ the network's predicted output, and y the expected class membership. The typical loss function for classification problems is the cross-entropy between the training data and the model distribution:

$$L = -\sum_{k=1}^{K} y_k \log(a_k) \quad , \tag{3.25}$$

$$\mathcal{C}(\boldsymbol{\theta}) = -\mathbb{E}_{(x,y)\sim\widehat{p}_{data}} \log(f(\boldsymbol{x}, \boldsymbol{\theta}), y) \quad . \tag{3.26}$$

Typically, the answer is obtained by evaluating the gradient of the cost function using mini-batches. The gradient of the cost function must be sufficiently large and predictable to serve as a solid guide for the learning process, which is a repeating theme in neural network design. Mini-batching is a technique where the model is only trained on smaller subsets of data, called batches, instead of the entire dataset at once. The training dataset is divided into batches, and the model is trained on each batch, updating the model's parameters after each batch. This approach allows for more efficient use of computational resources and faster convergence compared to training on the entire dataset at once. Additionally, mini-batching can improve generalization performance and avoid overfitting by introducing randomness and reducing the model's sensitivity to individual examples in the dataset. Using the stochastic gradient descent (SGD) algorithm, the gradient is computed as [Good 16]:

$$\nabla\mathcal{C}(\boldsymbol{\theta}_t) = \frac{1}{N_b} \nabla_{\boldsymbol{\theta}_t} \sum_{i=1}^{N_b} L(f(\boldsymbol{x}_i, \boldsymbol{\theta}_t), y) \quad , \tag{3.27}$$

where N_b is the size of the mini-batch. Utilizing the cost function $\mathcal{C}$, the parameters $\boldsymbol{\theta}$ are iteratively adjusted using:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\nabla\mathcal{C}(\boldsymbol{\theta}_t) \quad , \tag{3.28}$$

guiding the learning process using the learning rate η. It is possible to set the learning rate through trial and error, but observing learning curves that display the objective function as a function of time is commonly advised. This is more of an art than a science. The backpropagation algorithm is used to compute gradients relative to the weights. Initially introduced by Rumelhart et al. [Rume 86], the backpropagation approach updates the weight of each neuron based on the amount of error it has given to the network, as assessed by the loss function computed in the last layer.

Therefore, the training technique consists of two steps: a forward pass, also called forward propagation, and a backward pass, also called backpropagation. The forward pass of the network predicts the class membership y by computing (3.21) at each layer. In the final layer, this means applying an activation function such as (3.22) or (3.23). The backward pass computes the gradient of the activation functions with respect to the network's weights, beginning at the output layer and working its way back to the input layer. The loss L_n at the last layer n depends on the weights w_{ji} via the summed input z_j at layer j [Bish 06]. According to the chain rule for partial derivatives, the derivative of a composite function equals the product of the derivatives of the individual functions that comprise the composite function. This means that the loss function's derivative with respect to a layer's weights can be written as the product of the derivative of that layer's output with respect to its weights and the derivative of the loss function with respect to that layer's output:

$$\frac{\partial L}{\partial w_{ji}} = \frac{\partial z_j}{\partial w_{ji}} \frac{\partial L}{\partial z_j} \quad . \tag{3.29}$$

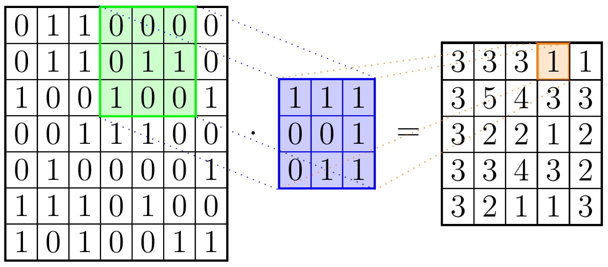


Figure 3.6: Convolutional operation used in CNNs, where a filter traverses the input data and calculates the dot product between the filter weights and the corresponding input value.

The same procedure is then performed for the network's hidden layers, with the gradient being calculated depending on the previous layer's error and the current layer's weights. Using this gradient, the weights of the hidden layers are then updated.

Convolutional Neural Networks

On problems where spacial features are relevant, Convolutional Neural Networks (CNNs) present a powerful type of NNs based on the convolution operation. A CNN's foundation is a convolutional layer, which applies a sequence of filters or kernels to the input data. Convolutional layers became of interest when applied to handwriting problems in 1989, enabling a single network to learn the entire recognition operation [LeCu 89]. For the convolution, each filter traverses the input data and calculates the dot product between the filter weights and the corresponding input values as depicted in Figure 3.6. The output is a feature map highlighting particular aspects of the original data. Multiple CNN layers allow the network to extract increasingly complex and abstract features from the input data. The first layer learns simple features like edges, curves, and blobs, while subsequent layers learn more complex features that are composites of the simpler features learned earlier. The network can also learn hierarchical representations of the input data, with each layer building on the features learned in the previous layer. Using multiple layers, CNNs are capable of capturing local and global information.

Convolutional layers have three distinguishing characteristics [Good 16]. (1) *Sparse interactions*, connectivity, or weights result from the kernel being much smaller than the input. Small kernels recognize edges and other small, representative features. As a result, fewer parameters must be saved, reducing memory usage, increasing efficiency, and decreasing the number of operations required. (2) *Parameter sharing* re-uses the kernel and weights across input data. On the other hand, fully connected layers have distinct weights for each input. Parameter sharing reduces memory usage even further and prevents overfitting. (3) *Equivariance* to translation, meaning that the output changes the same way as the input. As a result, moving the input in one direction results in a corresponding shift in the activation map, which is useful because early layers may have edges or other basic features at different locations. The convolution operation can be implemented using matrix operations, improving performance through sparsity and parameter sharing while maintaining gradient flow via backpropagation for fully connected layers.

A common addition to convolutional layers is pooling layers, which are often applied after nonlinear activation functions to minimize the size of the activation maps. Pooling is employed by almost all convolutional networks [Good 16]. It is achieved by applying a mathematical operation, such as maximum or average pooling, to non-overlapping subregions of the input data. Max pooling, for instance, uses the maximum value of each subregion as the output. This reduces the size of the feature maps, rendering them more manageable and computationally efficient. The subregion's size can be fixed with a kernel size of, for example, 2×2, or in the case of adaptive pooling; the size is defined based on the desired output size. In the case of a 2×2 kernel, the dimension of the feature map is reduced by one-fourth. The pooling layer has numerous benefits. First, it aids in the management of overfitting, which occurs when the model memorizes the training data rather than learning to generalize from it. The pooling layer minimizes the number of network parameters by decreasing the feature maps' size. Second, it enhances translation invariance, or the model's capacity to recognize shapes in various locations or orientations. Pooling achieves this by increasing the model's sensitivity to modest changes in input.

3.5 Performance Metrics

Measuring the effectiveness of an algorithm requires a performance metric. A metric provides two functions, a measure of the effectiveness of the tested approach and the basis for comparison to other approaches. In general, two kinds of metrics are commonly used: binary-attribute and value-based. Binary-attribute-based metrics compare the binary output of a classifier by testing whether the output is correct or incorrect. The binary states are usually called positive or negative. Binary-attribute-based metrics are the common choice for classification problems such as identification and edge detection. Value-based metrics are a measure of the difference in value between ground truth and prediction. For disaggregation tasks, it is common to express the error as the distance between the disaggregation load curve of an appliance and the ground truth.

Choosing a meaningful performance metric is significant to the evaluation as different metrics express different characteristics. Several authors in various fields have addressed the problem of selecting an appropriate metric before [Cook 07, Hand 09, Powe 11a, Mako 15].

The basis for binary-attribute metrics are the number of true positives (TP), true negatives (TN), false positives (FP), false negatives (FN). TP means that the positive class was correctly classified as positive. TN means that the negative class was correctly classified as negative. FP and FN are the numbers of erroneously classified samples, either falsely as positive or negative. These numbers are the result of summarizing the predictions of a classifier and form a 2×2 contingency table [Powe 11b]. Based on this contingency table, multiple single-value metrics can be described.

3.5.1 Accuracy

Accuracy (ACC) expresses the ratio of correct predictions to total predictions:

$$ACC = \frac{\text{correct predictions}}{\text{total predictions}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \quad . \tag{3.30}$$

The main issue with this commonly used metric is known as the *accuracy paradox* [Zhu 07, Valv 14]. When the classes (positive and negative) are imbalanced, as is the case for rare events, a high ACC is easily obtained by always predicting the most frequent class. This means, for imbalanced tasks, ACC is not meaningful. However, rare events are very common in appliance identification or segmentation, and with some exceptions, the default case in appliance usage prediction. For instance, consider a dishwasher: the appliance is commonly used regularly, say every other day, and it takes about 2h to complete its cycle. Thus, the dishwasher is switched off 96% of the total time. Even in scenarios when a dishwasher is used twice as often, i. e., every day, it is still switched off 92% of the time. Considering a classifier that predicts a dishwasher to be ON or OFF, a high accuracy will be obtained by using a hard-coded predictor, that always predicts the appliance as being switched off. Therefore, publications that present an evaluation based on accuracy such as [Pate 07, Srin 06, Jian 12, Ridi 13, Lai 13, Jian 13, Kato 09, Jana 13, Kim 19, Kolt 11, Rein 12] must be treated with caution, as the evaluation may suffer from the accuracy paradox. Despite all this, especially in combination with other performance metrics, accuracy does give insight into an important characteristic.

3.5.2 Precision and Recall

Precision and recall describe the relevance of the classification result. Precision is the ratio of correct positive predictions to total positive predictions. It describes the validity or value of the positive return predictions and is therefore also called *positive predictive value* (PPV).

$$\text{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \quad . \tag{3.31}$$

Recall on the other hand is the ratio of correct positive predictions to the total number of positive samples. Thus it describes the completeness of the prediction. It is also known as *true positive rate* (TPR) or *sensitivity*.

$$\text{recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \quad . \tag{3.32}$$

3.5.3 F_1-score

The F_1-score is defined as the harmonic mean of precision and recall. It is a very common metric for classification problems. The metric is calculated as:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\mathrm{TP}}{\mathrm{TP} + 0.5(\mathrm{FP} - \mathrm{FN})} \quad . \tag{3.33}$$

The interpretation of the F_1-score is less intuitive compared to ACC. The highest possible value is 1, and the lowest is 0. In this definition, precision and recall are valued evenly. Hand et al. criticize the common use of balanced precision and recall since the relative relevance of precision and recall are problem specific [Hand 18]. Similar to the ACC, the ratio between positive and negative samples is not taken into account, thus on imbalanced datasets the F_1-score may be misleading [Chic 20].

3.5.4 Micro and Macro F_1-score

The F_1-score can also be computed for multi-class problems. In this case, either the averages of precision and recall (micro) or the average of the individual F_1-scores (macro) are computed. The value difference between the two can be significant. On imbalanced datasets, the micro-average result can be dominated by the largest class, while the macro-average will give each class equal weight.
The micro F_1-score is defined as:

$$F_{\text{micro}} = 2 \cdot \frac{\text{micro-precision} \cdot \text{micro-recall}}{\text{micro-precision} + \text{micro-recall}} \tag{3.34}$$

where

$$\text{micro-precision} = \frac{\sum_{n=1}^{N} \text{TP}_n}{\sum_{n=1}^{N} (\text{TP}_n + \text{FP}_n)} \quad , \tag{3.35}$$

$$\text{micro-recall} = \frac{\sum_{n=1}^{N} \text{TP}_n}{\sum_{n=1}^{N} (\text{TP}_n + \text{FN}_n)} \quad , \tag{3.36}$$

N the number of classes and TP_n, FP_n and FN_n the result of each class n. In multi-class problems, however, each misclassification is with respect to one class an FP, and with respect to another class an FN [Zhan 15]. Therefore, the sum of FP and FN is equal:

$$\sum_{n=1}^{N} \text{FP}_n = \sum_{n=1}^{N} \text{FN}_n \quad . \tag{3.37}$$

The implication of this is that micro-precision equals micro-recall and the micro F_1-score can be simplified to:

$$F_{\text{micro}} = \frac{\sum_{n=1}^{N} \text{TP}_n}{\sum_{n=1}^{N} (\text{TP}_n + \text{FP}_n)} = \frac{\sum_{n=1}^{N} \text{TP}_n}{\sum_{n=1}^{N} (\text{TP}_n + \text{FN}_n)} \tag{3.38}$$

For the macro F_1-score, a class-specific F_1-score is calculated independently and the final score is calculated by averaging the individual scores. It is therefore defined as:

$$F_{\text{macro}} = \frac{1}{N} \sum_{n=1}^{N} F_n \tag{3.39}$$

where F_n is the F_1-score of each class.

3.5.5 Matthews Correlation Coefficient (MCC)

The Matthews Correlation Coefficient (MCC) is a balanced metric taking TP, TN, FP, and FN equally into account. It is defined as:

$$\text{MCC} = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} \; . \tag{3.40}$$

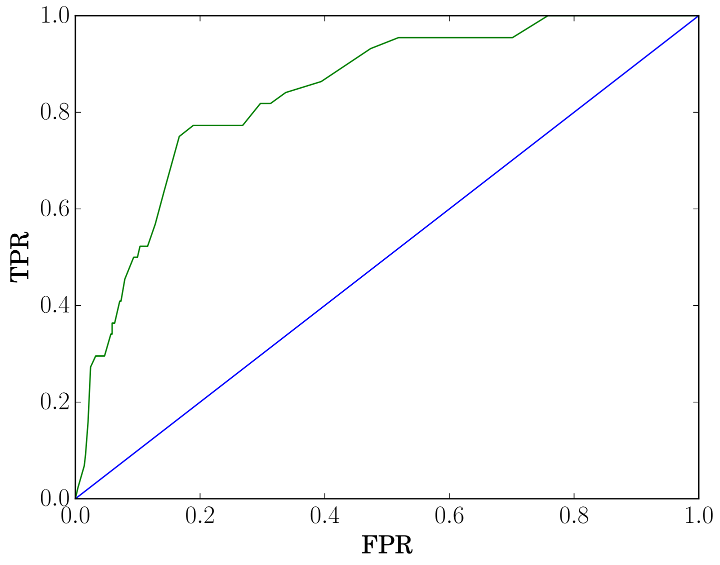


Figure 3.7: Example ROC curve, plotting the FPR against TPR. The blue curve shows the result of a random classifier. The green curve is an example classifier, performing better than random.

The MCC results in a value between -1 and 1, where -1 is the worst value and 1 the best. A total random classifier will have MCC $= 0$.

Unlike ACC and F_1-score, the MCC produces a reliable result even for imbalanced datasets [Chic 20]. This means, that a classifier will obtain a high score by predicting negative and positive cases equally well, even on imbalanced datasets. The MCC is undefined when one of the classes, positive or negative, is empty. This can be overcome by substituting the missing values with an arbitrarily small value.

3.5.6 Receiver Operating Characteristic (ROC) Curve

The Receiver Operating Characteristic (ROC) curve is created by plotting the false positive rate (FPR) against the true positive rate (TPR) at different thresholds. For a threshold-dependent classification approach, the result will vary depending on the selected threshold. When for example using a simple, single threshold-based appliance segmentation approach, a small threshold will easily capture all positive cases where the appliance is switched ON, but it will at the same time produce many false positives. Increasing the threshold will now decrease the FPR while decreasing the TPR. All previously mentioned metrics would only measure the performance at a single point on the ROC curve, at a single threshold. Figure 3.7 shows an example ROC curve.

A single value metric that tries to capture the quality of the whole ROC curve can be calculated as the Area under the Curve (AUC). It is defined as:

$$\mathrm{AUC} = \int_0^1 \mathrm{ROC}\ \mathrm{dFPR} \quad . \tag{3.41}$$

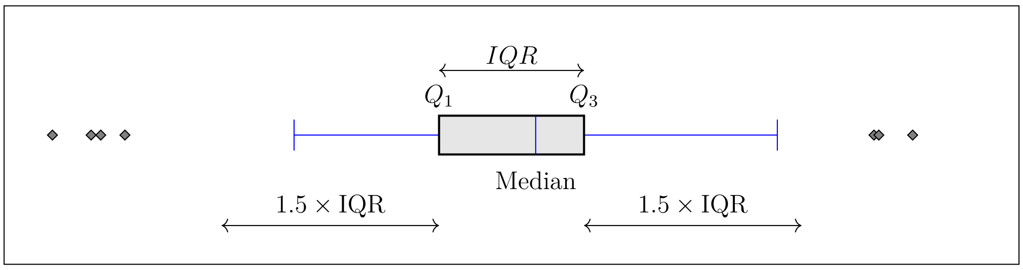


Figure 3.8: Example boxplot where Q_1 and Q_3 are the boundaries of the IQR and the vertical bar is the median. The whiskers mark the range of data within $1.5 \times$ IQR distance. The rhombus-shaped dots indicate outlier data points, which are outside the $1.5 \times$ IQR distance.

The AUC is in the interval $[0, 1]$, where 0 is the worst, 0.5 is the result of a total random classifier, and 1 is the best.

3.5.7 Interquartile Range (IQR)

The interquartile range (IQR) is a measure of the variability of data [Dekk 05]. In the case of evaluations, this data can be a performance measure such as an F_1-score or accuracy. The data is divided into quartiles, providing a five-number summary of the data denoted as Q_0 to Q_4: minimum (Q_0), maximum (Q_4), median (Q_2), and the 25th (Q_1) and 75th (Q_3) percentiles. Therefore, 25% of the data are smaller or equal to Q_1, 50% of the data are smaller or equal to Q_2, and 75% of the data are smaller or equal to Q_3. The range between Q_1 and Q_3 is known as the IQR:

$$\text{IQR} = Q_3 - Q_1 \quad . \tag{3.42}$$

A boxplot, as shown in Fig. 3.8, is a combined visualization of the data distribution based on these quartiles. Q_1, Q_3 are visualized as a box, and the median (Q_2) is marked as a vertical bar. The data points that are within $1.5 \times$ IQR distance to the left or right of Q_1 and Q_3 are marked by whiskers. Points outside this area are seen as outliers and marked using rhombus-shaped dots.

CHAPTER 4

Principles of Machine Learning in Demand Response

Demand Response creates a decision-making challenge for residents where for every usage of an appliance, the electrical load and the electricity price need to be estimated to make a cost versus need decision (cf. Fig. 4.1).

In an ideal and oversimplified scenario, the optimum Demand Response (DR) policy is: to run the appliance whenever the electricity price is lowest.

This policy no longer works when the electricity price changes during a single appliance usage. As a result, the optimal time to start an appliance depends on multiple pricing steps. Additionally, appliances do not necessarily have a constant load profile, so mapping the load profile and electricity price can be a complex decision. Delaying appliance usage comes with discomfort as it interferes with the resident's daily life and habits. This discomfort is intrinsic to each household, resident, and appliance, and must be considered each time an appliance is used.

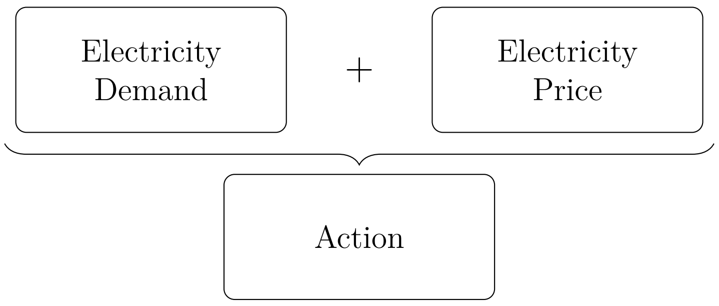


Figure 4.1: Simple illustration of the Demand Response process. The action taken by a resident is based on the demand/need to use an appliance and the current electricity price.

This complex system is modeled by [ONei 10] as an optimal control system containing a number of appliances $m \in \{1, \dots, M\}$. The system state is defined by

$$\boldsymbol{\Omega}(t) = [\boldsymbol{x}(t)^T, \boldsymbol{y}(t)^T, \boldsymbol{z}(t)^T, p(t)]^T, \qquad \boldsymbol{\Omega} \in \mathbb{R}_+^{3M+1}, \tag{4.1}$$

where $\boldsymbol{x}(t)$ is the pending backlog of required electricity, $\boldsymbol{y}(t)$ the average pending workload, $\boldsymbol{z}(t)$ a vector describing the residents' intention/demand to use an appliance, and $p(t)$ the dynamic electricity price.

The vector $\boldsymbol{z}(t)$ only captures the residents' demand on an appliance. The postponed start is then managed by a control policy and captured in $\boldsymbol{u}(t)$, which is the electricity allocated to the appliances at time t.

The residents' demand at time t is modeled as

$$\begin{aligned} z_m(t) &= \gamma_m(t - t_0) && \text{Appliance demanded,} \\ z_m(t) &= 0 && \text{Appliance NOT demanded,} \end{aligned}$$

where $\gamma_m(\tau)$ is an appliance's electrical load profile at time τ, defined by the time difference between the given time t and the appliance's start time t_0.

The pending electricity backlog is written as

$$\boldsymbol{x}(t+1) = \boldsymbol{x}(t) + \boldsymbol{z}(t) - \boldsymbol{u}(t), \quad \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t) \in \mathbb{R}_+^M, \tag{4.2}$$

where $\boldsymbol{u}(t) \leq \boldsymbol{x}(t)$.

At time t, the residents' intention/demand $\boldsymbol{z}(t)$ and the electricity price $p(t)$ is known, and the allocated energy (action) $\boldsymbol{u}(t)$ is determined by a control policy with the goal to minimize financial cost while minimizing the residents' discomfort.

The financial cost at time t is composed of the allocated electricity $\boldsymbol{u}(t)$ multiplied by the current electricity price $p(t)$:

$$\sum_{m=1}^{M} p(t) u_m(t) \quad . \tag{4.3}$$

A resident's dissatisfaction, caused by delaying appliance usage, can be expressed as the negative of what is known in the literature as utility functions. Utility functions are a mathematical abstraction to model a customer's preference for owning or using a product. The negative utility, here called the dis-utility $\bar{\boldsymbol{U}}$, is used to define the total cost of delaying an appliance usage:

$$\sum_{m=1}^{M} \bar{U}_m(y_m(t)) \quad , \tag{4.4}$$

where y_m is a function that models the resident's dissatisfaction caused by a delay. As an example, y_m can be defined as the average pending workload of the system, meaning that the resident's satisfaction is driven by appliances to operate sooner rather than later. This can be defined as:

$$y_m(t+1) = \theta_m y_m(t) + (1 - \theta_m) x_m(t) \quad , \tag{4.5}$$

where $0 < \theta_m < 1$ defines smoothing preferences over time. When $\theta_m \approx 1$, residents are sensitive to the average pending workload, and when $\theta_m \approx 0$ they are sensitive to the pending backlog of required electricity. A high θ_m, therefore, expresses the resident's preference to lower dissatisfaction in the long run.

The sum of (4.3) and (4.4) defines the cost of DR for a given state $\boldsymbol{\Omega}(t)$ as

$$\Phi(\boldsymbol{\Omega}(t), \boldsymbol{u}(t)) = \sum_{m=1}^{M} (p(t)u_m(t) + \lambda \bar{U}_m(y_m(t))), \tag{4.6}$$

where $\lambda \geq 0$ is used to put a price on the dis-utility, helping to determine the trade-off between financial cost and dis-utility.

In the simplest implementation, this control process is directly managed by the residents by minimizing (4.6). The required willingness for this optimization task increases proportionally with the complexity of the system state $\boldsymbol{\Omega}$. A complexity reduction of any part of the system, therefore, has the potential to influence the consumer's willingness to participate. As a consequence, it is desirable that the underlying DR control process is at least assisted, or at best fully operated by machine learning algorithms, reducing complexity, and possibly increasing willingness to participate. This can be achieved by automating the following processes:

- Identifying appliance m (type or model),
- determining appliance load profiles γ,
- determining boundaries of load shifting, modeled as the dis-utility $\bar{\boldsymbol{U}}$,
- predicting appliance usage $\boldsymbol{z}(t)$,
- estimating total cost $\Phi(\boldsymbol{\Omega}(t), \boldsymbol{u}(t))$ and providing recommendations or automatically performing actions.

4.1 Data

A data source for DR is everything that helps to understand the demand, behavior, and needs of a household. The source of these data can be anything from geographic features, standard load profiles or weather data, to high-resolution electricity monitoring using smart meters. These data can be roughly categorized as *microscopic* and *macroscopic.*

4.1.1 Macroscopic Data

These do not directly relate to the individual household and rather provide insight into the residential sector in general. According to [Swan 09], such data "include macroeconomic indicators (gross domestic product (GDP), employment rates, and price indices), climatic conditions, housing construction/demolition rates, and estimates of appliance ownership and number of units in the residential sector". Standard load profiles such as H0 (see Fig. 2.1) are also classified as macroscopic data [Bitt 99].

4.1.2 Microscopic Data

These have a direct relation to the household and residents, providing direct electricity consumption information. Based on microscopic data, gaining insight into the residents' usage habits can lead to an explicit understanding of a user's decision. The primary source of microscopic data is high-resolution electricity monitoring equipment such as smart meters. Smart meters are being installed throughout the EU, as a result of the EU directive 2009/72/EC [EU D 19], increasing the availability of higher-resolution electricity consumption data in the near future.

Electricity load monitoring or sometimes called Appliance Load Monitoring (ALM) can be divided into two main categories: Intrusive Load Monitoring (ILM) and Non-Intrusive Load Monitoring (NILM). NILM was first described in 1985 by George Hart where he referred to it as Non-Intrusive Appliance Load Monitor (NALM) [Hart 85, Hart 92]. The main difference between ILM and NILM lies within the number of metering points used to monitor a household's loads and therefore also within the level of detail provided by the metering operation. NILM uses only a single metering point, thus relying on decomposing the aggregated load into its components through mathematical algorithms. In this regard, the single metering point refers to a single location within the house and not a single power line, therefore in the case of a three-phase installation, the phases may still be monitored separately. Non-intrusive means that no extra equipment is installed in the house, as opposed to ILM, where "intrusive means that the meter is located in the habitation" [Ridi 14]. By these definitions, ILM is less strictly defined. The depth at which additional meters are installed may vary from per room to per appliance, meaning that decomposition may still be required. More detailed ILM subcategories are proposed by [Ridi 14], where ILM 1 refers to zones, ILM 2 to plug level, and ILM 3 to appliance level metering. The mathematical decomposition of NILM and ILM 1/2 data into the individual loads, also called disaggregation, is a nontrivial task and still an active research field. The disaggregation problem is the decomposition of the total load $L(t)$ into the sub loads $l_m(t)$ and is expressed as:

$$L(t) = \sum_{m=1}^{M} l_m(t) \quad . \tag{4.7}$$

Figure 4.2 shows an example of aggregated loads $L(t)$ and the attribution to each individual appliance m.

The data that can be utilized are not restricted to load monitoring data (values over time), but also additional data explaining the circumstances under which the data were collected. Such secondary sources can be dwelling type, occupancy level, household income, and education level [Tasc 18]. Furthermore, for many tasks, the raw load monitoring data is enriched with expert knowledge in the form of additional data points that provide the base for specific tasks or simply help to clean the data.

In the household electricity domain, additional data are:

- Appliance labels: Model name, number, category.
- Data collection setup: Equipment, sensor placement.
- Demographic data: Location, inhabitants.

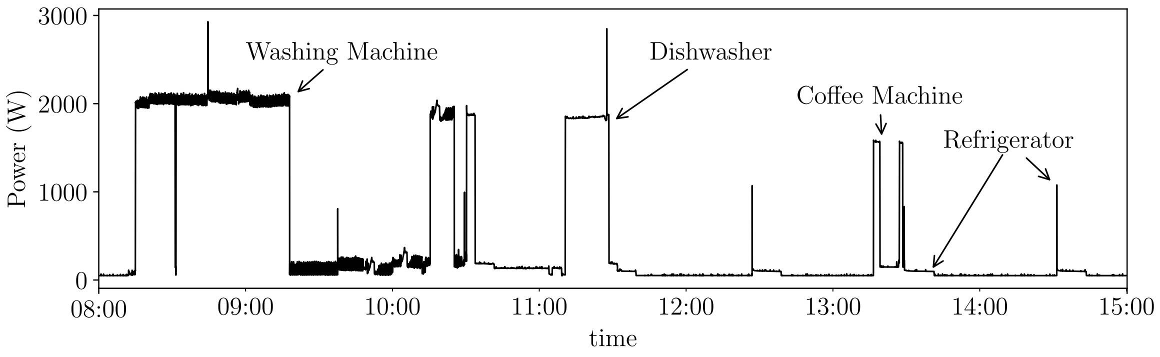


Figure 4.2: Smart-meter measurement accumulating electricity demand of refrigerator, dishwasher, washing machine, and coffee machine. The total household load is the sum of all active individual appliances. Sample measurements are taken from the Domestic Energy Demand Dataset of Individual Appliances in Germany (DEDDIAG) dataset [Wenn 21b].

- Segmentation labels: Occupancy, ON/OFF switching event.
- Error labels: Explanation of errors in the recorded data.

4.1.3 Public Datasets

The most significant enablers of machine learning research are available datasets. Even more important are publicly available datasets, since only with this availability can different research teams evaluate publications and develop established techniques. Since the early 1990s, a large number of public datasets in all kinds of domains have been released. Inspired by the success of datasets in image recognition domains such as MNIST or PASCAL, in 2011 [Kolt 11] published the first dataset for energy disaggregation. At the time [Kolt 11] argue, that "although there are vast amounts of data relevant to energy domains" ... "the majority of this data is unavailable to researcher". Furthermore, the authors argue that many other domains have greatly benefited from public benchmark datasets such as MNIST [Lecu 98] for handwritten digit recognition or PASCAL [Ever 10] for visual object category recognition and detection.

Since then, more and more datasets have been released with different target energy applications in mind. Table 4.1 provides an overview of available datasets in the year 2021. The datasets can in general be differentiated by:

- location
- count
- duration
- sample frequency
- level at which data was collected (whole house, rooms, plugs, appliances)
- additional information provided.

Table 4.1: Public appliance and whole-house level energy consumption datasets.

Data set	Reference	Location	Duration per house	Number of houses	Appliance sample intervall	Aggregate sample intervall	Citations
REDD	[Kolt 11]	MA, USA	3–19 days	6	3 sec	1 Hz & 15 kHz	898
Smart*	[Bark 12]	MA, USA	2–3 years	7 / 114	60sec / –	60sec / 15min	341
Tracebase	[Rein 12]	–	24h	–	3s	–	184
BLUED	[Ande 12b]	USA	7 days	1	12 kHz	1 kHz	52
iAWE	[Batr 13]	New Delhi, India	73 days	1	1 Hz	1 Hz	107
ACS-F1	[Gisl 13]	–	1h	–	10 sec	–	–
COMBED	[Batr 14b]	Delhi, India	2 years	2	30s	30s	89
Dataport	[Peca 14]	TX, USA	0–2.75 years	824	1 min	1 min	–
ECO	[Beck 14]	Switzerland	8 months	6	1 sec	1 sec	180
GREEND	[Mona 14]	Italy & Austria	8 months	8	–	1 Hz	109
DRED	[Utta 15]	Netherlands	2 months	1	1 Hz	1 Hz	53
REFIT	[Murr 15]	UK	20	1 year	8s	8s	55
UK-DALE	[Kell 15]	London, UK	3–26 months	5	6 sec	1–6 sec & 16 kHz	264
AMPds v2	[Mako 16]	BC, Canada	2 years	1	1 min	1 min	162
WHITEDv1.1	[Kahl 16]	Austria, Germany, Indonesia	–	–	44.1 kHz	–	–
COOLL	[Pico 16]	Orléans, France	0–19ms	–	100 kHz	–	58
RAE	[Mako 17]	Vancouver, Canada	72 days	2	1 Hz	1 Hz	10
BLOND	[Krie 17]	Munich, Germany	213 days / 50 days	1	50 kHz / 250 kHz	6.4 kHz / 50 kHz	54
PLAID 2017	[Gao 14, De B 20]	USA	55	–	30 kHz	–	–
IDEAL	[Pull 21]	UK	286 days	255	5 sec Hz	1 Hz,	2
DEDDIAG	[Wenn 21b]	Germany	1–3.5 years	15	1 Hz	1 Hz	–

A review on electricity datasets highlights the diversity of the datasets as they differ in location, duration, frequency, file format, and several other aspects [Klem 19]. The authors of the aforementioned publication suggest two primary objectives for the collection and provisioning of datasets, (1) interoperability and (2) comparability. This would help to decrease heterogeneity, thus increasing the comparability of publications. An aspect highlighted by the authors is missing metadata describing the circumstances of the data recording. This is especially relevant for it to be possible to find explanations for behavior and behavior changes.

Based on citation count, REDD is the most prominent and often used dataset. The dataset offers recording of up to 19 days of 6 houses with a varying number of appliances [Kolt 11]. The dataset offers single appliance measurements of 1 Hz; the overall house measurements of the two circuits are 15 kHz. The dataset was developed predominantly for the disaggregation task and because of the short time span it can in all likelihood be used only for that purpose. Tasks such as user behavior analysis require measurements over a much greater time span because most devices (e. g., dishwasher or washing machine) are not often used. The Smart* dataset does cover a longer time span, but with a period of 3 months, it also does not provide many samples [Bark 12]. The AMPds 2 dataset does provide 2 years of measurements, but only samples at 1/60 Hz, which is enough for behavior analysis, but currently not for disaggregation tasks [Mako 16]. ECO and GREEND both provide an acceptable amount of data, but a short evaluation of the datasets reveals very low quality as a result of long, undocumented interruptions, which is not ideal for usage prediction, behavior analysis, or similar tasks where high-quality measurements are required [Beck 14, Mona 14].

In this thesis, a new public dataset called DEDDIAG is presented in Chapter 5. The dataset has previously been described in the following published manuscript: [Wenn 21b].

4.2 Knowledge

Knowledge is retrieved from data using a *bottom-up* or *top-down* approach, depending on the data source being *microscopic* or *macroscopic* [Swan 09]. These terms are used in literature in order to group data processing methods. Methods that make use of both, microscopic and macroscopic data, are called hybrid [Proe 21].

4.2.1 General Approaches

Top-Down

Top-down approaches rely on macroscopic data, requiring only aggregated data that are more readily available and are usually less sensitive in terms of the resident's privacy. In top-down approaches, the electricity demand is primarily modeled to obtain requirements for the supply side, where the individual household is less important, modeling long-term changes and transitions rather than individual usage patterns [Swan 09].

The total residential sector electricity demand and the characteristics of dwellings, age, sex, income, level of education, and family constellations within a sector are used to derive profiles by which individual households can be categorized. Since this approach relies on historical data, these models have “no inherent capability to model discontinuous advances in technology” and further the “lack of detail regarding the energy consumption of individual end-uses eliminates the capability of identifying key areas for improvements for the reduction of energy consumption” [Swan 09].

The following six steps are described in [Proe 21] as the common procedure to develop a top-down model:

- **Step 1:** Find a historical electricity dataset of a sufficient sampling rate.
- **Step 2:** Determine the macroscopic data needed.
- **Step 3:** Cluster different combinations of data by available dimensions.
- **Step 4:** Determine stochastic predictors based on time-series analysis of the dataset.
- **Step 5:** Combine stochastic predictors and clusters.
- **Step 6:** Validate the model.

Bottom-Up

Bottom-up approaches rely on microscopic data and have the potential to provide a much more detailed model of a household’s demand as electricity consumption is closely monitored and does not solely rely on historical data. Bottom-up methods “calculate the individual dwelling energy or electricity consumption and extrapolate these results over a target area or region” [Swan 09]. Such methods follow the DR system understanding described by [ONei 10], where individual appliances m and load profiles γ_m have to be determined. Further, the residents’ behavior patterns, i. e., the relation between appliances’ usages, are combined with the load patterns to compute a household’s electricity demand pattern. This results in a very detailed and household-specific profile, where the individual contribution of each appliance to the total electricity demand is known. The main advantage of the bottom-up model is that it provides a very detailed analysis of the households, permitting the model to account for individual differences. The main disadvantages are that these models require very detailed data about each household and that they are computationally heavy as they need to be calculated for each household individually. Regarding privacy, the answer is two-headed: while in cases where the household-specific, high-resolution data are processed on-premises, the privacy of the household is well respected; data remain a high privacy concern, however, in many cases, they have to be transferred to a data-center.

The following five steps are described by [Proe 21] as the common procedures to develop a bottom-up model:

- **Step 1:** Determine the model’s variables such as appliances m and load profile γ_m.

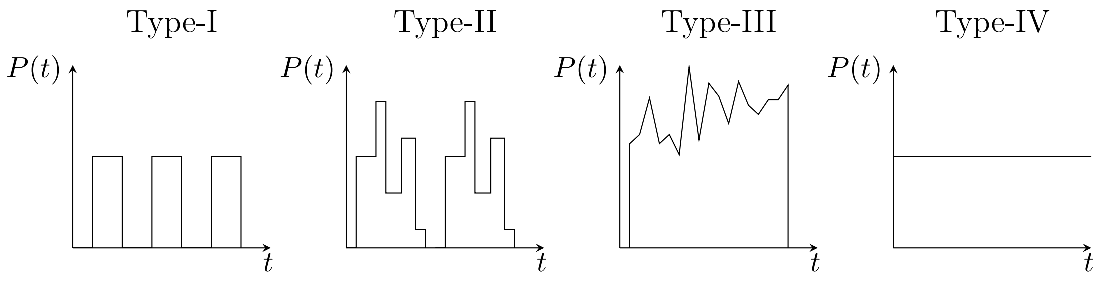


Figure 4.3: Example power plots of the four appliance categories Type-I–IV: On-Off, Finite State Machines, Continuously Variable, Permanent On.

- **Step 2:** Determine each appliance's usage pattern from historical data.
- **Step 3:** Generate the individual load profile for each appliance.
- **Step 4:** Aggregate the individual load profiles from single or multiple households.
- **Step 5:** Validate the model.

The appliances' load profiles can be categorized by the operational state and who controls the appliance (Active vs. Autonomous Control) [Hart 92, Bara 03]. The load patterns, also called signatures, are composed of operational states that can be categorized into four types. Type-I, II, and III have been described by [Hart 92], and Type-IV was added by [Bara 03]. Examples are shown in Table 4.2.

- **Type-I: On-Off**
  These are appliances with only two operational states (ON/OFF). In these two states, the load is constant and can be described by a single repeating pattern.
- **Type-II: Finite State Machines**
  Appliances with multiple operational states can be described by a combination of multiple patterns. The number of operational states and combinations of patterns is finite and can be described by a Finite State Machine (FSM). The switching pattern of these appliances is repeatable and forms a start-stop cycle.
- **Type-III: Continuously Variable**
  These appliances do not have a fixed number of states and the load pattern varies constantly. There is no relation between loads and states of the appliance making the identification and disaggregation very challenging.
- **Type-IV: Permanent On**
  Appliances that have a permanent load and are switched ON over several days or months.

The difficulty of knowledge extraction heavily depends on the appliance operation states, since only appliances of Type-I, II, and IV follow recurring patterns. Type-IV is always switched ON and therefore not relevant for load shifting in DR, leaving Type-I & II as the appliances relevant for DR.

Table 4.2: Classification of various household appliances [Hart 92, Bara 03].

Type	Active Control	Autonomous Control
I	Lamps, Kitchen Appliances, Hair-Dryer	Fridges, freezers
II	Washing Machines, Dryers, Dish Washers	Ventilation
III	Lighting with Dimmers, Office Desks	–
IV	Manual Air Conditioning, Routers	Automatic Air Conditioning

Hybrid

Hybrid methods try to combine both microscopic and macroscopic data, and therefore combine techniques used in both bottom-up and top-down approaches. Combining bottom-up knowledge about a household's appliances, load demand profiles, and usage habits with top-down knowledge from clustering households allows hybrid methods to utilize more available data [John 14]. As demonstrated by [Zhon 16], such algorithms can overcome the bias of load profile clustering as they facilitate identifying unusual groups of customers that share spikes in their electricity demand profile. Their model combines smart-meter data from several households and identifies several clusters including outliers.

The following five steps are described by [Proe 21] as the common procedure to develop a hybrid model:

- **Step 1:** Determine the micro- and macroscopic data to be used.
- **Step 2:** Execute step 1–3 of the bottom-up approach.
- **Step 3:** Execute step 1–4 of the top-down approach.
- **Step 4:** Combine the results to generate demand load profiles.
- **Step 5:** Validate the model.

4.2.2 Features

The knowledge extraction is done on derived values (features) or certain segments of the raw signal. Feature extractions depend on the available data and the methods used. The typical raw measurements provided by a metering device are voltage, current, and power recorded over time. Depending on the sample rate, different properties of the signal can be utilized, as appliances produce features at different detail levels.

Current and Voltage

Modern metering devices record current and voltage and derive all further values such as power from it. The voltage does not depend on the measured appliance and is a property of the network. Since the electrical power is an AC (alternating current) circuit, the voltage and current are sinusoidal and can be expressed as:

$$V(t) = V_{\max} \sin(\omega t + \phi_v) \quad , \tag{4.8}$$

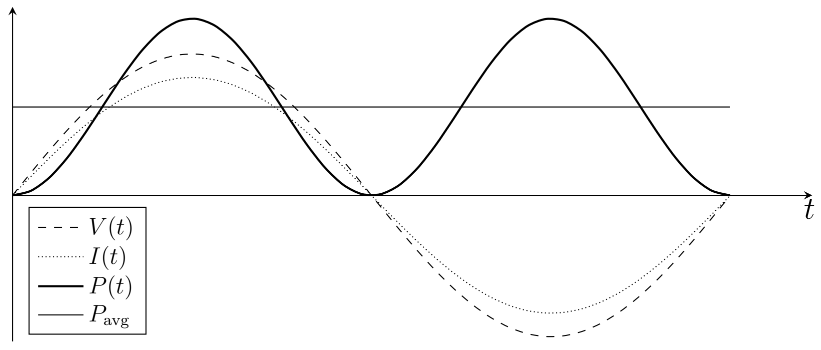


Figure 4.4: Plot of power when voltage and current are in phase. Y-axes show the power P as the product of voltage V and current I and the resulting average power P_{avg}.

$$I(t) = I_{\max} \sin(\omega t + \phi_I) \quad , \tag{4.9}$$

where $V(t)$ and $I(t)$ are the voltage and current at given time t, $V_{\max}$ and $I_{\max}$ the maximum voltage and current, ω the angular frequency, and ϕ_v and ϕ_I the phase shift. The angular frequency ω is defined as radians per seconds, where for 60 Hz it is given as $\omega = 2\pi \cdot 60\,\text{Hz} = 377\,\text{rad}/s$. In practice, the current and voltage waveforms do not conform to this precise mathematical definition as shown in Fig. 4.4 [Meie 06]. While the waveform will be periodic and have periodic zero crossings, the shape is not round and smooth. A phase shift between voltage and current may also occur. This deviation from a perfect, single frequency, sinusoidal function is a property that can be used for appliance identification and is discussed in Section 4.2.2.

It is common to describe a network's voltage as the root-mean-square (RMS) value and not as the sinusoidal function. In most countries, the RMS voltage fluctuates around 230 V, and e. g., in the United States of America at 110 V. Fluctuations in the voltage signal's RMS value is caused by load change on the network. The RMS of voltage and current can be calculated by taking the RMS of (4.8) and (4.9), which reduces to:

$$V_{rms} = \frac{1}{\sqrt{2}} V_{max} \quad , \tag{4.10}$$

$$I_{rms} = \frac{1}{\sqrt{2}} I_{max} \quad . \tag{4.11}$$

On the other hand, the measured current depends on the recorded appliance. Since the required power of an appliance is dependent on current and voltage, the current will increase with decreasing voltage. The *instantaneous power* is defined by:

$$P(t) = I(t)V(t) \quad . \tag{4.12}$$

This product of voltage and current is uncommon to use, and it is more common to describe the *average power* as the product of the RMS values [Meie 06]:

$$P_{\text{avg}} = I_{\text{rms}} V_{\text{rms}} \quad . \tag{4.13}$$

This equation is only true for the resistive case, where current and voltage are not shifted in time and their maxima coincide, thus only describing an idealized scenario.

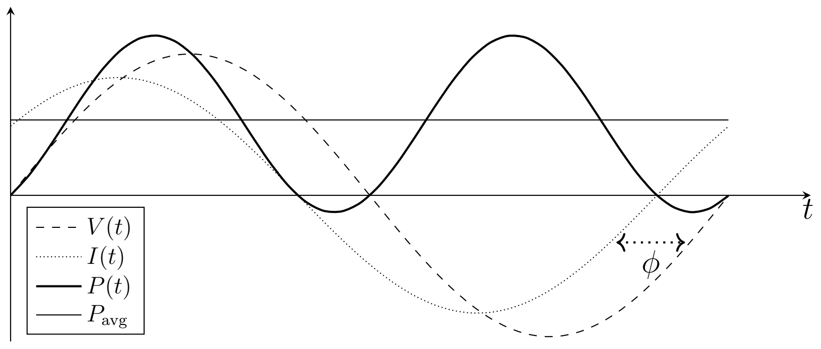


Figure 4.5: Plot of power when current lags behind voltage by the phase angle ϕ. Y-axes show the power P as the product of voltage V and current I and the resulting average power P_{avg}.

When current and voltage are shifted in time ($\phi_I \neq \phi_V$), this equation will calculate the so called *apparent power* (S):

$$S = I_{\text{rms}} V_{\text{rms}} \quad . \tag{4.14}$$

When using current and voltage as a feature, the sampling rate must be high enough to quantify a possible phase shift, or it must be acknowledged that I_{rms} and V_{rms} may be subject to a phase shift and their product does not represent an appliance's consumed power. Nonetheless, the properties provided by instantaneous voltage and current or their RMS values provide rich, appliance-specific features that can be exploited for the appliance identification task. Voltage has been used by [Kato 09, Meeh 12, Weis 12, Kahl 19, Berg 10, Faus 20, Faus 21, Kolt 11, De B 18b] and current by [Kato 09, Meeh 12, Weis 12, Kahl 19, Berg 10, Lai 13, Jana 13, Faus 20, Faus 21, Kolt 11, De B 18b].

Real and Reactive Power

When voltage and current are shifted in time as shown in Fig. 4.5, the average power is defined by:

$$P_{\text{avg}} = I_{\text{rms}} V_{\text{rms}} \cos\phi \quad , \tag{4.15}$$

where ϕ is the angle of the phase shift between voltage and current. This average power, is an appliance's actual transmitted and consumed power and is also known as *real power* or *active power*. The counterpart of real power (P) is the *reactive power* (Q) which is expressed as:

$$Q = I_{\text{rms}} V_{\text{rms}} \sin\phi \quad . \tag{4.16}$$

The relation between P, Q and S can be illustrated as a power triangle as shown in Fig. 4.6.

Real power (P) and/or reactive power (Q) consumption are common feature used for appliance identification [Abub 16]. PQ are usually used when the appliance is in a steady state, facing the challenge that the PQ signature of different appliances is less distinct as the signature is very similar in a steady state. P and/or Q has been used by [Rein 12, Ridi 13, Weis 12, Zaid 10, Wenn 19b, Figu 11, Giri 13]. Following the appliance Types-I–IV described in Section 4.2.1 (On-Off, Finite State Machines,

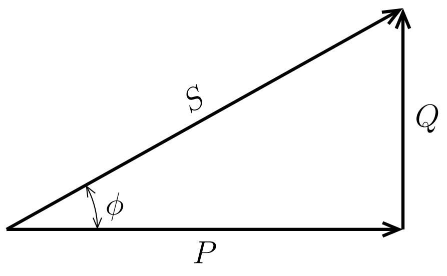


Figure 4.6: Power triangle showing relation of apparent power S, real power P and reactive power Q.

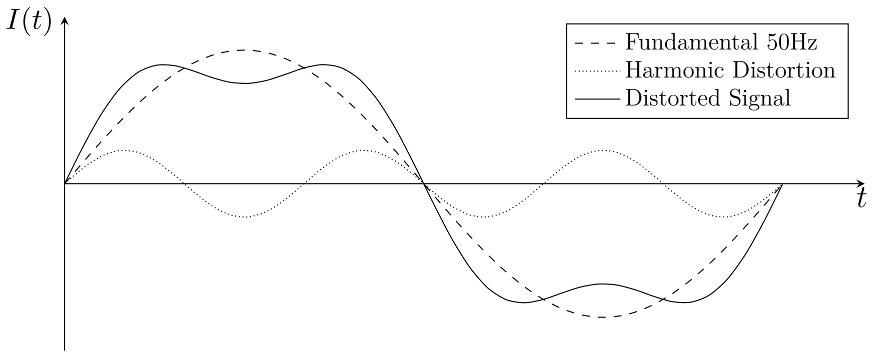


Figure 4.7: HAR in a 50 Hz current signal. The distorted measured signal is a combination of the fundamental 50 Hz signal and the introduced distortion.

Continuously Variable, Permanent On), the available features based on PQ are different. On-Off appliances have a certain step-like change in electricity demand. When the appliances are switched on, there is insubstantial information on the appliances' power consumption. Type-II (Finite State Machines) appliances on the other hand, will produce a signature when being switched on as well as having a certain pattern or power changes during their operation. The PQ features of Type-III (Continuously Variable) appliances do not have a certain level when being switched on and also do not follow any recursive pattern. As Type-IV (Permanent On) appliances are always switched on, no switching related PQ changes are available.

Harmonic Distortion

When high sample rate data is available, the harmonics in a power system can be exploited. Appliances, that draw abrupt short current pulses rather than smooth sinusoidal currents provide an exploitable signature. These features are known as harmonic distortion (HAR) [Feng 13, Meeh 12, Srin 06, Hail 96, Jana 13]. An example for a HAR in a current signal is shown in Fig. 4.7. It shows the fundamental 50 Hz sine wave, as well as the distortion signal and the resulting distorted current signal. When the current and voltage waveforms oscillate proportionally the load is linear, otherwise the load is non-linear. The distortion phenomena of non-linear loads can be described using harmonic analysis.

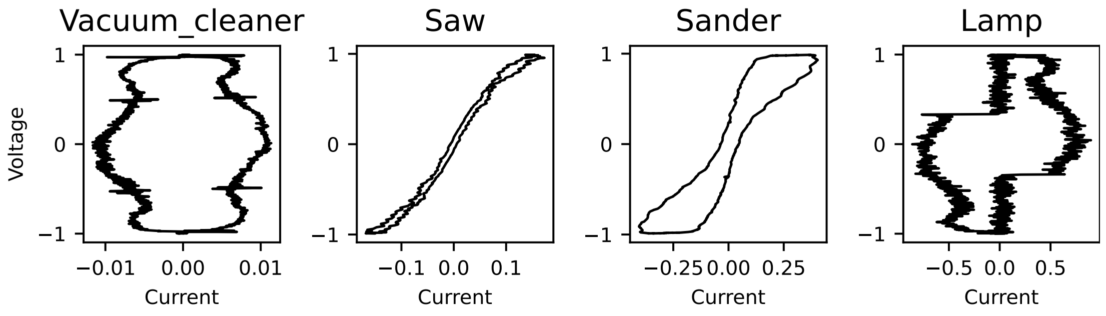


Figure 4.8: Standard V-I trajectory plots where current and voltage are plotted against each other. Examples are taken from the COOLL dataset. The data are scaled to range −1 to +1.

V-I Trajectory

[Lam 07] describe another high sample rate features known as voltage-current (V-I) trajectory. The V-I trajectory is a 2-dimensional feature containing one normalized wave cycle of voltage and current. Voltage and current are usually plotted on two axes as shown in Fig. 4.8. The different shapes of the plots are clearly visible. The mutual locus of the steady state instantaneous voltage and current form provide unique shapes for each appliance. Based on the plotted V-I trajectory feature, 8 different characteristics have been described by Lam et al. [Lam 07]: asymmetry, looping direction, area, the curvature of the mean line, self-intersection, slope of the middle segment, area of left and right segments, and the peak of the middle segment. While such hand-crafted features help to understand the different characteristics, it has been shown that by using a deep neural network the classification of V-I plots can directly be learned by a classification algorithm [Du 16, Gao 15, De B 18a, Faus 20, Faus 21].

4.2.3 Event detection

Next to the feature extraction, event-based algorithms use only certain segments of the signal known as events. Events and their detection are a major topic surrounding appliance identification, segmentation, and usage prediction. The definition of an event in DR, ILM, or NILM context is less clear than anticipated. In literature the terms events, edges, and transient states are to some extent used as synonyms, all describing the segment between steady states or the appliance's ON/OFF switching events. Algorithms that find such segments are called event/edge detection algorithms. Kahl et al. conclude that no generally acknowledged event definition exists, although "The event detection performance depends strongly on the event definition itself" [Kahl 19]. Event detection in the NILM context is seen as a pre-processing step for the classification task. In general, there are two understandings of events that we propose to call *edge events* and *switching events*. Fig. 4.9 shows a dishwasher cycle and the two different types of events.

Edge events are events, that are defined by significant changes in the power signals, the edges. These edges are mentioned in the early publications from Hart where he defines edges as step-like changes in the signal [Hart 92]. This is also the event

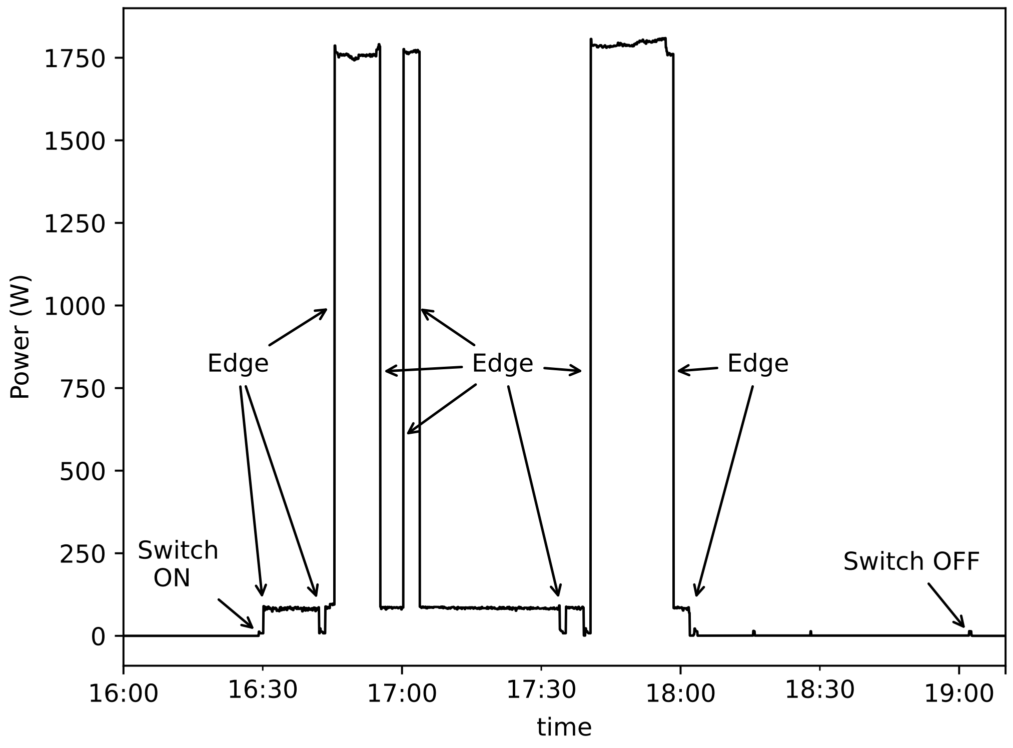


Figure 4.9: Switching and edges events off a dishwasher (appliance 4) from the DEDDIAG dataset.

definition of Wild et al., who defines an event as the “transition from one steady state to another steady state which definitely differs from the previous one” [Wild 15]. These edge-defined events can be detected by applying a threshold filter to the first derivative of the signal [Alca 17].

The second type of events, the switching events, are defined by the switch ON and OFF action. Kahl et al. argue that “appliance ON / OFF events that have a causal origin (i.e., from user interaction or physical appliance state changes) are more relevant than transients that simply satisfy the rule set” [Kahl 19]. Their reasoning is, that users are more interested in appliance cycles such as the start and stop of a washing machine than the temporary increase in energy consumption of a laptop or similar appliances. Anderson et al. describe their event detection algorithm filters as only windows where an appliance’s electricity demand is above a trained threshold [Ande 12a]. Jiang et al. define these events as the transient state between defined appliance states such as ON/OFF [Jian 13]. They describe an edge symbol detector (ESD) algorithm to detect the switch-on and switch-off events.

Edge events are a single point in time and are described by a single timestamp t. Switching events exist as two interconnected ON and OFF event pairs. The switching-event pair describes a time span defined by the lower and upper bounds (t_0, t_1). They follow the natural order of the ON event preceding the OFF event: $t_0 < t_1$. Further, the switching-event pairs describe the start and end of the associated load profile γ.

4.3 Action

Based on the acquired knowledge, personalized recommendations can be formulated and actions scheduled and executed. An ideal recommendation system will minimize the total cost $\Phi(\boldsymbol{\Omega}(t), \boldsymbol{u}(t))$ in (4.6), by balancing electricity cost and the residents' comfort. Actions are understood as the execution of recommendations, where the execution may either be done by a resident or an actuator, or in some cases by both. An example of the participation of both residents and actuator is load shifting of a washing machine, which has to be prepared by the resident and is started by an actuator. Examples of simple recommendations formulated by a load shifting systems proposed by [Fisc 13] are:

- Save £13 by shifting a quarter of your daytime use of washing machines, dishwashers, or tumble dryers to night-times.
- Save £25 by shifting half of your daytime use of washing machine, dishwasher, or tumble dryer to night-times.
- Save £51 by shifting all of your daytime use of washing machine, dishwasher, or tumble dryer to night-times.

The recommendations are presented using a Graphical User Interface (GUI) as part of a walk-through interface that guides the users.

Another system described by [Qayy 15], formulates recommendations based on a total cost function similar to (4.6), while respecting a number of electricity and time constraints. The electricity constraint makes sure that a predefined upper peak limit is not exceeded and that an appliance's load profile is well in line with the dynamic electricity price. The time constraint models uninterruptible operations, constraints between sequential usages where, e. g., a tumble dryer is always used after the washing machine, and lastly respecting the resident's time preferences. Based on this they present the user a list of optimal day-ahead scheduling times for each appliance using 15min slots.

4.4 A Machine Learning Demand Response Model

4.4.1 General Model

Based on the system understanding of the DR process described at the beginning of this Chapter 4, the simple system model in Fig. 4.1 is extended into a more complex model called Machine Learning Demand Response Model (MLDR), first presented here.

The MLDR (cf. Fig. 4.10) is organized in three levels: *data* $\rightarrow$ *knowledge* $\rightarrow$ *action.* It is inspired by the data–information–knowledge–wisdom hierarchy (DIKW) [Rowl 07], a way to model the relationship between data, information, knowledge, and wisdom as a pyramid. Information and knowledge can be understood as: "Information is data processed for a purpose" [Curt 08], "Knowledge is the combination of data and information, to which is added expert opinion, skills, and experience,

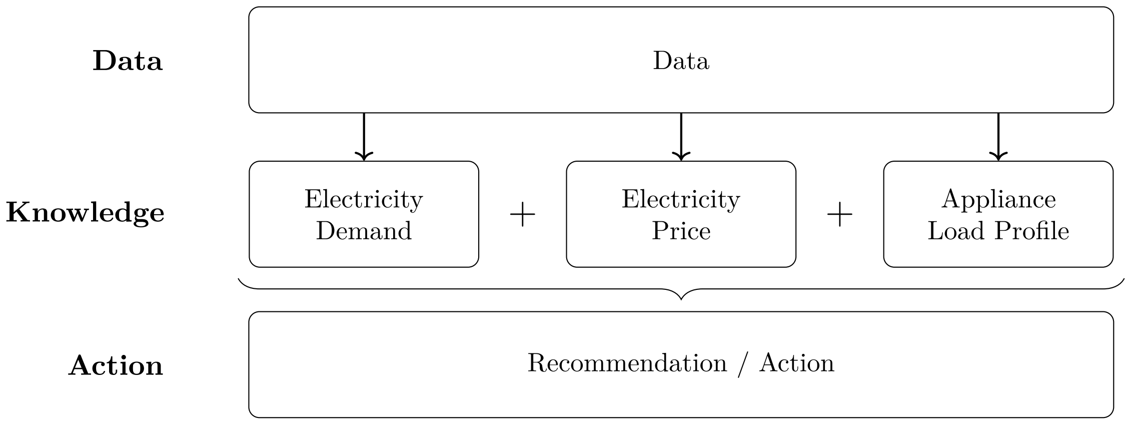


Figure 4.10: MLDR, organized in three levels. The data level describes all available data sources that are used by the knowledge level to formulate actions and recommendations.

to result in a valuable asset which can be used to aid decision-making" [Chaf 11]. MLDR therefore does not distinguish between information, knowledge, and wisdom, as its purpose is to transform data into actions.

4.4.2 Specific Model

The following chapters present a more specific bottom-up DR model for Type-I & II appliances, highlighting the individual machine-learning steps required for knowledge extraction from microscopic data. The first step after data acquisition is the identification of the appliance category or type. This is followed by a segmentation step to find ON/OFF-switching events from which the load pattern and usage pattern can be derived. As shown in Fig. 4.11, these patterns and events are the basis for forecasting and recommendations.

The segmentation of ON/OFF-switching events is often described as a pre-processing step of appliance identification. We deliberately describe this step after the appliance identification, as this is not done in order to simplify the identification, but in order to obtain precise ON/OFF-switching events for each appliance.

The difficulty of knowledge extraction depends on the input data: For ILM 3 data, where the data points are not an aggregate of multiple appliances, each data point belongs to a distinct category, whereas on the other hand, a NILM and ILM 1/2 data point belongs to a set of appliances, thus to a set of categories. When only the aggregated loads are available (NILM and ILM 1/2), the complexity of the MLDR steps increases. Hart describes that NILM (the used abbreviation was NALM) "estimates the number and nature of the individual loads, their individual energy consumption, and other relevant statistics such as time-of-day variations." [Hart 92]. Thus, his understanding of NILM goes beyond the disaggregation task as expressed in (4.7), but also includes further statistics. Following his understanding, NILM is the knowledge extraction from microscopic data using a bottom-up approach where only aggregated loads are present. The NILM process is described by [Abub 16] as three stages: Data Acquisition, Feature Extraction, and Appliance Classification.

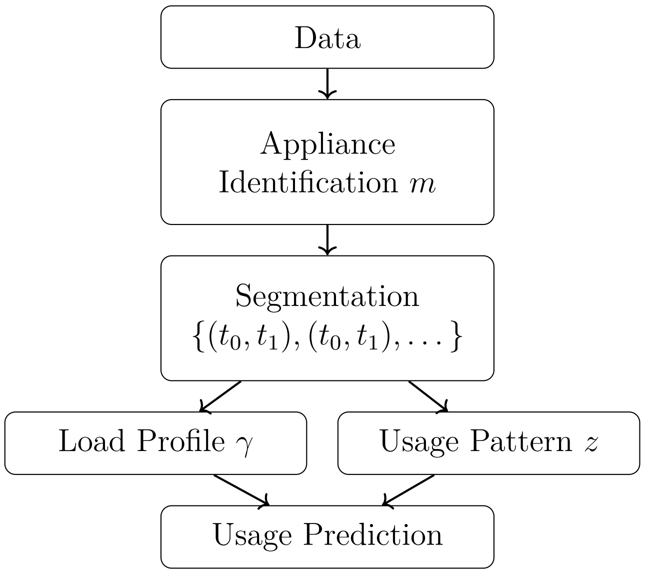

Data
Appliance Identification m
Segmentation $\{(t_0, t_1), (t_0, t_1), \dots\}$
Load Profile γ
Usage Pattern z
Usage Prediction

Figure 4.11: Knowledge extraction steps required for machine learning-based demand response using a bottom-up approach.

Therefore, from a DR perspective, NILM algorithms follow the MLDR steps shown in Fig. 4.11 and require identification of appliances as well as event segmentation.

CHAPTER 5

DEDDIAG Dataset

Substantial parts of this chapter have been published in the following manuscript: [Wenn 21b]. The majority of the manuscript has been authored by the author of this thesis. The main scientific contributions are based on his own work and thoughts.

5.1 Introduction

This chapter describes the creation of a new dataset called Domestic Energy Demand Dataset of Individual Appliances in Germany (DEDDIAG).

The collection of such datasets is no trivial task since the data has to be collected over a long period at high frequency. Data is an integral part of machine learning, required for training in supervised learning and for evaluation of all learning efforts. It, therefore, represents the base of information discovery, providing ground truth. The machine learning community has a history of making Intrusive Load Monitoring (ILM) and Non-Intrusive Load Monitoring (NILM) datasets publicly available for comparability purposes. The release of public datasets embraces the scientific paradigm of "standing on the shoulder of giants" as it is a large enabler for iterative improvements.

5.2 Data Collection System

In the following, an electricity data collection system for single appliances as well as whole-house level is described. The general requirements derive from a 2-year load-shift monitoring project, where homes were monitored. The goal of the project was to record real-world behavior to understand the load-shift potential, and at a later stage provide the homes with RTP (Real-Time-Pricing) incentives for load-shifting. As part of the project, a statistics-based appliance usage prediction algorithm has been developed that will help to reduce the complexity of RTP behavior recommendation systems [Wenn 17], as well as an event segmentation algorithm that helps to identify appliance usage in raw electricity measurements [Wenn 19b].

The full data collection system is published under MIT license and is available under `https://DEDDIAG.github.io`. The dataset itself is published as tab-separated text files, together with code, to import all data into a PostgreSQL instance. An Structured Query Language (SQL) function called `get_measurements()` is provided to get seconds-based measurements, where readings are converted from value-changes to seconds-based readings using interpolation; timestamps are rounded to nearest seconds. There is also a python package available `https://github.com/DEDDIAG/DEDDIAG-loader.git` that assists in retrieving data into a pandas-DataFrame/numpy-array.

5.2.1 Requirements

The collection system was built with a focus on recording data for a user behavior change scenario such as Real-Time-Pricing (RTP). Therefore, the focus lies on recording appliances with the following properties:

- easily shiftable load,
- significant electricity consumption,
- standard power plug.

Appliances falling under these terms are washing machines, dishwashers, fridges, and freezers, thus the datasets will mostly contain these appliances.
The general requirements were:

- collect appliances and, optionally, whole-house electricity usage,
- store data locally,
- upload data frequently to a central server,
- collect over a long time period ($>$ 2 years) at $\approx$ 1 Hz,
- no technical knowledge required to install the system,
- keep costs low.

Hardware Requirements

For a non-technical user to install the hardware and software in their home environment, it should be user-friendly. For this reason, the hardware should be placed in between the wall plug and the appliance to reduce installation complexity. Wireless communication between plugs would lower the complexity of the setup and would reduce the need to install more cables throughout the home. For safety reasons, an electrician would be required to install house-level smart meters, therefore the technical knowledge required could be somewhat increased. The sampling frequency for both appliance-level and home-level meters should be at least 1 Hz, as this rate is being used by a large number of algorithms already, following the suggestion of Klemenjak, et al. [Klem 19] for macroscopic datasets. Higher sample rates increase the data handling complexity as it needs to be kept in mind that doubling the sample rate will also double the data volume that needs to be handled. Collected data should be stored on a small on-premises computer. The overall cost should be kept as low as possible.

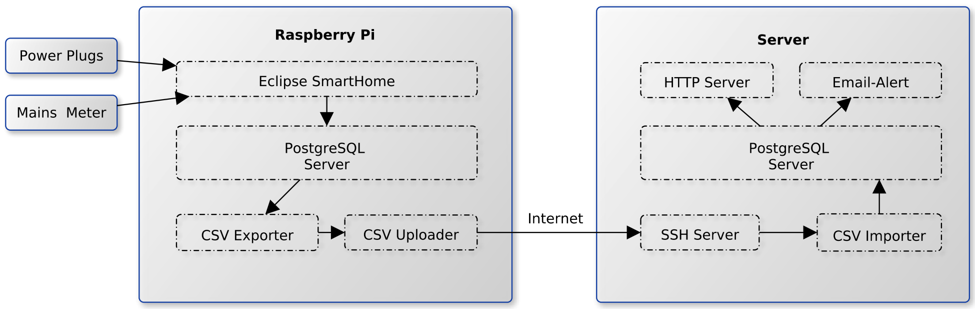


Figure 5.1: Diagram showing detailed client-server components and data flow of the developed domestic energy measurement system. The Raspberry Pi (Client) is installed in the household collecting measurements of the meters. Data is persisted locally and automatically uploaded each hour to a central server using the household's internet connection. [Wenn 21b], License: CC-BY 4.0

Software Development Requirements

The software system would need to be able to read the measurements from appliance- and home meter and store it on an on-premises computer. The users would be provided with a web interface where they can monitor real-time meter values. All local data (client) are to be uploaded hourly to a centralized server over the internet. The uploads should be self-healing, meaning the upload system needs to be robust against a broken internet link, aborted/partial uploads, and server downtime.

With a limited overall time for building and using the monitoring system, the goal was to measure as early in the project as possible, meaning the system was installed in the homes as early as possible. Therefore the system was developed in a Minimum Viable Product (MVP) fashion. As the project team did not have direct physical access to the homes, which were distributed throughout the south of Germany, the software needed to be updated automatically for the shipping of bug fixes and new features. For security reasons it is required to avoid direct network access to the households' network, thus having the system update autonomously by providing a new software version on a server.

5.2.2 System Architecture

Based on the requirements listed above, the following system architecture was designed (cf. Fig. 5.1). All components on the server and client are designed as microservices and run as containers using Docker (`https://www.docker.com`). This allows for a guaranteed software state on each client and also provides a simple update mechanism.

Server-Side

The server side provides three main functionalities: Data storage including secure upload, visualization, provide software updates, and monitoring. For data storage a PostgreSQL (`https://www.postgresql.org`) database is used, an open-source relational database that provides a rich set of query functionality. The measurements

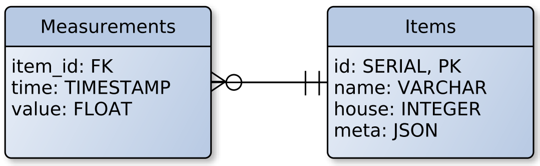


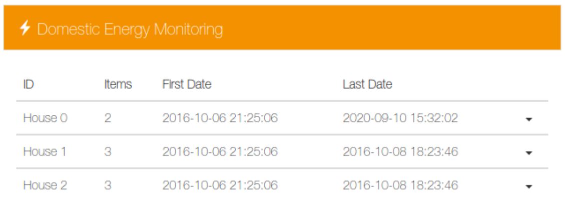
Figure 5.2: Entity relationship model of server-side data storage. [Wenn 21b], License: CC-BY 4.0

of all homes and appliances are stored in a single table. In the following, all measurement sources, i. e. appliance or smart-meter phase, are called item. Each item is identified by a unique ID that is assigned on the first upload of measurements to the server. All measurements are stored in a database scheme as shown in Fig. 5.2, where all measurements are stored in a single table with a foreign key to an item's table. The item's table contains the assigned unique ID as well as metadata such as the item name and which household it is installed in. An alarm system was implemented that informs the residents in case there was no new data uploaded within the last 6 hours.

A visualization was implemented as a responsive Graphical User Interface (GUI) using Angular*. It provides an overview of all collected data as shown in Fig. 5.3. An overview of all houses includes the date of the first and last recorded measurements to provide a quick check for newly added houses or check on upload failures. For each house, a details view of each appliance was added which also listed all manual usage annotations as explained in the following Section 5.2.3. A visualization tool that includes plotting of the recorded data helps to identify false readings.

As server-side hardware we used an Intel Xeon Gold 6140 CPU @ 2.30 GHz with 256 GB RAM. Such powerful hardware is generally not required but increases SQL query speed when a large number of measurements are recorded. It must be noted that in order to utilize the hardware power, the PostgreSQL instance needs to be configured to allocate more memory per query. The server requires an internet link capable of handling the uploaded data of about 140 KB per appliance per hour which is insignificant compared to the required download capabilities for daily backups as well as large queries that require gigabytes to be transferred as fast as possible, and we therefore made use of a 1-Gbit internet link.

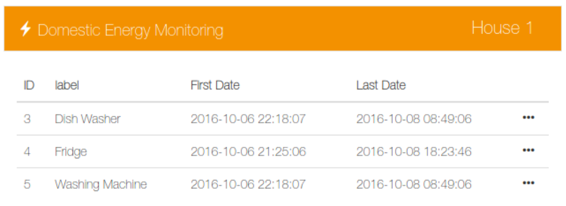
*https://angular.io/



⚡ Domestic Energy Monitoring

ID	Items	First Date	Last Date	
House 0	2	2016-10-06 21:25:06	2020-09-10 15:32:02	▾
House 1	3	2016-10-06 21:25:06	2016-10-08 18:23:46	▾
House 2	3	2016-10-06 21:25:06	2016-10-08 18:23:46	▾

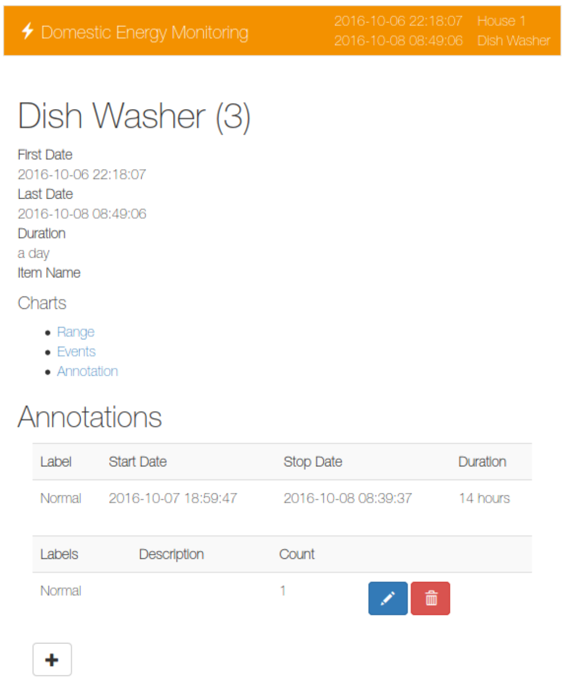
(a) All Houses



⚡ Domestic Energy Monitoring House 1

ID	label	First Date	Last Date	
3	Dish Washer	2016-10-06 22:18:07	2016-10-08 08:49:06	•••
4	Fridge	2016-10-06 21:25:06	2016-10-08 18:23:46	•••
5	Washing Machine	2016-10-06 22:18:07	2016-10-08 08:49:06	•••

(b) Details of House 1



⚡ Domestic Energy Monitoring 2016-10-06 22:18:07 House 1 2016-10-08 08:49:06 Dish Washer

Dish Washer (3)

First Date
2016-10-06 22:18:07
Last Date
2016-10-08 08:49:06
Duration
a day
Item Name

Charts

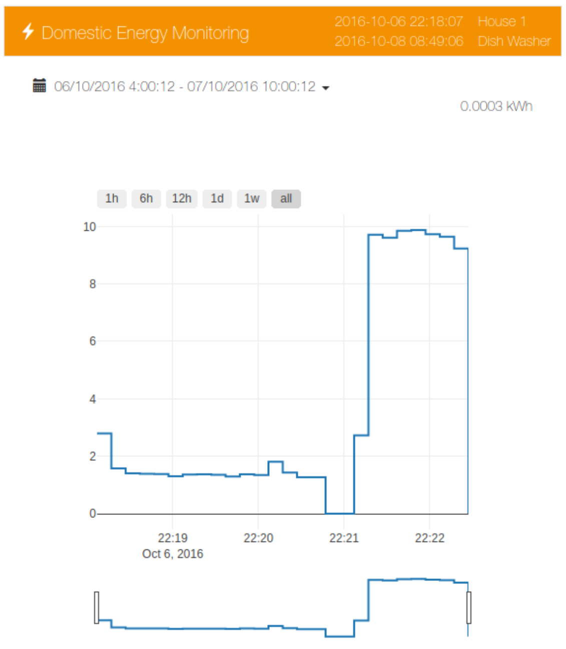
- Range
- Events
- Annotation

Annotations

Label	Start Date	Stop Date	Duration
Normal	2016-10-07 18:59:47	2016-10-08 08:39:37	14 hours

Labels	Description	Count
Normal		1

(c) Details of dishwasher in House 1



⚡ Domestic Energy Monitoring 2016-10-06 22:18:07 House 1 2016-10-08 08:49:06 Dish Washer

06/10/2016 4:00:12 - 07/10/2016 10:00:12 ▾

0.0003 KWh

(d) Visualization of dishwasher in House 1

Figure 5.3: Responsive web GUI implemented as part of the central energy server. The tool gives an overview of all collected data including (a) an overview of all houses, (b) detailed view of all appliances in each house, (c) detailed view of an appliance, and (d) a visualization of the collected data.

Home-/Client-Side

The hardware components used for each home are:

- computer: Raspberry Pi 3 Model B with 32 GB storage,
- WiFi Access-Point: TP-Link TL-WR802N,
- individual appliance meter: TP-Link SmartPlug HS110 (see Fig. 5.4(c)),
- whole-house meter: ABB B23 112-100 + MAX485 module (see Fig. 5.4(d)),
- 5V DC power supply.

For the on-premises computer a Raspberry Pi 3 Model B with 32 GB storage with a standard casing (see Fig. 5.4(b) was used. It offers WiFi, Ethernet, and GPIOs for Modbus communication with a whole-house meter at a very low cost. In cases where a whole-house meter was installed, a DIN rail casing as shown in Fig. 5.4(a) was used to mount the Pi next to the meter since reading data using Modbus requires a cable connection. The full system was pre-configured in our lab, which reduced the steps required for installation by the residents in each home to: Plugging in individual appliance meters and Raspberry Pi, and connecting the Access-Point to a local router. This also allowed us to ship the system as a package without requiring an electrician when only appliance data was collected (i. e., no whole-house meter installed).

The computer runs Hypriot OS (https://blog.hypriot.com), a slim operating system that focuses on Docker container support. We made use of the Eclipse Smarthome (https://github.com/eclipse-archived/smarthome) system for collecting and persisting the data locally into a PostgreSQL database. The appliance meters are read over the wireless network every second, where the timestamp is added at the time the value is received from the meter, thus introducing a reading latency equivalent to the response time of the device plus network latency. The system time is set using the NTP protocol. In order to keep the 1 Hz rate, the next value reading request will take this latency into account. Readings are persisted only on value change to minimize storage space. Additionally, one value is always stored at the full hour in order to detect connectivity problems.

Each hour the new data is exported into a CSV file and uploaded to a central server using SSH with key authentication. In order to identify incomplete uploads on the server side, the SHA checksum was used as a filename and checked before import on the server side.

5.2.3 Event Annotations

The collection of raw data is an important step of all electricity machine learning efforts as it can act as ground truth. For some applications such as appliance usage analysis, the raw electricity measurements are only one part of the required ground truth, as electricity consumption does not necessarily correlate with usage. Appliances with a stand-by mode still have an electric load when not in use. While the BLUED [Ande 12b] dataset does provide ground truth event annotation, the dataset cannot be used for tasks such as user behavior analysis as the recorded period is only about 12 hours.

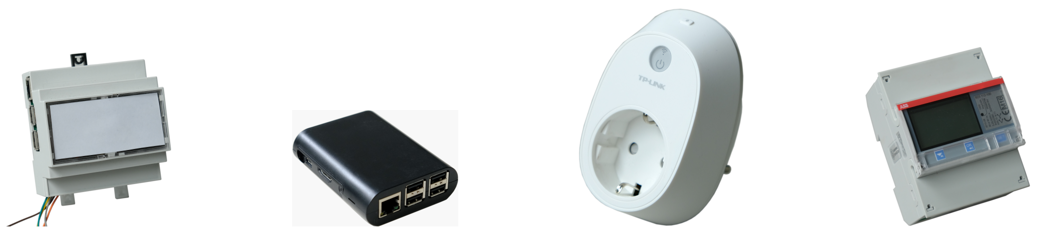

(a) DIN rail (b) Standard (c) Smart Plug (d) Smart Meter

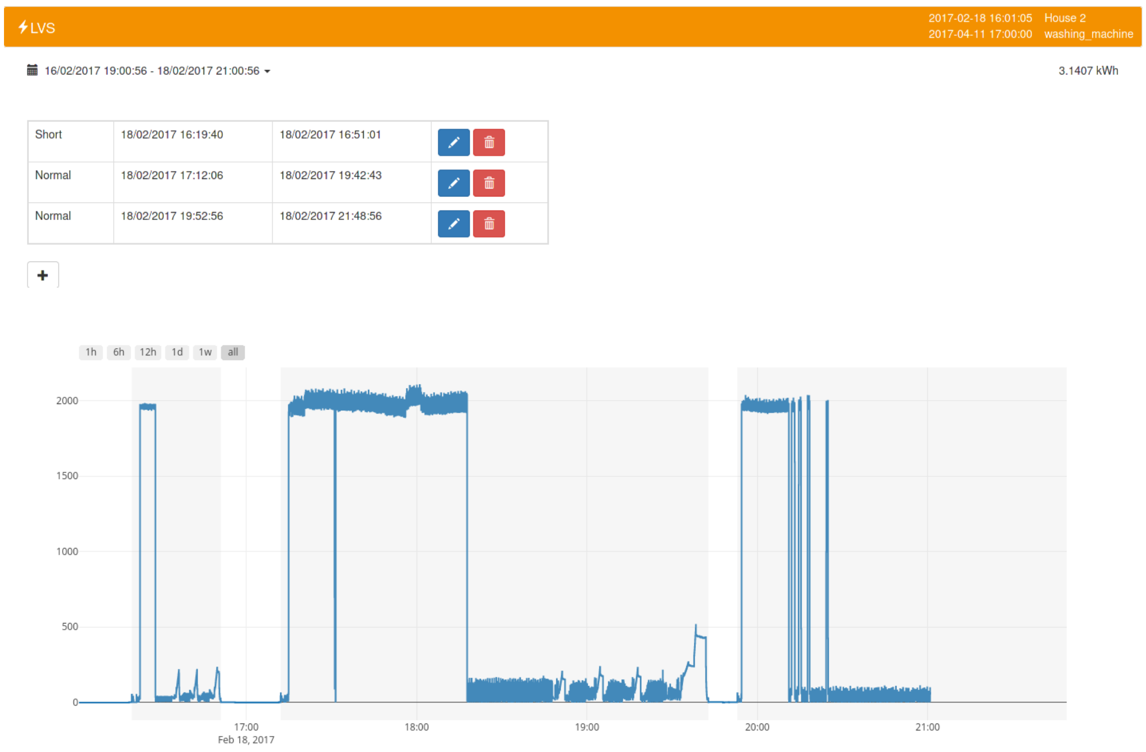
Figure 5.4: Images of the different Raspberry Pi cases, a smart plug, and a smart meter used. (a) was to be used for mounting on a DIN rail next to a whole-house meter, and (b) for all other setups. The smart plug (c) was used to record the appliances and the smart meter(d) was used to record the mains [Wenn 21b], License: CC-BY 4.0

Pereira [Pere 19] takes the annotations from BLUED, adds manual annotations to the UK-DALE [Kell 15] dataset, and combines the results into a new event annotation dataset called NILMPEds [Pere 19]. In the NILM context, events are usually defined as switch-on or -off using a single timestamp. These two events have a logical relationship as a switch-on/start must be followed by a switch-off/stop. We, therefore, publish manual annotations for some of our data where an event is defined by two timestamps: $e_0 = (t_0,\ t_1)$ where $t_0 < t_1$. Manual annotations were created using expert knowledge. For this purpose, an annotation tool has been implemented as part of the central energy server's GUI (see Fig. 5.5) The tool allows multiple people to annotate data at the same time.

Figure 5.6 shows manual annotations of the compressor and light for a refrigerator in house 5. The measurements in the annotation tool have been rounded to full seconds using the provided SQL function `round_timestamp()`, and the annotations are therefore created to full seconds.

5.3 Dataset

The DEDDIAG dataset was mainly recorded to provide the basis for automated domestic load-shift applications where, e. g., a Real-Time-Price is used as an incentive. Hence, the dataset contains appliances that have the potential for automated load-shifting such as fridges, freezers, dishwashers, dryers, and washing machines. According to the German Federal Environment Agency, other applications, outside of space heating and cooling, such as washing machines, dryers, stoves, refrigerators, and similar, account for 52.6% of a household's electricity demand in 2018 [Germ 20]. It is less common to use electricity-based space heating, and therefore space heating accounts only for 5.8%; space cooling systems are generally uncommon and only account for 1.0% [BDEW 19]. In some cases additional appliances that may not contribute to load-shifting were recorded as they give insight into habits. We find it particularly important to provide annotations for appliance usage in the form of Start-End annotations that attribute a certain time span to a label. For instance, on a washing machine, this is the cycle start and end combined with the program used;



LVS

2017-02-18 16:01:05 House 2
2017-04-11 17:00:00 washing_machine

16/02/2017 19:00:56 - 18/02/2017 21:00:56

3.1407 kWh

Short	18/02/2017 16:19:40	18/02/2017 16:51:01
Normal	18/02/2017 17:12:06	18/02/2017 19:42:43
Normal	18/02/2017 19:52:56	18/02/2017 21:48:56

Figure 5.5: Annotation tool that was implemented as part of the central energy server where data was collected from different households.

for a fridge, this highlights the compressor cycles as well as the light that indicates an open fridge door. The latter is especially interesting as it gives a clear indication of an occupant being present.

This is why, in the DEDDIAG dataset, not only the raw data was published, but also manual annotations, demographic information, and model names for some appliances to provide a basis for research of device identification, behavior analysis, and load-shift recommendation systems.

Data Records

The DEDDIAG dataset provides raw 1 Hz power readings of load-shifting relevant appliances over periods of 21 to 1351 days. For one home (house 8) we also provide whole-house mains readings. In total, the dataset contains 15 homes with a total of 50 appliances. A detailed listing of all homes and appliances, duration, and missing data can be found in Table A.1.

Figure A.1-A.4 show stacked daily average consumption for all houses. The aligned charts show the periods the data was recorded and can be used for visual examination of missing data or analyzing seasonal changes where, e. g., refrigerators require more energy during summer periods.

For 14 cycle-based appliances DEDDIAG provides manual annotations. These annotations provide start and stop timestamps of a cycle; there are multiple labels per appliance where possible in order to annotate different modes. For example, for the refrigerator with item ID 10 in house 0, we provide separate annotations for light

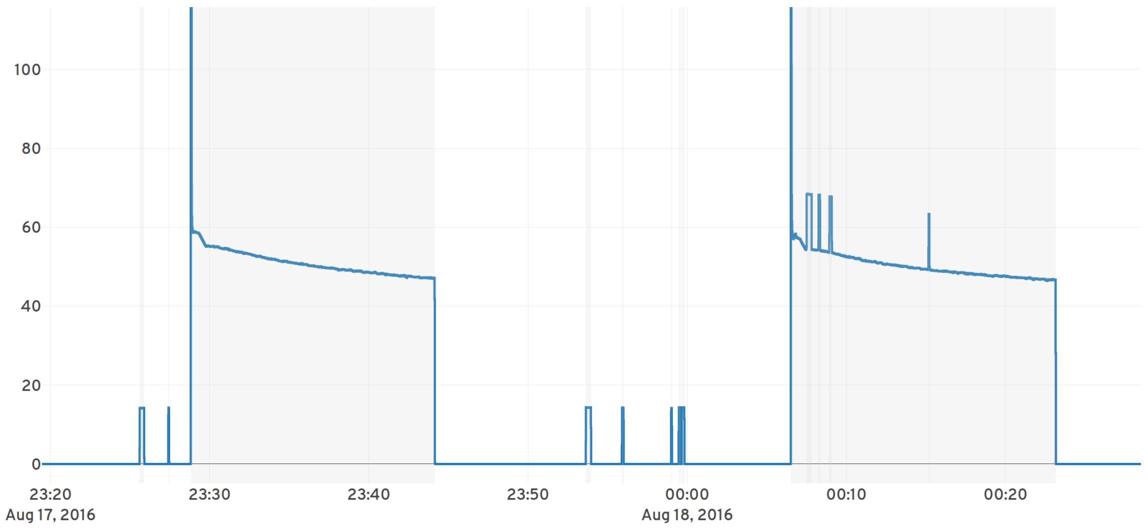


Figure 5.6: Measurements (blue curve) and annotations (light grey) of a refrigerator (House 5/Appliance 09) showing annotations for compressor and light cycles over a period of 1.5 hours on the 2016-08-17. Compressor and light are annotated as separate events in order to classify user interaction (door opening) by using the light annotations.

and compressors. Thus, the annotations can be used as ground truth for appliance usage predictions which, to our best knowledge, does not exist for any long-term electricity dataset available. This will allow training and evaluation of classifiers that can detect "usage" of the refrigerator based on the light being switched on when the door is opened. Additionally, we provide demographic data for each household's residents such as age, absence duration, and regularity of absence. The DEDDIAG dataset is open-access and is hosted on Figshare [DEDD 21].

Structure

All data are provided as plain text, tab-separated value (TSV) files. Figure 5.7 shows the dataset file structure. There is one directory per house (house_00/, house_01/, ...) containing a `house.tsv` file with the house description. The description provides demographic data such as the number of residents, their age, regularity of absence, and normal absence duration. Appliances metadata is provided in `items.tsv`, containing a unique ID, name, device category, and house ID. Appliance categories provide a grouping label. Available categories are Refrigerator, Freezer, Washing Machine, Dryer, Dish Washer, Coffee Machine, TV, Office Desk, Smart Meter Phase, Smart Meter Total, and Other.

The measurements, annotation, and annotation labels of each appliance are provided in separate files: `item_XXXX_data.tsv.gz`, `item_XXXX_annotations.tsv`, `item_XXXX_annotation_labels.tsv`. The measurement files are each compressed using gzip to reduce size. The data is split per house and appliance to make it possible to work on a single home or appliance. While the tsv files can easily be used directly, an import bash script called `import.sh` is provided to automatically import the data into a Docker-based PostgreSQL database. Detailed instructions are

```
/
├── house_00
│   ├── house.tsv ........................................ House description
│   ├── items.tsv ...................................... Appliance descriptions
│   ├── item_0001_data.tsv.gz ......................... Power measurements
│   ├── item_0001_annotations.tsv ............................. Annotations
│   ├── item_0001_annotation_labels.tsv ............... Annotation labels
│   ├── item_XXXX_data.tsv.gz
│   ├── item_XXXX_annotations.tsv
│   └── item_XXXX_annotation_labels.tsv
├── house_XX
│   └── ...
├── import.sh ............................................. Import script
├── create_tables.sql .................... Database tables and functions
└── README.md ......................................... Dataset description
```

Figure 5.7: Dataset directory structure. Each appliance is provided in a separate file in order to simplify only using a subset of appliances.

provided as part of the dataset archive in `README.md`. The compressed dataset has a size of 14 GB, and the uncompressed, imported data requires about 140 GB. All measurements are stored in a sparse manner where only value changes are recorded. In order to be able to detect connectivity problems of the metering device, there is at least one reading per hour. Therefore, if the time span between two measurements is greater than 1h 5sec, it should be assumed that there are missing values. A measurement record is stored as `<item_id> <time> <value>` where `<item_id>` corresponds to an ID in `items.tsv`. `<time>` is a UTC datetime string using the `YYYY-MM-DD HH:MI:SS.US` format, measurements are stored as float values.

Annotations are provided as `<id> <item_id> <label_id> <start_date> <stop_date>` where `start`, `stop` are UTC datetime strings using the `YYYY-MM-DD HH:MI:SS` format. Annotations are only provided to full seconds, thus the corresponding measurement can be found using the provided custom SQL function `round_timestamp()`.

5.4 Validation

All parts of the data collection system have been tested in a lab environment. The collected measurements are captured as provided by the metering device. The mains meter is certified to be MID (Measuring Instruments Directive) calibrated, the appliance-level meters are not. A lab test on the wireless appliance meter devices has been performed to assure constant readings over different metering devices. Another lab test has been performed to compare readings of the appliance and mains meters, showing that the appliance meters systematically provide 2% to 4% lower readings compared to the mains meters.

Table 5.1: Comparison of available data of item 69 and 4 as an example for missing data caused by interruption of the monitoring. The table shows how much of the missing data is of a certain length, illustrating the differences in missing data. 89.3% of the missing data of item 69 come from interruptions > 5 days, meaning there are mostly long-term interruptions. 50% of missing data on item 4 come from interruptions > 2 days, meaning there are mostly short-term interruptions.

Gap Length	**Item 69**	**Item 4**
> 1 hours	100.0%	100.0%
> 2 hours	100.0%	97.5%
> 3 hours	100.0%	94.5%
> 4 hours	100.0%	92.7%
> 5 hours	100.0%	92.4%
> 6 hours	100.0%	92.2%
> 1 days	97.4%	82.9%
> 1 days 12 hours	97.4%	64.4%
> 2 days	97.4%	50.0%
> 3 days	97.4%	12.6%
> 4 days	94.0%	3.9%
> 5 days	89.3%	0.0%
> 33 days	34.0%	0.0%

Sample Rate Precision

Although there is one reading per second, the exact rate varies due to the technical limitations of the metering device. For single appliance measurements, the rate will usually fluctuate between 0.9 Hz and 1.1 Hz. The timestamp precision is sub-second, rounding to full seconds may result in having two measurements for the same second. The recommended handling for such cases is to only keep the later value to avoid changing the time order of the two values, since this may introduce a significant error considering only value changes are recorded.

Missing Values

There are missing values within the dataset. In order to be able to detect failures, the system recorded at least one value per hour per appliance, and therefore time gaps that are greater than 1h 5sec must be treated as missing values. The additional 5 seconds are added as a fair dealing gap because small deviations from the 1h are of no significance as these gaps only represent periods where the electricity demand has not changed, which in reality only applies to 0 Watt cases. An overview of monitoring gaps is given in Table A.1 as the sum of missing periods relative to the total duration with a gap size of >1h 5sec and >1day, where the latter must always be smaller. The >1h 5sec missing data ranges from 0.28% on item 38 to 59.40% on item 4, and the >1 day missing data ranges from 0.00% to 49.26% for the same appliances.

As an example Table 5.1 provides a more detailed insight into monitoring gaps of item 4 and item 69 that have an overall $>$ 1h 5sec missing data of 59.40% and

Table 5.2: Listing of published data and source code. All source code is published under MIT license, the data and publication under CC BY 4.0.

Dataset on Figshare	CC BY 4.0	http://dx.doi.org/10.6084/m9.figshare.13615073 [DEDD 21]
Scientific publication	CC BY 4.0	https://rdcu.be/coGqL [Wenn 21b]
Python loader	MIT	https://pypi.org/project/deddiag-loader/
Collection system	MIT	https://DEDDIAG.github.io
NILMTK converter	MIT	https://github.com/nilmtk/nilmtk/tree/master/nilmtk/dataset_converters/deddiag

35.06%. The items were chosen as an example since they have very different missing data patterns. The table shows how much of the missing data falls within a certain gap length, clearly indicating that item 69 suffers from long-term interruptions where 89% of the missing data are from gaps $>$ 5 days where there is no gap of that length for item 4. Item 69 has a single monitoring gap of about 33 days, accounting for 34% of its missing data, while the largest gap for item 4 is 4 days.

As the analysis as well as the impacts of the missing data depend on the application of the dataset, detailed inspections of missing data must be performed when using the dataset. The DEDDIAG-loader software library described in Section 5.5 provides a method to find missing periods to assist analysis. Missing data can also be seen visually in the daily average power demand, Figs. A.1–A.4.

5.5 Availability

The dataset as well as all described software components have been published online. An overview is given in Table 5.2. The full data collection system is published under MIT license and is available under https://DEDDIAG.github.io. The dataset itself is published as tab-separated text files together with code to import all data into a PostgreSQL instance. An SQL function called `get_measurements()` is provided to get seconds-based measurements, where readings are converted from value-changes to seconds-based readings using interpolation; timestamps are rounded to nearest seconds. There is also a python package available https://github.com/DEDDIAG/DEDDIAG-loader.git that assists in retrieving data into a pandas-DataFrame/numpy-array. The package[*] has also been uploaded to https://pypi.org and it can be installed using the package-management system pip: `pip install deddiag-loader`

Additionally, the dataset has been added to the largest available NILM toolkit, known as Non-Intrusive Load Monitoring Toolkit (NILMTK)[†]. The parser is mainly created for house 8, where aggregated main readings are available. NILMTK is a python toolkit containing parsers for multiple electricity datasets and implementations of common energy disaggregation algorithms [Batr 14a, Kell 14, Batr 19].

[*] https://pypi.org/project/deddiag-loader/
[†] https://github.com/nilmtk/nilmtk

5.6 Summary

This chapter introduced a new data collection system and the resulting new dataset with ILM research in focus. No open-source collection system has been previously available that fulfills all requirements for such a task. Since the system is published as open source and only uses open-source components, it can potentially be extended for further research. Possible extensions are the implementation of Machine Learning Demand Response Model (MLDR) components in order to provide actions that can either be operated automatically or by eliciting customer feedback.

The obtained and published dataset called DEDDIAG has some unseen properties. The collected dataset gives insights into German households and covers a long time period, ranging from 1–3.5 years. Furthermore, it includes manual annotations of appliance switching-events, providing ground-truth for appliance segmentation and usage predictions. A list of public datasets, including this one can be found in Section 4.1.3.

CHAPTER 6

Appliance Identification

Substantial parts of this chapter's Section 6.3 have been published in the following manuscript: [Wenn 21b]. Substantial parts of the Section 6.4 have been published in the following manuscript: [Wenn 21a]. The majority of the manuscript has been authored by the author of this thesis. The main scientific contributions are based on his own work and thoughts.

6.1 Introduction

Appliance identification comes in various shapes. It is referred to as appliance identification, appliance recognition, appliance classification or appliance event detection. The identification is done in two steps, a pre-processing step, where features are extracted, followed by the actual identification task.

As mentioned in Chapter 4, the difficulty of these tasks depends on the input data. For Intrusive Load Monitoring (ILM) 3 data, where the data points are not an aggregate of multiple appliances, each data point belongs to a distinct category, whereas on the other hand in Non-Intrusive Load Monitoring (NILM) and ILM 1/2, a data point belongs to a set of appliances, thus to a set of categories.

In the NILM context, the identification and segmentation is sometimes seen as a single step, as the device identification is done based on the characteristics of the appliance's switching events [Froe 11]. Following the Machine Learning Demand Response Model (MLDR) described in Section 4.4, here the identification and segmentation are treated as a two-step process. The segmentation step is further discussed in Chapter 7.

In machine learning terms, the determination of the appliances m is a classification problem. Since the data points are recorded chronologically, appliance identification is a time-series classification problem. Appliance identification is the task of associating the temporal data points with an appliance category (Dishwasher, Refrigerator, Office Desk, ...), a manufacturer, or the appliance's model. Further, the data points belong to a specific mode or state the appliance is currently in. The result is an association of temporal data points with a specific class m, where $m \in \{1, \ldots, M\}$. This association is performed using a classifier.

For simplicity, the term appliance identification is defined as the task of identifying the appliance category.

The available appliance identification techniques can be divided into two main categories: low and high sample rate approaches. In the following low sample rates are defined as $<$100 Hz and high sample rates as $>$100 Hz. Although the 100 Hz is equivalent to double the common utility frequency in Europe 2×50 Hz, which would, in theory, satisfy the Nyquist–Shannon sampling theorem of $f_{\text{sample}} > 2f_{\text{max}}$, the boundary is set based on typically used sampling rates. Taking harmonic distortion into account (see Section 4.2.2), f_{max} is far above the utility frequency of 50 Hz, thus high sample rate approaches require sample rates in orders of magnitude higher (kHz).

6.2 Related Work

6.2.1 Low Sample Rate

Zaidi et al. investigated the use of templates of power consumption profiles which are used to match new readings using Dynamic Time Warping (DTW) [Zaid 10]. Additionally, they investigate building a Hidden Markov Model (HMM) to identify loads based the consumption profiles. The profile is made up of features such as average energy consumption, edge counts, and percentage energy consumption. The use of Discrete Fourier transformation (DFT) coefficients was also tested, but not deemed a relevant feature due to "the mostly permanent character of loads with no or very less transitions" [Zaid 10]. Their work was discussed on a nonpublic dataset recorded at a sample rate of 1/10 Hz and includes appliances such as Fridges, Microwaves, Dishwashers, Coffee Machines, Computers, and Printers. Unfortunately, the publication does not present an evaluation based on performance metrics.

Weiss et al. present a NILM appliance identification based on edge detection and load profile matching [Weis 12]. They propose a multi-step process: normalization, edge detection, power and delta level computation, load profile matching. The selected input feature is apparent power S. The edge detection is based on the absolute value change in the apparent power signal. Potential edges are filtered using a single, lower-bound threshold. The authors tested smoothing the signal to remove unwanted edges using different filters such as a median filter, a mean filter, and a kernel-weighted average filter (Nadaraya-Watson filter with Gaussian kernel). They conclude that using a large Gaussian kernel gives the biggest reduction in potential edges, while not missing the wanted edges. The described edge detection falls in the category of switching events, and the authors pair the ON/OFF events. Based on the found switching-event pairs a feature matrix is created containing the step-up and step-down signature and the start and end. The authors search for matching signatures in the database using a nearest-neighbor algorithm. The described approach inverts the classification and segmentation steps in the MLDR by first finding the load profile γ, before identifying the appliance type. As discussed in Section 4.2.3, and illustrated in Fig. 4.9, such edge-detection algorithms can be insufficient for many appliance types due to the nature of their load profile.

Reinhardt et al. present an ILM appliance identification approach based on multiple power features [Rein 12]. The authors use an empirically determined single lower bound threshold approach to detect switching events. Based on the segmented appliance cycles, time-related and power consumption-related features are extracted. The time-related features, usage time, usage frequency, and total usages, are extracted as well as day- and week-based features. A feature representing the common usage time of the day is created by dividing each day into 144 bins, forming a vector where each scalar presents whether the appliance has been active or not. Additionally, a similar vector indicates the usage during weekdays or weekends. Next to the time-related features, power-related features such as maximum power consumption, average power, average variance, and power levels (5, 10, 50, 200, 500, and 2,000 watts) are used. Furthermore, noise level and statistical features are also considered. The noise or high fluctuations are described by comparing a low-pass filtered curve with the original. The extracted features are the number of points of inflection and the value of the first minimum. In total, 517 different features are used to describe each power consumption trace in the dataset. Using 25-fold cross-validation, nine different classification algorithms have been evaluated: Bagging, Bayesian Network, J48, JRip, LogitBoost, Naive Bayes, Random Committee, Random Forest, and Random Tree. The resulting accuracies range from 84.21% to 95.5%, where JRip gives the worst and Random Committee the best result.

Single appliance recordings were used to evaluate the approach. These recordings have been published under the name *tracebase*. The data were recorded at a sample rate of 1/3 Hz over a period of 24 hours.

Ridi et al. describe an ILM approach based on power features and the derived delta and delta-delta coefficients [Ridi 13]. The features used are the measured real power P, the reactive power Q, the root-mean-square (RMS) current I, the RMS voltage V, the electric frequency f, and the phase ϕ. Additionally, the delta and delta-delta coefficients of all observations $O = (o_1, o_2, \ldots, o_N)$ are calculated. The delta features are calculated as:

$$\Delta o_n = \sum_{k=-K}^{K} k o_{n-k} \quad , \tag{6.1}$$

where the window length equals $(2K + 1)$ [Henn 98]. The delta-delta coefficient is defined as:

$$\Delta\Delta o_n = \Delta o_{n+1} - \Delta o_{n-1} \quad . \tag{6.2}$$

The final feature vector is composed of the z-normalized original measurements O and the computed coefficients delta and delta-delta. The authors evaluate the performance of two different classification algorithms, Gaussian Mixture Model (GMM) and k-Nearest-Neighbor (kNN), on the Appliance Consumption Signature-Fribourg 1 (ACS-F1) dataset [Gisl 13]. This dataset is sampled at 1/10 Hz and contains one-hour readings from 10 different appliance categories. In total, two sessions of 100 appliances have been recorded. The authors present their performance evaluation using a confusion matrix as well as accuracy. The best overall performance, with an accuracy of 93.6% was obtained using the GMM, while the kNN achieved an accuracy of 90%. They conclude that the delta and delta-delta features improved the accuracy, although on the kNN classifier, the accuracy only improved by 2 percentage points.

6.2.2 High Sample rate

Saitoh et al. proposed an identification approach evaluated using 4.44 kHz data [Sait 10]. The proposed method contains three pre-processing steps: phase shift alignment, feature extraction, and normalization. The phase alignment is done by finding the first peak value in the voltage signal and shifting the signals to the left, so the measurement starts at the maximum. From the current signal, the authors extract three features from a single cycle: the maximum (I_{peak}), average (I_{avg}), and RMS current (I_{rms}). Based on these features, three derived features are calculated: $\text{CF} = \frac{I_{\text{peak}}}{I_{\text{rms}}}$, $\text{FF} = \frac{I_{\text{rms}}}{I_{\text{avg}}}$, and $F_{\text{pta}} = \text{CF} \cdot \text{FF} = \frac{I_{\text{peak}}}{I_{\text{avg}}}$. Additionally, the rising and falling edge angle of the voltage signal as well as two ratios that describe the proportion of the values in one cycle being above $0.1 \cdot I_{\text{peak}}$ and $0.8 \cdot I_{\text{peak}}$ are used. These 10 features form the final feature vector that is classified using two different classifiers: Support Vector Machine (SVM), and kNN. The kNN has been tested using $k = 1$ and $k = 3$. The authors found no significant difference between the classifiers in their evaluation, and that I_{rms}, I_{avg}, and I_{peak} are the most significant features.

De Baets et al. propose a method based on voltage-current (V-I) trajectory features classified using a Convolutional Neural Network (CNN) [De B 18b]. Their method used the voltage and current signature when the appliance is in steady-state and create the normalized V-I trajectory as a 2-dimensional, binary-image-like, matrix. This matrix is fed into a CNN which outputs the appliance category. Their approach has been evaluated on two public datasets: COOLL and WHITED [Pico 16, Kahl 16]

Faustine et al. proposed a method similar to De Beats et al., which is based on classifying the Recurrence Plot (RP) of the V-I trajectory using a CNN [Faus 20, Faus 21]. This very recent approach is the basis for the algorithm presented in Section 6.3, where a more detailed description is provided.

6.3 A Low Sample Rate k-Nearest Neighbor Method

An appliance identification algorithm must be capable of identifying an appliance category (Fridge, Dishwasher, ...) based on the available data. In low sampling rate scenarios (<100 Hz), only power-based approaches have been developed, as all other features such as harmonics are only observable at high resolution. At low sample rates, the measurements are averaged, commonly using the RMS value (cf. Section 4.2.2). A few publications have been made using different classification algorithms, all based on power features (cf. Section 6.2.1).

In the following, a power feature-based appliance identification is presented. The method was developed for the Domestic Energy Demand Dataset of Individual Appliances in Germany (DEDDIAG) dataset, working on measurements recorded at 1 Hz sample rate [Wenn 21b].

The appliance identification task requires boundary definition by formalizing the goal of the task. In the DEDDIAG dataset, some appliances have been recorded for up to 3.5 years. With real-world scenarios in mind, the goal for an appliance identification task cannot be to identify an appliance based on all available data, but rather on a window that is as small as possible. It would be preferable to reduce the required data points to a few minutes. Further, the identification is based on ILM 3 data (single appliance recordings).

6.3.1 Method

The classification process contains two steps: data pre-processing/feature extraction and classification. Since the task is performed on power measurements, the input data will be the raw measurements as available in the DEDDIAG dataset. These raw measurements are the RMS power for each appliance. The pre-processing step is composed of a feature extraction based on a sliding window. The appliance identification task is performed on all available appliances using the non-aligned recordings, meaning that the classifier has to be able to perform on a randomly chosen segment. The sparse recorded data is taken as is, which prevents trying to classify high-duration switched-off states where the power demand is 0 watts. Each window is classified independently, testing the influence of input data length on the classification result. Multiple experiments are performed to evaluate different feature extraction and classification methods.

Feature Extraction

A single long time-series needs to be split from the other series for classification. A sliding window approach as shown in Fig. 6.1 is used to cut the time series into smaller parts. Each window is then used as the input for the classification task, thus the window represents the portion of the signal that is required to identify the appliance. The window sizes ws used for the evaluation are defined as:

$$\mathrm{ws} = 2^x, \quad x \in [2, 11] \quad . \tag{6.3}$$

Based on the sliding windows, a feature vector is formed using the windows mean value and Discrete Wavelet Transformation (DWT) coefficients. The DWT is performed to the maximum level, resulting in four coefficients per window. Combined, this results in a five-dimensional feature vector, which is normalized and standardized independently based on the training data.

Mean Value is simply defined as the mean of each window:

$$\frac{1}{\mathrm{ws}} \sum_{n=1}^{\mathrm{ws}} \mathrm{windows}_n \quad . \tag{6.4}$$

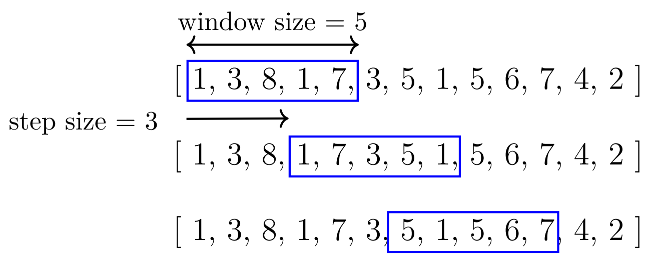


Figure 6.1: Sliding window over a sample time-series sequence. The blue box defines a window. The window size defines the number of values per window taken from the sequence. The step size defines the number of values jumped between each window. When step-size < window-size, the windows are overlapping.

Discrete Wavelet Transformation is a mathematical decomposition of a discrete signal into a set of coefficients describing the signal. The decomposition is done using a wavelet, a wave-like oscillation that is localized in time. The advantage of DWT over the Fourier transformation (FFT) is the temporal resolution: the DWT reflects the time evolution of the frequency. The wavelet transformation was first introduced by Grossman et al. [Gros 84].

A wavelet theory is commonly described as the representation of a square-integrable function on $\mathbb{R}$ by small, square-shaped series that must decay to zero [Flee 08]. A discrete wavelet transform of a signal f is calculated by convoluting the wavelet Ψ over f:

$$y_t = \sum_{k=0}^{N-1} f_{t-k}\Psi_k \quad , \tag{6.5}$$

where Ψ is zero, expect a finite integer range. The most common wavelets are known as *Daubechies wavelets*, named after the mathematician Ingrid Daubechies [Daub 88]. In this thesis, the Daubechies wavelet 1 (db1) is used, which corresponds to the Haar wavelet, defined by the wavelet function Ψ as:

$$\Psi(t) = \begin{cases} 1 & 0 \leq t < \frac{1}{2}, \\ -1 & \frac{1}{2} \leq t < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{6.6}$$

In order to obtain the so-called detail coefficients and approximation coefficients, the resulting vector $\boldsymbol{y}$ is processed using a high-pass filter $\boldsymbol{h}$ and low-pass filter $\boldsymbol{g}$. This combination of a high- and low-pass filter is called *filter bank* [Flee 08]. Since each filter outputs half the frequency band of the input $\boldsymbol{y}$, the resulting vector can be sub-sampled by 2, resulting each in half the vector length of $\boldsymbol{y}$. Repeating this decomposition on the low-pass filtered results, multiple decomposition levels are obtained.

The multi-level decomposition has a signal compaction and noise reduction property while being computationally efficient. These properties are desired from a machine learning point of view, as any dimension reduction will decrease the complexity of the classification task. The decomposition iterations can be performed until the signal becomes shorter than the length of the interval the filter is defined. The maximum level of decomposition is therefore defined given by:

$$l_{max} = \left\lfloor \log_2 \frac{|f|}{|\Psi| - 1} \right\rfloor \quad , \tag{6.7}$$

where $|f|$ is the length of the input vector f, and $|\Psi|$ the length of the filter. For the here used wavelet db1, $|\Psi| = 2$. In the presented evaluation, the decomposition is always performed to $l_{max} - 1$. On window sizes defined as in (6.3), the decomposition to the maximum level results in one detail coefficient and approximation coefficient each, thus $l_{max} - 1$ will result in two coefficients each. Thus, the number of features is independent of the used window size.

Classification

The DEDDIAG dataset has appliances from nine different categories: Coffee Machine, Dishwasher, Dryer, Freezer, Office Desk, Other, Refrigerator, TV, and Washing Machine. The classification is performed using the kNN algorithm, a very robust and commonly used classifier with the capability of separating non-linear class boundaries based on a distance metric. In this evaluation, the separation is performed based on 5 neighbors using the Euclidean distance.

6.3.2 Evaluation

The evaluation of the classification approach requires to splitting the recorded data into a training and test set. The available data for each category are a set of different appliance recordings. The main challenges when using the data are:

- Different number of data points available per category,
- Different number of data points available per appliance,
- Different number of appliances per category.

These challenges are overcome by using the following evaluation principles:

- Limit the number of used data points. For the DEDDIAG dataset 5.000.000 points per appliance are used. This corresponds to a time range of about 58 days per appliance.
- Use an equal amount of data per appliance in each category by dividing the per-category data point limit by the appliances in each category.

The data extracted in this manner is then evaluated using k-fold cross-validation.

k-fold cross-validation

k-fold cross-validation is a popular data split procedure for the performance evaluation of different classification algorithms. The procedure divides a dataset into k disjoint folds of equal size, where $k > 1$. $k - 1$ folds are used as the training set, and the remaining fold is used as the testing set. Each fold will act once as the testing set, resulting in k different evaluations. The result of each fold is then commonly averaged over all folds.

Cross-validation on long-time series data is not trivial. A scenario, where the data of each category originate from different appliances, requires a sophisticated splitting approach, especially in combination with sliding windows. In an example with two categories $C = \{\boldsymbol{\mathcal{C}}_0, \boldsymbol{\mathcal{C}}_1\}$, the data vectors are of the following format:

$$\boldsymbol{\mathcal{C}}_0 = [(a_1, a_2, a_3, \ldots, a_N), (b_1, b_2, b_3, \ldots b_N)] \quad , \tag{6.8}$$

$$\boldsymbol{\mathcal{C}}_1 = [(c_1, c_2, c_3, \ldots, c_N), (d_1, d_2, d_3, \ldots, d_N)] \quad , \tag{6.9}$$

where $\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}\}$ are the data points of different appliances and N the number of recordings for a given appliance. Here, for simplification N is assumed to be the same for each appliance. In reality, this may not be the case and each appliance or category has a different number of recordings per appliance.

The data is split into k parts so that each fold contains data from every appliance in each category as well as every category. This is done by splitting each appliance in each category separately into k folds:

$$\boldsymbol{\mathcal{C}}_0 = [((a_1, a_2, a_3), (a_4, a_5, a_6)), ((b_1, b_2, b_3), (b_4, b_5, b_6))] \quad , \tag{6.10}$$

$$\boldsymbol{\mathcal{C}}_1 = [((c_1, c_2, c_3), (c_4, c_5, c_6)), ((d_1, d_2, d_3), (d_4, d_5, d_6))] \quad , \tag{6.11}$$

where $k = 2$, and $N = 6$ thus resulting in two splits per appliance with 3 values each. Based on this split, each fold $\boldsymbol{F}$ will therefore result in a feature vector as follows:

$$\boldsymbol{F}_0 = [(a_1, a_2, a_3), (b_1, b_2, b_3), (c_1, c_2, c_3), (d_1, d_2, d_3)] \quad , \tag{6.12}$$

$$\boldsymbol{F}_1 = [(a_4, a_5, a_6), (b_4, b_5, b_6), (c_4, c_5, c_6), (d_4, d_5, d_6)] \quad . \tag{6.13}$$

In the first iteration, $\boldsymbol{F}_0$ is the training set $\boldsymbol{F}_1$ as the testing set and vice versa in the second iteration.

In a sliding window-based approach, the windows have to be created independently for each element (appliance) data in $\boldsymbol{F}$. Otherwise, artifacts would be created on the edges between appliances, e. g., between $\boldsymbol{a}$ and $\boldsymbol{b}$, resulting in windows containing values of two appliances: (a_3, b_1). Thus, for windows size $\mathrm{ws} = 2$ and a step size of 1, the final data vectors in $\boldsymbol{F}_0$ and its corresponding category vector (labels) $\boldsymbol{\mathcal{L}}$ are:

$$\boldsymbol{F}_0 = [(a_1, a_2)(a_2, a_3), (b_1, b_2), (b_2, b_3), (c_1, c_2), (c_2, c_3), (d_1, d_2), (d_2, d_3)] \quad , \tag{6.14}$$

$$\boldsymbol{\mathcal{L}}_0 = [0, 0, 0, 0, 1, 1, 1, 1] \quad . \tag{6.15}$$

In the following experiments, the evaluation is performed using $k = 5$, window sizes $\mathrm{ws} = 2^n$ with $n \in [2, 11]$, and step size 3.

Results

Evaluation results are presented in three different ways:

- boxplot in Fig. 6.2 combining results of all folds at all window sizes,
- confusion matrix in Fig. 6.3 showing the changes in misclassification at different window sizes,
- detailed results in Table 6.1 show the F_1-score per class and the average over all classes for tested window sizes.

In general, the results indicate that the F_1-score increases with increasing window size, which means that although the number of features presented to the kNN algorithm are independent of the window size, the extracted window features contain more distinct class boundaries when using a larger window size. An analysis using the boxplot indicates three main differences in the classification performance. An explanation of the boxplot and interquartile range (IQR) can be found in Section 3.5.7. The Coffee Machine category has the smallest IQR and the Dishwasher the largest. All others have similar IQR, although TV and Washing Machine have a few poorly performing outliers, indicated by the rhombus-shaped dots. The best-performing category at large window sizes is the Freezer, with a maximum score of 0.9551. The Coffee Machine category performs very well at all window sizes. This is indicated by an overall narrow distribution of the results, indicated by a small IQR and short whiskers. Detailed results table shows that even at $\mathrm{ws} = 4$ an F_1-score of 0.9204 is achieved, peaking at $\mathrm{ws} = 128$ with an F_1-score of 0.9487. The Dishwasher classification performance is by far the most dependent on large window sizes, with an increasing score and increasing window size ranging from 0.5145 to 0.8886, also indicated by the large IQR in the boxplot. Here the conclusion in the tested window size range is: the larger the better. The confusion matrix shows that on low window sizes, the Dishwasher is most commonly misclassified as a TV, and vice versa. Dryer, Freezer, and Others perform similarly with an average score just below 0.9. The TV performance improves significantly with increasing window size, with scores ranging from 0.7571 to 0.9408. The washing Machine and Office Desk have a similar result, both only having a score above 0.80 at large window sizes. While at window size 2048 nearly all categories perform well with scores > 0.88, the Washing Machine performs considerably less reliably with a score of 0.87092. In general, the misclassification patterns change with window size.

The original results published in the DEDDIAG manuscript has been evaluated using a different cross-validation approach. The published results can be found in Appendix B.1.

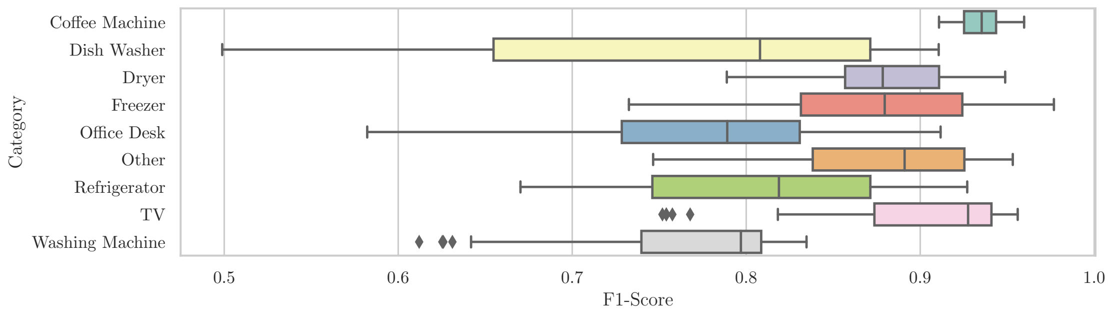


Figure 6.2: Classification results for the kNN classifier. The results of each fold and different window sizes are combined. Boxes indicate the quartiles of results, the vertical bar is the median, and the whiskers indicate the range of distribution within $1.5 \times$ IQR. The rhombus-shaped dots indicate outliers.

Table 6.1: Baseline result of appliance category identification using kNN classification. Results show F_1-score for each category at tested window sizes as well as the weighted average over all classes based on a 5-fold cross-validation. Results strongly indicate that F_1-score increases with increasing window size.

Window Size	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**	**2048**
Coffee Machine	0.9204	0.9269	0.9291	0.9353	0.9479	0.9487	0.9357	0.9319	0.9346	0.9335
Dishwasher	0.5145	0.5764	0.6497	0.7229	0.7893	0.8410	0.8633	0.8717	0.8745	0.8886
Dryer	0.8384	0.8452	0.8509	0.8705	0.8841	0.9013	0.9071	0.8975	0.8956	0.8901
Freezer	0.7964	0.8177	0.8368	0.8553	0.8757	0.8976	0.8967	0.9014	0.9253	0.9551
Office Desk	0.6810	0.7102	0.7351	0.7555	0.7719	0.7967	0.7969	0.8180	0.8517	0.8724
Other	0.8359	0.8442	0.8502	0.8565	0.8652	0.8756	0.8885	0.9010	0.9142	0.9333
Refrigerator	0.6916	0.7167	0.7436	0.7757	0.8041	0.8278	0.8459	0.8741	0.9040	0.9158
TV	0.7571	0.8246	0.8755	0.9034	0.9206	0.9350	0.9418	0.9443	0.9432	0.9408
Washing Machine	0.6273	0.6795	0.7383	0.7892	0.7966	0.7982	0.8032	0.8087	0.8082	0.8092
Average	0.7358	0.7679	0.7987	0.8275	0.8491	0.8677	0.8741	0.8826	0.8947	0.9053

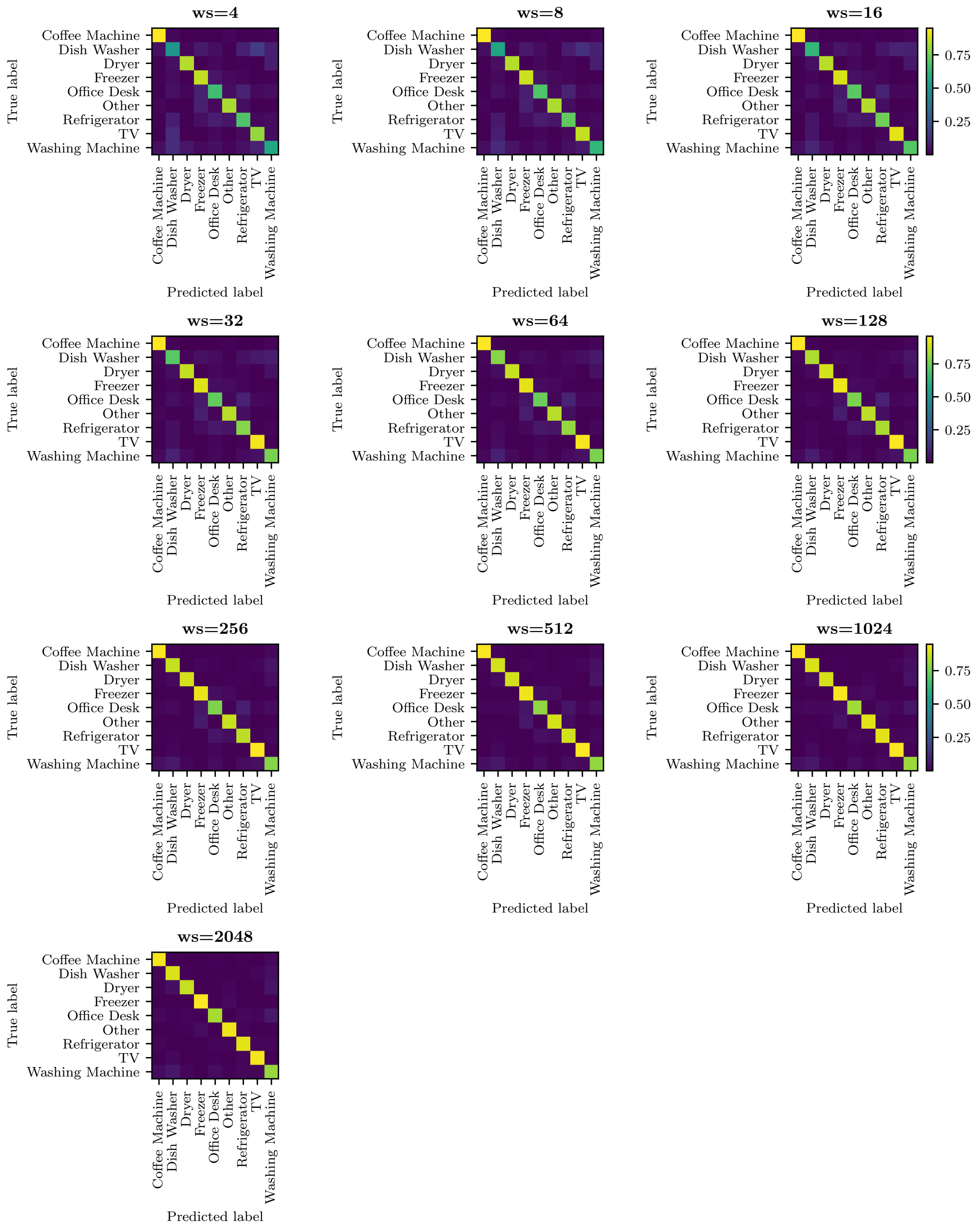


Figure 6.3: Combined and normalized confusion matrix of test split at all evaluated window sizes ws $= 2^n$ with $n \in [2, 11]$. Results from each fold are combined and normalized in the interval $[0, 1]$. Lighter color indicate high value.

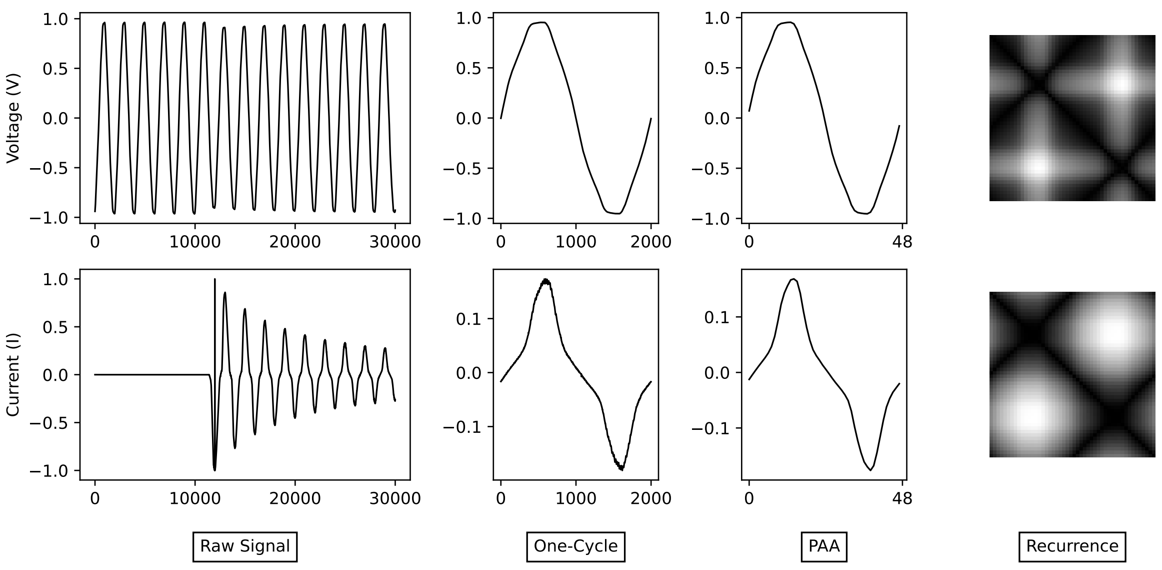


Figure 6.4: Feature extraction process, shown for two different signals: First a voltage zero crossing aligned cycle is extracted. The cycle is then subsampled using a Piecewise Aggregate Approximation (PAA) in order to reduce dimensionality. The voltage and current cycles are then transformed into a Recurrence Plot.

6.4 A High Sample Rate Neural Network Method

In the following a high sample rate appliance identification method called Recurrence Plot Spacial Pyramid Pooling (RPSPP) is discussed. The approach uses V-I trajectory, and a similar approach has been proposed by [Faus 21, Faus 20]. While their approach requires handcrafted parameters for each tested dataset, the approach presented here relies on the same parameters for each tested dataset while having equivalent or superior classification performance. The algorithm is a highly generalized appliance identification algorithm evaluated in an ILM 3 scenario, based on parameter-free recurrence plot. The calculation of the recurrence plot is implemented as a Neural Network layer, simplifying the processing pipeline and improving the required computation time by using the GPU. Evaluation is done on three public high sample rate datasets: COOLL, PLAID, and WHITEDv1.1 [Pico 16, Gao 14, Kahl 16].

6.4.1 Method

The method uses the V-I trajectory as described in Section 4.2.2. The V-I trajectory is transformed into two unthresholded RPs. The classification is performed using a Spacial Pyramid Pooling Convolutional Neural Network.

Features

The feature extraction process is shown in Fig. 6.4. First, one cycle is extracted from the data by searching for zero crossings in the voltage signal using the algorithm in Listing 6.1. In the case of the datasets used for the evaluation of this approach, the first step is a search for a steady-state cycle in the signal. This is done by taking 20

Listing 6.1: Python notation of search algorithm for single voltage and current cycle

```
# Input:
#  v_data = Voltage data
#  i_data = Current data
#  period_length = period length of one cycle

# Find all positive values
positive = v_data > 0

# Find Zero crossing indices
zc_idx = bitwise_xor(positive[1:], positive[:-1])

# Start on uphill slope
if v_data[zc_idx[0]] > v_data[zc_idx[0] + 1]:
    zc_idx = zc_idx[1:]

# Check for even number of zero crossings
if len(zero_crossings) % 2 == 1:
    zc_idx = zc_idx[:-1]

# Assure full period for last zero crossing
if zc_idx[-2] + period_length >= len(v_data):
    zc_idx = zc_idx[:-2]

# Choose last cycle
start = zc_idx[-2]
stop = zc_idx[-2] + period_length

v_cycle = v_data[start:stop]
i_cycle = i_data[start:stop]
```

cycles after the switch-on event and then using the last full cycle available by looking for the zero crossing in order to ensure we are in steady-state. The corresponding current measurements are extracted in alignment with the voltage zero crossings.

In order to reduce the number of data points, each cycle is scaled to 48 values using a PAA [Keog 01]. PAA is a fast dimensionality reduction algorithm, that approximates pieces of the signal by computing the mean value of equal frames of the original signal. Here, the implementation provided by the Python package *pyts* is used [Faou 20].

After the PAA as been applied, the reduced signals are then turned in two RPs. Recurrence is a property of many natural processes and systems. States that have been observed more than once are often followed by similar states. These recurrences can be represented in a recurrence matrix $\boldsymbol{R} = (R_{i,j})$ defined as:

$$R_{i,j}(\epsilon) = \Theta(\epsilon - |x_i - x_j||), i, j = 1, ..., N \quad . \tag{6.16}$$

N represents the number of samples x_i, ϵ is a threshold, and Θ the Heaviside or step function:

$$\Theta(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases} \quad . \tag{6.17}$$

For states that are in the ϵ-neighborhood the following applies:

$$x_i \approx x_j \iff R_{i,j} \equiv 1 \quad . \tag{6.18}$$

Plotting the recurrence matrix as a black and white image produces a visual representation called RP, that can be used for qualitative assessment by humans

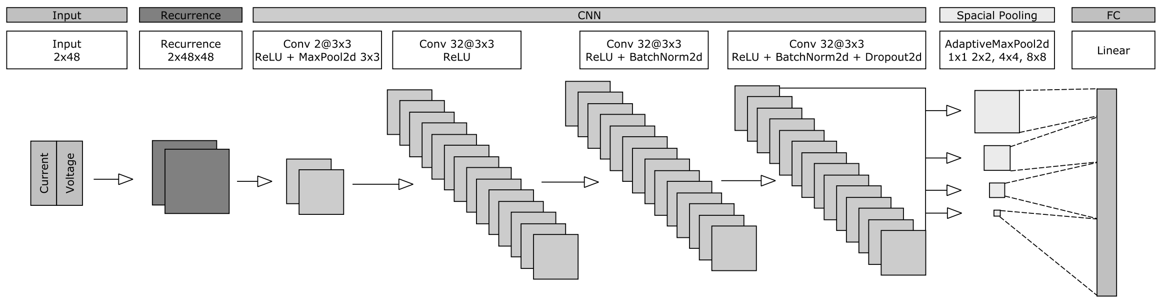

Input
Recurrence
CNN
Spacial Pooling
FC
Input 2x48
Recurrence 2x48x48
Conv 2@3x3 ReLU + MaxPool2d 3x3
Conv 32@3x3 ReLU
Conv 32@3x3 ReLU + BatchNorm2d
Conv 32@3x3 ReLU + BatchNorm2d + Dropout2d
AdaptiveMaxPool2d 1x1 2x2, 4x4, 8x8
Linear
Current
Voltage

Figure 6.5: Deep Neural Network used for appliance identification. The network is composed of four parts: Recurrence, CNN, SPP, and FC.

[Marw 07]. When using powerful classification algorithms, such as Neural Network (NN), the complexity reduction introduces by the threshold ϵ and step function Θ may not be desirable. Behavior is similar to cutting off or binarizing with an ϵ parameter can be learned by a NN. A denser representation can be obtained by calculating the pairwise distances to obtain a distance matrix $\boldsymbol{D} = (D_{i,j})$:

$$D_{i,j} = |x_i - x_j| \quad . \tag{6.19}$$

These pairwise distances are then plotted. This modification of the RP is also known as *unthresholded recurrence plot* or *distance plot* [Marw 07]. The Euclidean distance is used for the distance calculation. The RP processing steps are as described in [Wenn 19a], using a threshold cut-off at three times the standard deviation σ of all distances in $\boldsymbol{D}$:

$$D'_{i,j} = \begin{cases} 3\sigma & D_{i,j} \geq 3\sigma \\ D_{i,j} & D_{i,j} < 3\sigma \end{cases} \quad . \tag{6.20}$$

The developed PyTorch recurrence plot implementation is available online*. It implements (6.19) as a Compute Unified Device Architecture (CUDA) kernel, allowing parallel computation of the RP on a Graphics Processing Unit (GPU).

Unthresholded recurrence plots have been shown to work as feature inputs for a vast range of time series classification problems [Wenn 19a]. [Faus 20] use a Weighted Recurrence Plot (WRG) introducing a weight parameter δ that is used to tune the cut-off. Both parameters are also fine-tuned as parameters during NN training [Faus 21].

Classification

The generated recurrence plots D' are classified using a CNN combined with Spacial Pyramid Pooling (SPP). Using a CNN and SPP has been shown to work effectively in visual recognition tasks [He 14]. Figure 6.5 shows the full network architecture, which consists of four blocks: Recurrence, CNN, SPP, and a fully connected (FC).

The network inputs are the two channels, voltage and current, which have been resized using PAA to a vector of length 48. Each channel is converted into a RP which is then fed into a series of CNN layers. The CNN part of the network contains four layers. All CNN layers use a 3×3 kernel and have a Rectified Linear Unit

*https://github.com/walwe/pytorch-recurrence

Listing 6.2: Calculation of stride and kernel size for adaptive max pooling

```
stride = in_size // out_size
kernel_size = in_size - (out_size-1)*stride
```

(ReLU) activation function. The first layer has the same number of input and output channels and a two-dimensional max pooling layer of size 3×3. The other CNN layers all use 32 channels. Two-dimensional batch normalization is applied in the last two layers. The last CNN layer also has a two-dimensional dropout with a probability of 0.2 applied. The batch normalization as well as the dropout proved to be crucial for the network to generalize quickly and to prevent overfitting.

The SPP layer is a combination of four two-dimensional adaptive max pooling filters of different sizes (1×1, 2×2, 4×4, 8×8). Adaptive max pooling is a dimensionality reduction mechanism that outputs a fixed size vector by adapting the max pooling kernel size. In our case, the output size of the CNN block and therefore the input size of the SPP layer is of size $32 \times 16 \times 16$, where 32 is the number of CNN channels. The size has been reduced from 48×48 to 16×16 by the 3×3 max pooling in the first layer of the CNN block. Since 16×16 is an integer multiple of the SPP filter sizes, adaptive max pooling is implemented by calculating the stride and kernel size as shown in Listing 6.2.

The max pooling filter outputs are stacked to form a new vector. The SPP layer, therefore, outputs a fixed length vector of size $32 \cdot (1 \cdot 1 + 2 \cdot 2 + 4 \cdot 4 + 8 \cdot 8) = 2720$, where 32 is the number of channels produced by the last CNN layer. $1 \cdot 1, 2 \cdot 2, \ldots$ are the output sizes of the adaptive max pooling layers applied to the channels. The layer has the purpose of recognizing different features in the recurrence plot by partitioning the plot into finer and coarser levels. This way, the network is capable of recognizing features in the plot of different sizes. The flattened and stacked SPP layer output is then fed into the FC layer. The SPP layer also works as a dimensionality reduction before the FC layer. The last layer's output size is equivalent to the number of classes to be classified.

6.4.2 Evaluation

For direct comparison, the evaluation was done in close alignment with the evaluation presented by [Faus 20]. The evaluation presented here is as described by the authors and their experiment has been reproduced as closely as possible by using the same dataset-split approach and random seed.

Data

The data used for evaluation of the proposed appliance identification method are three public datasets: COOLL, PLAID and WHITEDv1.1 [Pico 16, Gao 14, Kahl 16]. These datasets have been widely used for the evaluation of appliance identification methods.

COOLL (Controlled On/Off Loads Library) is a dataset containing controlled On/Off loads of 42 appliances of 12 categories. The data was recorded at 100 kHz sampling frequency, which is a very high sampling rate compared to other datasets. The dataset contains 20 samples per appliance, recorded in a laboratory in France. Similar to other publications, only the appliance categories with at least 2 appliances and therefore 40 samples are used. The categories used are Drill, Fan, Grinder, Hair Dryer, Hedge Trimmer, Lamp, Sander, Saw, and Vacuum Cleaner.

The second dataset used is PLAID 2017 (Plug Load Appliance Identification Dataset). The dataset was collected in the US. It was recorded at a sampling frequency of 30 kHz. It contains a total of 1793 records from 11 different appliance categories.

The third dataset used for evaluation is WHITEDv1.1 (Worldwide Household and Industry Transient Energy Data Set). It contains 1259 recordings of 47 appliance categories with a total of 110 different appliances. The dataset contains the first 5 seconds of the appliance start-ups, recorded at a sampling frequency of 44.1 kHz. Data from 21 of the available categories is used.

Validation

The evaluation is performed in a stratified k-fold and a leave-one-group-out cross-validation. The latter is done in order to test the generalization performance. The PLAID dataset provides groups in the form of locations called houses 1–55 that is used for the leave-one-group-out validation. COOLL and WHITED do not provide such groups as part of the dataset. Therefore, the appliances were assigned to random groups based on their provided appliance name as proposed by De Baets [De B 18b]. The COOLL dataset for example provides appliances' names in the form of: Drill_1, Drill_2, . . . , Drill_6. The maximum number of different appliances in one category is 8, thus the number of created groups/houses for COOLL is 8. A new label in the range 0–7 is randomly assigned to the 6 Drill groups. Therefore 6 of the 8 houses contain a Drill, but no house will share the exact same Drill, resulting in a real generalization task. This grouping results in 8 groups for COOLL and 9 for WHITEDv1.1. Faustine et al. [Faus 20] present their WRG model results, claiming to use the same data split approach. Based on a review of the published source code*, a conclusion is drawn: the results are not based on the group split described in the study, but rather a k-fold-like split. Therefore, corrected results for the WRG algorithm based on the new evaluation are also presented.

The final evaluation is done using a macro F_1-score (see Section 3.5.4). The macro F_1-score is calculated per fold and finally averaged over all folds:

$$F_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{N}\sum_{a=1}^{N}F_{k,a} \tag{6.21}$$

where K is the number of folds, N the number of appliance categories and $F_{k,a}$ the F_1-score of each category a in fold k.

Results

The results of the evaluation of the appliance identification task show an improvement on COOLL and PLAID in both evaluations in comparison to De Baets [De B 18b] and

*https://github.com/sambaiga/WRG-NILM

Table 6.2: Leave-One-Group-Out results (F_{macro}) on COOLL, WHITEDv1.1, and PLAID. WRG results have been re-evaluated. De Baets' results are presented as in their publications.

Algorithm	COOLL	WHITEDv1.1	PLAID
De Baets [De B 18b]	N/A	**0.7546**	0.7760
WRG [Faus 20]	0.447	0.3954	0.8921
RPSPP	**0.5329**	0.4310	**0.8942**

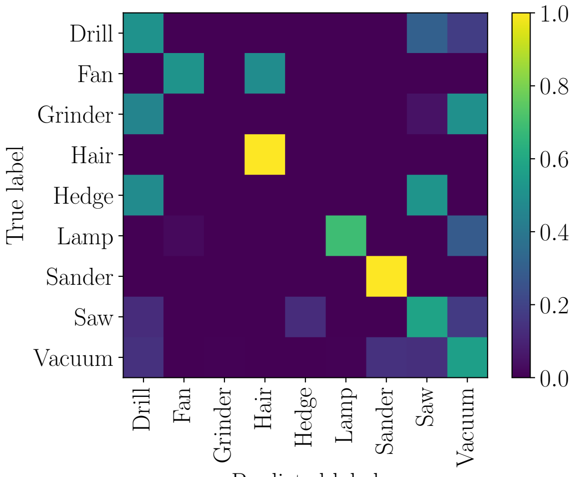
Table 6.3: 5-fold cross validation results (F_{macro}) of our experiments on COOLL, WHITEDv1.1, and PLAID. De Baets is not listed as they do not provide results.

Algorithm	COOLL	WHITEDv1.1	PLAID
WRG [Faus 20]	0.8957	**0.9984**	0.8082
RPSPP	**0.9213**	0.9924	**0.8456**

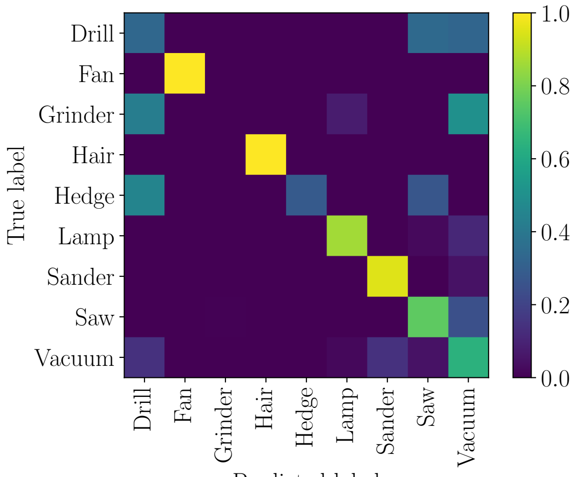
WRG [Faus 20]. Results are presented in Table 6.2 and 6.3. The proposed Spacial Pyramid Pooling CNN network is capable of achieving the highest F_1-score for the PLAID and COOLL datasets in both evaluation scenarios. The De Baets results are presented as published by the authors. The authors did not publish results for the COOLL dataset. As they did not publish their source code, it cannot be assured that their experimental setup is comparable. The re-evaluation of the WRG algorithm shows very different results compared to the authors' published results due to the difference in generating the groups for the leave-one-group-out evaluation.

The leave-one-group-out evaluation of our approach for COOLL and WHITEDv1.1 results in a F_1-score of 0.5329 and 0.4310. While on COOLL the RPSPP approach achieves a higher score compared to WRG, the absolute scores are low. The confusion matrix in Fig. 6.6 shows that the Drill, Grinder, Hedge Trimmer, Saw, and Vacuum are frequently confused. The current signature of all these appliances can in some cases be very similar, leaving no grounds for separation. The Grinder category cannot be classified at all by either of the algorithms, the WRG also fails to classify the Hedge Trimmer. The new RPSPP approach can improve the results for two categories (Hair Dryer and Fan) to a perfect score of 1.0. On WHITEDv1.1 the approach presented by De Baets achieves the highest score of 0.7546. Similarly to the COOLL dataset, both WRG and RPSPP do not result in good scores. The main reason can be found in the per class F_1-score in Fig. 6.9. The WRG algorithm completely fails to classify 7 out of the 21 classes with a F_1 of 0.0. The RPSPP does not classify 6 out of the 21 classes. Since this is not the case in the 5-fold cross-validation shown in Fig. 6.10, it must be assumed that in the case of the WHITED dataset, both algorithms do not learn a generalized representation of these classes. The confusion matrix in Fig. 6.7 shows that the CFL class is always misclassified as Light Bulb by the RPSPP approach. In general, there are some clear patterns of misclassifications resulting from evaluation folds where the training split was insufficient for the test split.

On PLAID both WRG and the here presented approach outperforms De Baets with an average F_1-score of 0.8921 and 0.8942. Figure 6.8 shows that Air Conditioner and Fan are getting confused as well as Hair Dryer and Heater. The misclassification



(a) WRG



(b) RPSPP

Figure 6.6: Confusion matrix of leave-one-house-out test split result on COOLL dataset. Cross-validation results have been aggregated.

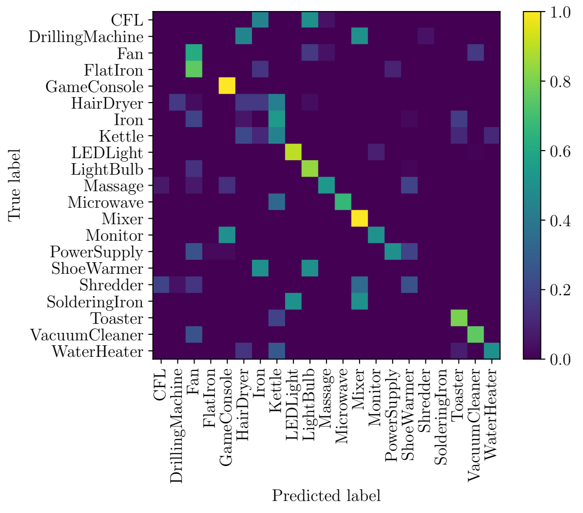
pattern shows that very similarly operating appliances are hard to separate by their V-I trajectory. Air Conditioner and Fan both have a similar ventilation mechanism. Hair dryer and Heater both use a heating coil. The V-I trajectory recurrence plot seems to not provide a good enough feature in order to separate these very close appliance categories. Figure 6.9 reveals that the new approach particularly improved the results for the worst-performing categories: Air Conditioner, Fridge, and Heater.

The results on the 5-fold cross-validations are much better. For the COOLL and PLAID datasets, the new RPSPP approach outperforms the WRG algorithm. On COOLL RPSPP achieves a F_1-score of 0.9213 while WRG achieves 0.8957. On WHITEDv1.1 both approaches achieve near-perfect scores, where WRG has a score of 0.9984 and RPSPP 0.9924. The RPSPP score for the PLAID dataset is 0.8456 compared to WRG with a score of 0.8082.

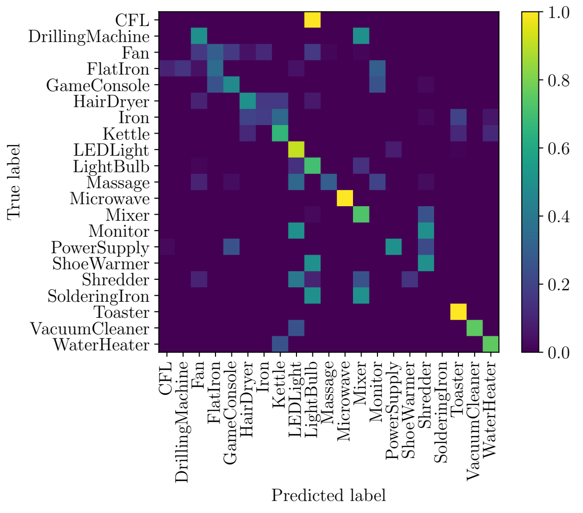
The evaluation using the leave-one-group-out for the COOLL and WHITEDv1.1 dataset is a challenging task for NNs as the datasets are small and the risk of overfitting is high. While the 5-fold validation results in very good scores, the leave-one-house-out task only results in good scores for PLAID, likely due to the PLAID dataset being much larger compared to the others.

6.5 Summary

In this chapter, a low and high sample rate appliance category identification algorithm has been described. Both approaches have been designed for ILM 3 scenarios, where each appliance is monitored independently. Such scenarios are less complex than, e. g., NILM scenarios. With the recent advancing of Internet of Things (IoT) technology, ILM 3 scenarios become more and more realistic as the monitoring hardware for individual appliances becomes more affordable.

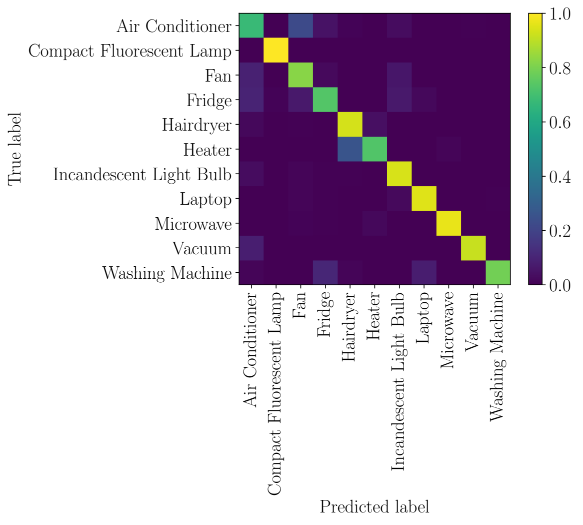


(a) WRG

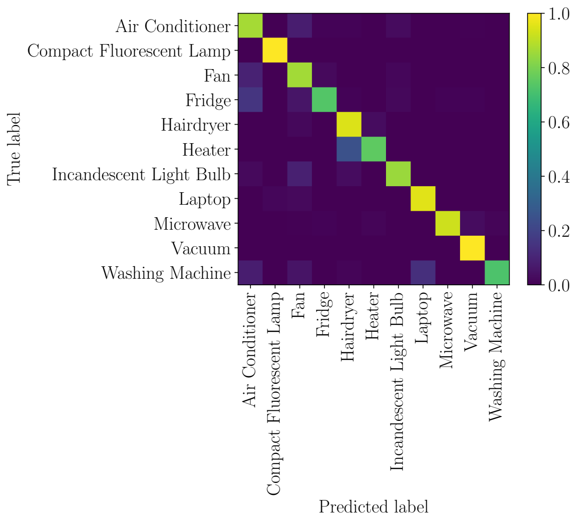


(b) RPSPP

Figure 6.7: Confusion matrix of leave-one-house-out test split result on WHITEDv1.1 dataset. Cross-validation results have been aggregated.



(a) WRG



(b) RPSPP

Figure 6.8: Confusion matrix of leave-one-house-out test split result on PLAID dataset. Cross-validation results have been aggregated.

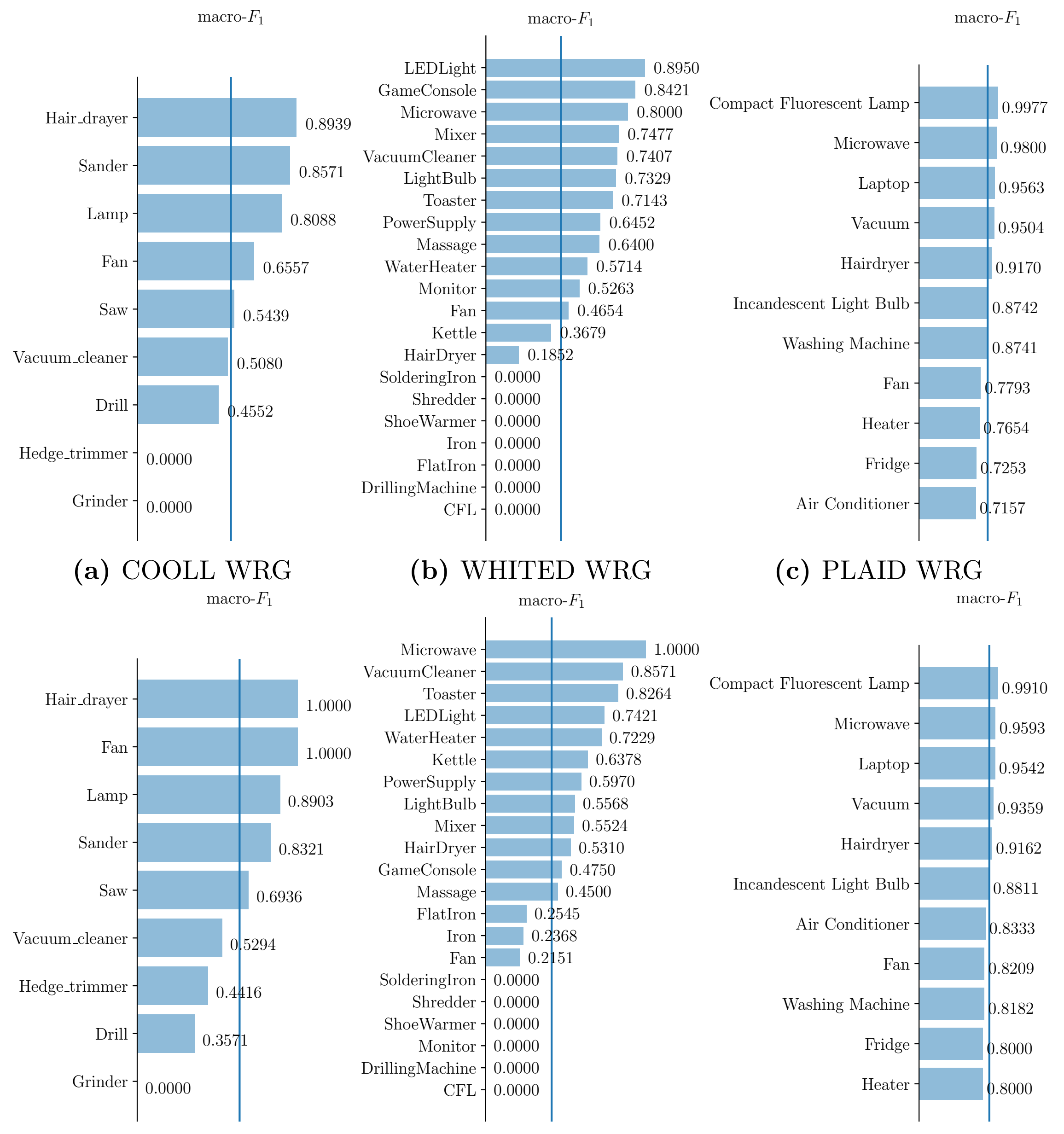
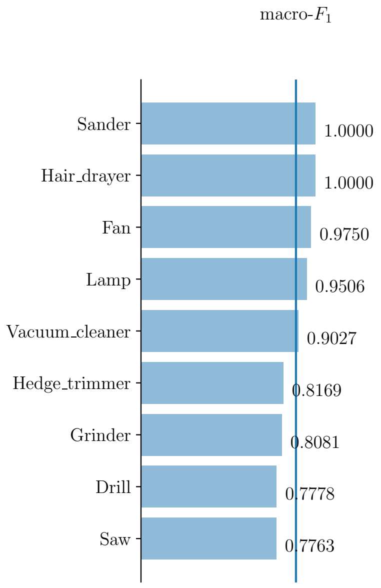
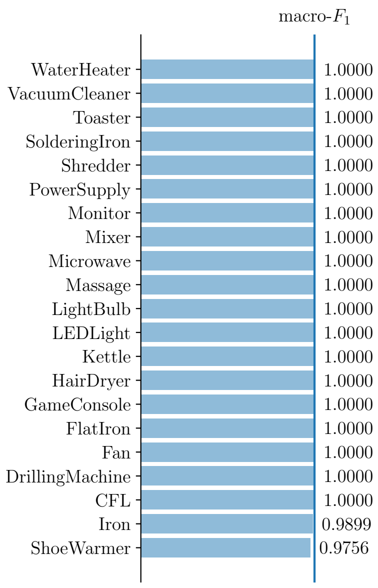


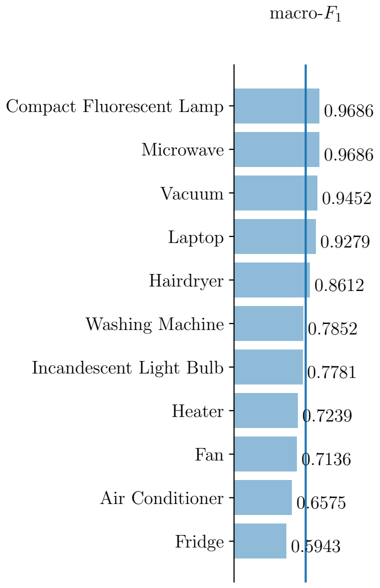
Figure 6.9: F_1-score per category in the leave-one-group-out validation. (a)–(c) show the WRG approach and (d)–(f) the here presented RPSPP approach.



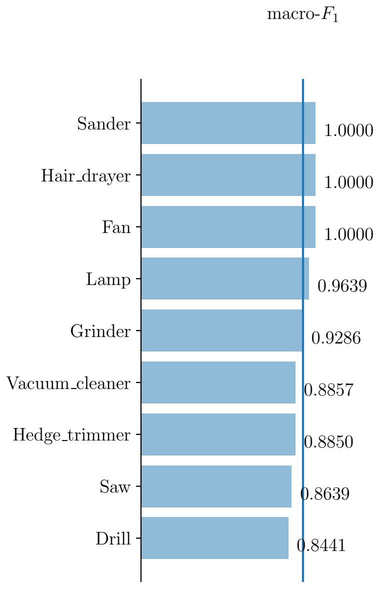
(a) COOLL WRG



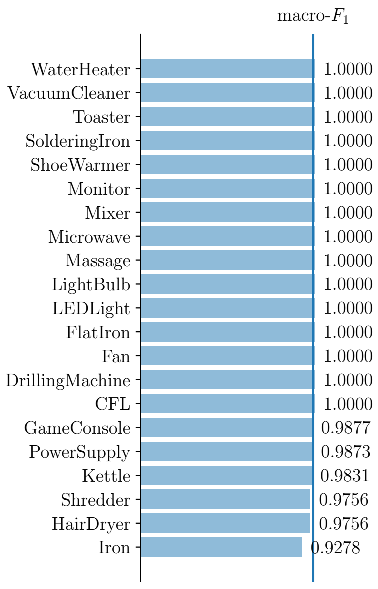
(b) WHITED WRG



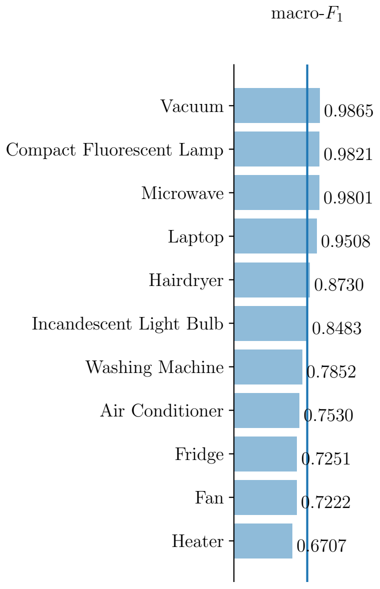
(c) PLAID WRG



(d) COOLL RPSPP



(e) WHITED RPSPP



(f) PLAID RPSPP

Figure 6.10: F_1-score per category in the 5-fold cross validation. (a)–(c) show the WRG approach and (d)–(f) the here presented RPSPP approach.

The low sample rate approach described is based on the commonly used RMS averaged power value (cf. Section 6.2.1). The evaluation on the DEDDIAG dataset (cf. Chapter 5) shows that the identification improves with increasing window sizes. The approach has been published as a challenge as part of the dataset [Wenn 21b]. The evaluation presented here was performed using an improved data split approach. This made the task more challenging and the results slightly less reliable compared to the published results. In the previous publication, the k-fold split has been done on window bases and not based on the appliance cycles. When using overlapping sliding windows, this breaks the required strict separation of train and test data.

An average F_1-score of 0.91 can be achieved when classifying based on 2048 seconds of data. This means that it requires about half a minute to reliably identify an appliance category. In cases where the features of an appliance are very unique, such as a Coffee Machine, even a 4-second window is sufficient to determine the category. It could therefore be shown, that a kNN approach and low sample rate data provide sufficient information for the identification step in the MLDR.

Furthermore, a high sample rate approach has been developed called RPSPP. The approach is an improvement to a method described by Faustine et al. [Faus 20] known as WRG. It uses the V-I trajectory features which are transformed into a RP. The classification is done using a SPP deep neural network. Unlike the WRG approach, the presented algorithm does not rely on dataset-specific parameter tuning. The evaluation was performed using three public datasets: COOLL, WHITEDv1.1, PLAID. It was performed using a 5-fold cross-validation as well as a leave-one-group-out validation. For the 5-fold cross-validation presented in Table 6.3, the RPSPP algorithm results in superior F_1-scores compared to WRG for COOLL and PLAID, and a comparable result for WHITEDv1.1. On WHITEDv1.1, a near-perfect score of 0.9924 was achieved. The leave-one-group-out validation is presented in Table 6.2, the result in this case is different. Performing a validation such as this tests the generalization capability of the algorithms. For COOLL and WHITEDv1.1, only F_1-scores of 0.53 and 0.43 were achieved. For the PLAID dataset, a good score of 0.89 was achieved by both WRG and RPSPP. The results for the WRG approach has been re-evaluated here and yield very different results compared to the one published by the authors [Faus 20]. The data split implemented and published by the authors does not implement the logic described in the publication. For the WHITEDv1.1 datasets, the approach published by De Baets et al. [De B 18b] outperformed both WRG and RPSPP. It should be noted that their result has not been re-evaluated and comparability is uncertain. It is concluded, therefore, that for a large dataset such as PLAID, the approach presented works reliably, even in a leave-one-group-out scenario. On a 5-fold cross-validation, the approach is very reliable.

It could be shown in the evaluation that both low and high-sample rate approaches are valid approaches to appliance identification. Which option is recommended in real-life scenarios will therefore be a question of the speed with which an appliance needs to be identified and what type of hardware will be made available for mass markets.

CHAPTER 7

Appliance Segmentation

Substantial parts of this chapter's Section 7.4 and 7.5 have been published in the following manuscript: [Wenn 21b]. Substantial parts of Section 7.6 have been published in the following manuscript: [Wenn 19b]. The majority of the manuscript has been authored by the author of this thesis. The main scientific contributions are based on his own work and thoughts.

7.1 Introduction

After the appliance category has been determined, on cycle-based appliances, the starting and stopping time of each cycle can be extracted. In the Machine Learning Demand Response Model (MLDR), this process is called segmentation and leads to a list of appliance-specific start-stop timestamps and associated load patterns. Based on this, the individual load profile γ of each appliance can be determined as the electricity demand between the start and stop event. The result is a list of ON (start) and OFF (stop) timestamp pairs (t_0, t_1) and the associated load profiles γ. The ON/OFF events are the basis on which to build a usage pattern. Examples of start and stop annotations for different appliances are shown in Fig. 7.1.

The usage pattern and load profiles are the foundation for forecasting a household's load and for providing recommendations or automatically performing actions. These event detection tasks are often performed using thresholding methods [Girm 16]. In the following, a simple lower bound thresholding algorithm, as well as a Support Vector Machine (SVM) based algorithm, are discussed. Both algorithms classify based on a sliding window over the power measurements. Using a sliding window helps to overcome the fact that load measurements are a continuous time series that

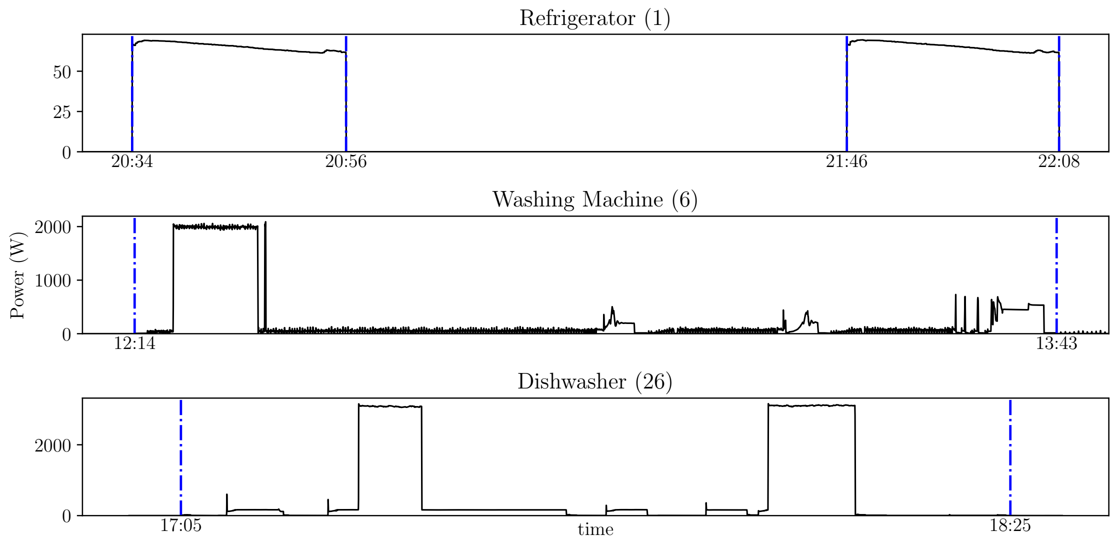


Figure 7.1: ON/OFF-annotation examples from three appliances with id 1, 6, 26 in the DEDDIAG dataset. The blue line shows the provided ground truth annotation. It can be seen that the segmentation of the refrigerator is a much simpler task compared to the washing machine or dishwasher.

contain too much data to be classified all at once. This sliding window use also helps achieve the default series length that is required. The use of a sliding window will also help to further develop such methods into a streaming method. The two algorithms are based on two fundamentally different approaches. The thresholding algorithm predicts the appliance being ON or OFF (steady state) for each window. The SVM approach predicts the transient state between ON and OFF, so it predicts the actual switching event. In both cases, the event is derived using a heuristic that connects the ON and OFF event. The evaluation shown here is performed to represent the difference of the two approaches and their advantages and disadvantages.

Until the publication of Domestic Energy Demand Dataset of Individual Appliances in Germany (DEDDIAG), described in Chapter 5, no low sample rate public datasets that provided ground truth was available. Thus the evaluation of most publications does not, in essence, predict events, but rather the presence of electricity consumption. The DEDDIAG dataset provides manual event annotations that can act as ground truth to train and evaluate appliance segmentation algorithms. An overview of the available annotations is given in Appendix A.2.

7.2 Switching Event Definition

In literature, two different understandings of events in terms of Demand Response (DR) exist: edges events and switching events. The problem with different event definitions has already been addressed in Section 4.2.3. In terms of segmentation, the term events are always understood as switching events, and for simplicity in the following event is equivalent to a switching event. Event annotation is denoted as a segmentation task where an event segment $e = (t_0, t_1)$ is defined by the interval between two timestamps t_0, t_1, $t_0 < t_1$ during which an appliance is running, e. g., a washing machine being started at t_0, finishing a full washing cycle at t_1 (note that the appliance might consume only a very small amount of power at certain times while running, which may make it very hard to distinguish this case from the actual end of the cycle). For most appliances, such as washing machines and dishwashers, there are no overlapping segments. This is also the case if there are different labels defined for one appliance, e. g., PreWash and Normal. For refrigerators and freezers, overlap between the light and compressor labels is possible, but overlap within one label cannot occur.

7.3 Related Work

Giri et al. present a two-step approach. First, they find areas of interest based on a threshold step change in real power, e. g., 50 Watts [Giri 13]. If the power averaged over the first two seconds is above 50% of the event's threshold the area is labeled as ON. For OFF events the chosen average is 80%. This is described as a pre-processing step for the appliance identification task. The performance of the approach is unknown, as the publication does not present an evaluation.

Yang et al. use goodness-of-fit (GOF) event detection combined with ON/OFF pairing as described by [Giri 13] followed by k-means clustering [Chua 15]. They also do not present a numeric evaluation.

Kahl et al. propose a multivariate event detection approach [Kahl 19]. They use six different high sample rate features: current, Δ(current), admittance, spectral flatness, cumulative sum, Δ(cumulative sum). They evaluate the separation of their features using a k-Nearest-Neighbor (kNN) and SVM on the BLOND-50 and BLUED datasets. Since both datasets provide high sample rate data of 50 Hz and 12 kHz, their approach requires high sample rates. Overall they achieve the best F_1-score for the BLUED dataset using the Δ(cumulative sum) with a kNN using a MinMax normalization. Their overall achieved F_1-score for the BLUED dataset is 0.78. On the BLOND-50 dataset, the best result was achieved using the cumulative sum and a SVM classifier. Using a variance normalization, their overall F_1-score is 0.67.

Rollins et al. describe an activity annotation collection system to assist in collecting usage annotations. The authors conclude that a simple lower bound thresholding algorithm is not sufficient to annotate switching events: "We deployed this preliminary algorithm for two houses in our study and discovered that while the 20 W threshold was appropriate for all devices in the first house, it was not appropriate for all devices in the second". The authors propose to identify load patterns using a density-based spatial clustering of applications with noise (DBSCAN) algorithm to cluster different power states to overcome manual threshold parameter tuning [Roll 14]. The authors also describe a feedback loop to end-users using a smartphone app and push notifications and ask the user to validate segmentation and gather data to retrain.

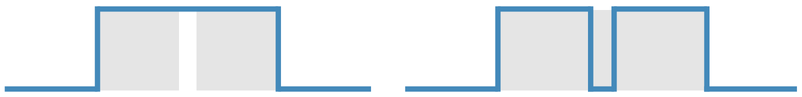


Figure 7.2: False annotations that are very mildly punished when evaluating per timestamp using F_1-score or similar. The blue line shows the real event segment, the gray area shows the predicted event segment, where on the left the real event segment is predicted as two segments, and on the right two real event segments are predicted as a single one. Using the Jaccard-Time-Span-Event-Score (JTES) both cases will result in a score ≤ 0.5.

7.4 Jaccard-Time-Span-Event-Score (JTES)

Since in a Non-Intrusive Load Monitoring (NILM) context events are commonly defined by a single timestamp, the evaluation of such events is usually done based on the correctness of the label for each time step using standard metrics such as true positive rate, false positive rate, accuracy or F_1. For usage analysis or event attribution, these metrics are problematic because all such events are extremely rare, especially when only annotating the start or end. To put it in perspective: an appliance that is only switched on once per day will, in a dataset recorded with 1 Hz, result in having to find one value within 86400 data points. Evaluating an appliance based on ON/OFF status for each available time-step will for long-running appliances result in a much more balanced task. While for electricity attribution tasks such as disaggregation, this evaluation can be seen as fair, for appliance usage analysis it is still not strict enough. A per timestamp evaluation will not provide a strong enough penalty for many unwanted annotations such as splitting or combining of event segments as shown in Fig. 7.2.

For usage analysis, a split or combined event segment will result in wrong overall event counts, wrong usage lengths, and thus a false analysis of usage behavior. As part of the development of appliance segmentation algorithms (see Section 7.5, we have proposed a new metric called JTES [Wenn 21b]. The metric is based on the Intersection-over-Union (IoU) metric, also known as the Jaccard index. The Jaccard index computed on a single segment is:

$$\mathrm{iou}(e_\mathrm{T}, e_\mathrm{P}) = \frac{\mathit{Intersection}}{\mathit{Union}} = \frac{\min(t_{\mathrm{T}1}, t_{\mathrm{P}1}) - \max(t_{\mathrm{T}0}, t_{\mathrm{P}0})}{\max(t_{\mathrm{T}0}, t_{\mathrm{P}0}, t_{\mathrm{T}1}, t_{\mathrm{P}1}) - \min(t_{\mathrm{T}0}, t_{\mathrm{P}0}, t_{\mathrm{T}1}, t_{\mathrm{P}1})} \quad , \quad (7.1)$$

where e_T is the true event segment and e_P is the predicted one; this basically assumes that a box of height one is used for comparing regions. The Jaccard index is widely used to evaluate image segmentation tasks such as object detection, where commonly true positives are defined as having an overlap of more than 50% [Ever 10], which are then used to calculate an overall accuracy score. The JTES avoids setting an arbitrary threshold and therefore eliminates the drawbacks of using pure IoU. In the first step, we compute an average IoU for each true event segment, only taking into account predictions that actually have at least a partial overlap with the true event segment. Let $e_{\mathrm{T}i}, i = 0, \ldots, N_\mathrm{T} - 1$ be a true event segment and $e_{\mathrm{P}j}, j = 0, \ldots, N_\mathrm{P} - 1$

a predicted segment, where $N_\mathrm{T}, N_\mathrm{P}$ is the number of true and predicted segments, respectively. The average IoU A_i for true event i is then given by

$$A_i = \frac{1}{N_{\mathrm{NZ}i}} \sum_{j=0}^{N_\mathrm{P}-1} \mathrm{iou}(e_{\mathrm{T}i}, e_{\mathrm{P}j}) \quad , \tag{7.2}$$

where $\mathrm{iou}(e_{\mathrm{T}i}, e_{\mathrm{P}j})$ computes the IoU for the segments $e_{\mathrm{T}i}$ and $e_{\mathrm{P}j}$ as defined in (7.1), and $N_{\mathrm{NZ}i}$ is the number of non-zero IoU-values in the sum (i. e. the predictions with an overlap). The final score is then calculated by summing all A_i and normalization to the number of true event segments N_T corrected by the number of false positive predictions N_FP:

$$\mathrm{JTES}(\boldsymbol{e}_\mathrm{T}, \boldsymbol{e}_\mathrm{P}) = \frac{\sum_{i=0}^{N_\mathrm{T}} A_i}{N_\mathrm{T} + N_\mathrm{FP}}, \tag{7.3}$$

where

$$\begin{aligned} \boldsymbol{e}_\mathrm{T} &= (e_{\mathrm{T}0}, e_{\mathrm{T}1}, \ldots, e_{\mathrm{T}N_\mathrm{T}-1}) \, , \\ \boldsymbol{e}_\mathrm{P} &= (e_{\mathrm{P}0}, e_{\mathrm{P}1}, \ldots, e_{\mathrm{P}N_\mathrm{P}-1}) \quad . \end{aligned} \tag{7.4}$$

The JTES requires non-overlapping true event segments, i. e. $e_{\mathrm{T}i} \cap e_{\mathrm{T}j} = \emptyset \; \forall i, j, i \neq j$. It is designed to reflect real-world expectations on event annotation algorithms, where an event that is split in half will at best only be evaluated as half correct, the same applies to scenarios where two true events have been merged into one event as shown in Fig. 7.2. Splitting a real event in two successive events will result in a JTES of at most 0.5, while common scores such as accuracy or F_1 will evaluate each time step independently and still result in perfect scores.
The principles of JTES are:

- Score is in range [0, 1], where 0 is lowest and 1 best,
- false positives and false negatives are equally bad,
- if a true event spans over multiple predicted events, the result is averaged,
- if the predicted event spans over multiple true events, the result is accounted for accordingly,
- if $N_\mathrm{T} = 0$ and $N_\mathrm{P} > 0$, the score is 0,
- if $N_\mathrm{T} > 0$ and $N_\mathrm{P} = 0$, the score is 0,
- if $\boldsymbol{e}_\mathrm{T} = \boldsymbol{e}_\mathrm{P}$ (assuming the segments are ordered by start time), the score is 1.

A python reference implementation, called python-jtes, has been published on github.com*. Listing 7.1 shows a usage example of python-jtes.

*https://github.com/deddiag/python-jtes

Listing 7.1: Usage example of the reference implementation of the JTES called python-jtes.

```python
import numpy as np
from jtes import jaccard_timespan_event_score

y_true = np.array([
    (np.datetime64('1900-01-01T00:00:00'), np.datetime64('1900-01-01T01:00:00')),
    (np.datetime64('1900-01-01T03:00:00'), np.datetime64('1900-01-01T04:00:00'))
])
y_pred = np.array([
    (np.datetime64('1900-01-01T00:00:00'), np.datetime64('1900-01-01T01:00:00')),
    (np.datetime64('1900-01-01T03:00:00'), np.datetime64('1900-01-01T05:00:00')),
])
# Returns 0.75
jaccard_timespan_event_score(y_true, y_pred)
```

7.5 A Lower Bound Thresholding Method

The lower bound thresholding method is the simplest and most commonly used algorithm for appliance switch event segmentation. As already concluded by Rollings et al., it is not sufficient for many appliances and does not generalize [Roll 14]. It has been implemented as a challenge baseline for the DEDDIAG dataset [Wenn 21b].

A lower bound thresholding algorithm defines a single threshold to separate the ON/OFF states. The principle of a threshold algorithm is, that the ON and OFF state is separable by a distinctive electricity demand. This means, that while the appliance is ON, the electricity level must stay above the threshold or it will lead to the fragmentation of a single event. As shown in Fig. 7.1, the existence of such a threshold is not guaranteed, as in between one washing machine or dishwasher cycle, the power demand can be very low.

Lower bound threshold approaches will only work in Intrusive Load Monitoring (ILM) 3 scenarios where only one appliance is recorded at a time. The major advantage is the implementations and computational simplicity.

7.5.1 Method

In the implementation used here, the threshold is calculated on the sliding window's average. For each window, the algorithm determines whether the appliance is switched ON or OFF. The appliance is seen as switched on when the average electricity demand E in a sliding window w reaches a threshold t:

$$E(w) = \begin{cases} 0 & \text{avg}(w) < t \\ 1 & \text{avg}(w) \geq t \end{cases} . \tag{7.5}$$

The threshold is defined as the non-zero minimum average window calculated over each event in the train split. This approach therefore does not determine the edges of the ON and OFF events (t_0, t_1) directly, it classifies each window as ON or OFF. When using an averaged sliding window approach, there are edge cases where t_0 or t_1 are within the window. Since each window has to be associated with a single label, it is defined that cases where t_0 is within the window are still labeled as OFF and cases where t_1 is within the window are seen as ON.

Listing 7.2: Pseudocode of the lower bound algorithm.

```
# input: 2D-Sequence of windowed input data -> rolled_data,
#        2D-Sequence of windowed input labels -> rolled_labels

# Find windows that only contain ON (True)
on_idx = []
for idx = 0 to rolled_labels.length:
    if False not in rolled_labels[idx]:
        on_idx.append(idx)
    end
end

# compute average of each ON window
averages = []
for idx in on_idx:
    a = average(rolled_data[idx])
    if a > 0:
        averages.append(a)
end

lower_bound = min(averages)
# output: Lowest, non-zero average value of window with only ON states ->
    lower_bound
```

Listing 7.3: Python implementation of the lower bound algorithm using numpy (np).

```
rolled_average = np.average(data[labels], axis=1)
rolled_average.sort()
lower_bound = next(filter(lambda x: x > 0, rolled_average))  # first-non-zero value
```

Listing 7.2 shows the pseudocode of the used algorithm. Listing 7.3 shows the used Python implementation utilizing the NumPy[*] package. The implementation highlights the simplicity of the approach as it can basically be implemented in two lines of code.

7.5.2 Finding Events

The algorithm does not directly predict events as $e = (t_0, t_1)$, as it only predicts an ON or OFF label for each window. After each window has been classified as ON or OFF, the event e is found by searching for state transitions in the predicted labels. An ON event therefore exists whenever an OFF window is followed by an ON window and vise versa. The algorithm used here works as follows:

1. Find all state transitions from OFF to ON.
2. Find all state transitions from ON to OFF.
3. For each ON event, find the closest OFF event, where $t_{\text{ON}} < t_{\text{OFF}}$.

[*] https://numpy.org

7.6 A Support Vector Machine Method

Next to the thresholding approach that classifies the steady state, a transient state approach was developed. Unlike the thresholding approach where each window is classified as ON or OFF (steady state), the SVMs classifies the transient state between ON and OFF, so the actual start and stop event. For appliances such as washing machines, the start and end timestamps are very rare events. The maximum number of usages per day is given by the usage cycle length, meaning if a cycle takes 3 hours, the maximum number of start events is given by $24h/3h = 8$. Measuring data at 1 Hz will lead to 86,400 measuring points per day, meaning the events looked for are at maximum of 8 timestamps out of 86,400, thus being extremely rare. Since we are using a sliding window, we define the classification task to identify whether a given window contains a start or stop event. The classifier has to classify the two transient states OFF to ON and ON to OFF, thus three classes exist: steady, start, and stop.

The classification task has been tested using Long Short-Term Memory (LSTM) and SVM in preliminary experiments. LSTMs are a type of Recurrent Neural Network (RNN) architecture designed to address the issue of vanishing gradients, which is common in traditional RNNs [Hoch 97]. LSTMs are useful for processing sequential data and have been used in a wide range of applications such as speech recognition, natural language processing, and time-series prediction. Preliminary experiments revealed that both classifiers, LSTM and SVM, produce promising results. Significant improvements were thus not anticipated by selecting a specific classification algorithm, but rather by improving pre-processing and connecting the discovered ON/OFF events. Dominik Stecher then investigated the SVM approach further, providing preliminary experiments on different pre-processing techniques [Stec 20].

7.6.1 Pre-processing

A sliding window is used to convert the continuous time series to individual feature vectors. The data used for training are composed of three parts: start sequence, stop sequence (transient state), and steady-state sequences. The sequences are extracted by taking the annotations from the DEDDIAG dataset. The extracted start and stop sequences are of length $4 \times \text{ws}$, where ws is the window size. The sequences are extracted so that the start and stop lie in the center. Next to these transient state sequences, steady-state measurements are extracted. In general, the steady state is defined as all windows that are not part of the transient state set. As this would also include many zero values whenever the appliance is OFF, the steady state data are defined as the data between the start and stop annotation, without the actual start and stop.

Label definition

As both the start and stop are single points in time, and only a single label per window is desired, the definition of a transient window must be determined. A window is labeled as the transient state when the start or stop timestamp is within

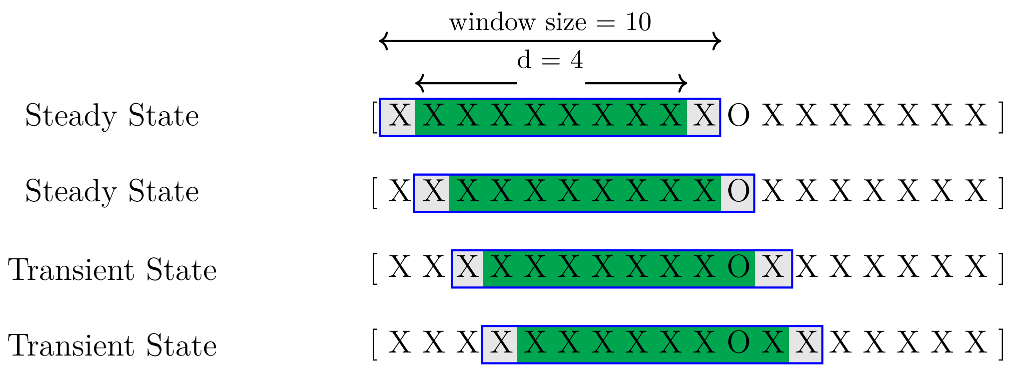

window size = 10
d = 4
Steady State
[X X X X X X X X X X O X X X X X X X]
Steady State
[X X X X X X X X X X O X X X X X X X]
Transient State
[X X X X X X X X X X O X X X X X X X]
Transient State
[X X X X X X X X X X O X X X X X X X]

Figure 7.3: Sliding window over time steps where X marks no event and O a start or stop event. For a window size of 10, the start or stop must be within the 8 green-colored fields of the window for the window to be seen as a transient state.

40% to the left or right of the center of the window (cf. Fig. 7.3), rounded to the nearest integer:

$$d = \lfloor 0.4 \times \mathrm{ws} + 0.5 \lfloor \quad , \tag{7.6}$$

where d is the distance from the center and ws the window size. This definition aims to increase the information available within the window for transient state cases because in many cases the measurements before or after a start are zero. Without such a definition the SVM would have to be able to determine a decision boundary based on a single non-zero value. Without this definition, there would be a decrease in robustness to noise, where only a single electricity spike is measured.

Data Reduction

On long-running appliances, such as a dishwasher, the steady state data available are much larger compared to the transient data, therefore, a limit was placed on the maximum amount of steady and transient state data. In total a maximum of 5,000 windows from the steady state set and 10,000 windows from the transient state are used. The ratio between steady and transient state is chosen to balance the higher feature variation in steady-state while not creating a large class imbalance. These windows are randomly chosen. Now this composed training data set is normalized by removing the mean and scaling it to unit standard deviation. Thus the measurements have a mean of 0 and a standard deviation of 1.

Further, each sliding window of the data is pre-processed using a Discrete Wavelet Transformation (DWT). Only the approximate wavelet coefficients are used to obtain a low-pass representation of the signal. The Daubechies wavelet with 30 coefficients (db30) was found to produce the most robust results. The wavelet transformation was done using the PyWavelets* python package. The final feature vector is composed of the wavelets' approximate coefficients and the arithmetic mean of the power measurements. The wavelet transformation and arithmetic mean were explained in Section 6.3.1.

*https://github.com/PyWavelets/pywt

7.6.2 Classification

Due to the need to create a methodology that can rely on a few annotation samples, the choice was made to incorporate SVM to classify sliding windows of the time series. SVMs use a hyperplane to separate the classes linearly in a dataset. The hyperplane not only separates two classes, it also maximizes the distance between the hyperplane and each class, making SVMs a large margin classifier [Bose 92]. Therefore the hyperplane is defined by the closest data points from each class which in turn become the support vectors. The benefit of using an SVM over other classifiers such as Neural Networks is that it is less prone to a sampling selection bias because it does class separation through a hyperplane instead of class-conditional probabilities. As the problem is, in all likelihood, not linearly separable, we use a non-linear polynomial kernel function to transform the data into a higher-dimensional space. SVMs supports the classification of more than two classes or binary problems called multi-class SVM. It is important to note that SVMs are inherently capable of separating only two classes since a single hyperplane divides the feature space into two parts. For this reason, they have to be implemented using workarounds such as multiple one-vs-rest classifiers. Multi-class SVMs are not used here. Instead, two completely independent SVMs were trained in order to classify start and stop events. The start and stop events are then connected using a heuristic.

The SVM provided by the sklearn[*] python packages were used for the reference implementation. Different hyperparameters have been tested as part of the publication [Wenn 19b]. Using a polynomial kernel proved to give the best overall result. A degree-d polynomial kernel K of the input vectors $\boldsymbol{x}, \boldsymbol{y}$ is defined as:

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^T \boldsymbol{y} + c)^d \quad , \tag{7.7}$$

where the constant c (often referred to as coef0 in implementations) is added to the dot product. This constant allows the kernel to adapt better to data that is not centered or normalized. It effectively controls the influence of higher-dimensional features in the transformed space. The experiments showed that using $d = 5$ and $c = 10$ is a good overall choice. Next to these kernel parameters, the SVM result is also influenced by a regularization parameter. The parameter influences the trade-off choice made when searching for the hyperplane that separates the two classes. The trade-off is between a hyperplane with the largest minimum margin and a hyperplane that correctly separates the samples as well as possible. In sklearn this parameter is called C and has a default value of 1.0. A small C value determines a favor for a large minimum margin while accepting higher misclassifications, or vice versa. Thus, for noisy or hard-to-separate problems, it can be necessary to choose a C value lower than 1.0. Choosing the C parameter is part of the here presented evaluation.

7.6.3 Finding Events

Since the trained classifier only provides independent classification results, a heuristic is required for a start and a stop to create events, defined as $e = (t_0, t_1)$. Starts and stops are connected by iterating over the found starts and connecting each start t_0

[*]https://scikit-learn.org

with the following closest stop t_1 so that $t_0 < t_1$. In cases where multiple starts are found before the next stop is found, only the first occurrence is used, favoring larger events over shorter ones.

7.7 Event Post Processing

7.7.1 Event Filtering

An event duration filter is applied after constructing the events, meaning that only events of similar duration are accepted. The *accepted* duration interval is defined as:

$$\text{accepted} = [0.8 \times \min(b), 1.2 \times \max(b)] \quad , \tag{7.8}$$

where b is the duration of the events in the training set. The 0.8 and 1.2 have been determined empirically. This post-processing step presents a very important improvement compared to the method used for the DEDDIAG baseline [Wenn 21b]. The difference will be shown in the following evaluation.

7.7.2 Extracting Exact Timestamp

Using a sliding window of e. g., 512 seconds, with a single label per window, will lead to an approximated event where we only know that the event lies within the window. The larger the window, the less exact the event classification is. Since we are classifying successive sliding windows, we can calculate the number of windows within which the event is present using:

$$\text{eventWindowCount} = 2 \times \frac{b}{\text{stepSize}} \quad .$$

In a case where all windows are classified correctly, the event is the last value of the first window and the first value of the last window. This allows for improvements of the classification by first calculating a probability on the found event by comparing the number of positive labeled successive windows to the expected window count. Second, we can estimate the exact timestamp by averaging the last value of the first window and the first value of the last window. In case all windows were classified correctly, we are able to calculate the exact second the event occurred, thus overcoming the problem of increasing the window size.

7.8 Evaluation

7.8.1 Data

The evaluation is performed on the DEDDIAG dataset using a 5-fold cross-validation based on the annotations available. Not all data in the dataset are annotated, but it is assumed that all annotations are in series. This means, that with respect to time, the first and last annotation marks a fully annotated area, and all data within can be used for an evaluation. As it cannot be assumed that the appliance usage is evenly distributed, the 5-fold was done based on the annotations and not on the measurements. This means that each fold will not have the same number of measurements, but the same number of annotations, and therefore usages. It is likely for some appliances, such as a washing machine (e. g., item 6), to not have annotations for a full month. Thus, if the split would be done based on the measurements, only very few or even no usages are part of the split. Considering this, the data is created as follows:

1. Split the annotations into five even and contiguous parts,
2. Find the test measurement range by using the first start and last stop from test annotations,
3. Find the training measurement range by using the first start and last stop from the test annotations, excluding the time span used for testing.

These annotations are converted into a sliding window as previously described. By choosing to cut the folds based on the annotations, the test set will always directly start with the appliance being used. In order to guarantee that the transient state is present, the measurements are taken one window size before the annotation. Thus the test set will always start with one OFF window followed by an ON window. The same applies to the end of the test split.

The thresholding algorithm is tested on window sizes (ws) in the interval $[1, 24]$. The SVM algorithm is tested using $\mathrm{ws} = 2^n$ with $n \in [3, 9]$. A constant step size of 5 was used for all cases where the step size was smaller than the window size, otherwise, a step size of 1 was used. Based on the annotations, the measurement windows are labeled as ON and OFF. For refrigerators, the different labels available to the appliance, that is compressor and light, have been evaluated as independent tasks. For dishwashers and washing machines, the different annotations are only the available modes and are therefore evaluated together. In total 15 different segmentation tasks have been evaluated using 12 different appliances: 3 refrigerators, 5 washing machines, and 4 dishwashers.

The classifiers are evaluated based on two different metrics. First using the F_1-Score, which evaluates each annotated window, providing a metric for the evaluation per window. Secondly, the classification is evaluated using the JTES, providing a metric for the evaluation per event. While the F_1-Score provides a good indication of how many windows have been classified correctly, the JTES provides an answer to how well the actual real-world events have been detected.

For the SVM approach, results are presented for different C parameters. When using the default value $C = 1.0$, the SVM did not converge for some of the tested appliances. Thus, results are presented for $C = 10^{-n}$, where $n \in \{1, 2, 3\}$.

7.8.2 Results

Table 7.3 lists results for the 15 segmentation tasks using the thresholding approach. Only the result for the best window size is presented, chosen based on the highest JTES. Table 7.4 and 7.5 list results for the SVM approach. Figure 7.4 shows the plotted F_1-Score and Fig. 7.5 the JTES of all window sizes tested for the thresholding algorithm. Figure 7.7 shows the plotted F_1-Score and Fig. 7.6 the JTES of all window sizes and C parameters tested for the SVM algorithm. Both the tables and boxplots show filtered and unfiltered results as described in Section 7.7.1 The lower bound thresholds calculated from the training fold are listed in Table 7.1. The original baseline results of the thresholding algorithm, that have been published for the DEDDIAG dataset [Wenn 21b], can be found in Appendix C. Overall it can be said that all thresholding results have been improved compared to the published results due to the algorithmic improvements described.

Refrigerators

The results clearly show that the refrigerator compressor cycles can reliably be annotated using the thresholding algorithm having a near perfect JTES of 0.9908, 0.9888, and 0.9917 for appliances 1, 9, and 10 without filtering events by length. There is no significant difference using the length-based event filtering explained in Section 7.7.1. Here the JTESs are 0.9853, 0.9979, 0.9970. For the compressor, the F_1-Scores for both classes are good and show that nearly all windows have been classified correctly. The refrigerator light, however, shows a large difference in results between filtered and unfiltered versions. Unfiltered results show very low JTESs of 0.2182, 0.2809, and 0.0950, while filtered are 0.4081, 0.4126, 0.6667. Comparing the chosen thresholds shown in Table 7.1, the light and compressor have a similar threshold of around 1.1742 – 6.4787 W, compared to 0.2902 – 7.8069 W. This means, that both the classifiers will not be able to separate the light and compressor based on the threshold. As shown in Table 7.2, the event length of the compressor can be significantly longer compared to the light. Thus many false positives can be detected based on the duration. The best result was achieved at window sizes of 1, 2, and 3. Based on the JTES box plot in Fig. 7.5(b), it can be seen that for appliance 9 label 31 and 10 label 30 the JTES is stable between folds and window sizes, although appliance 9 has a few low score outliers.

For the same appliances (1, 9, 10), the SVM approach achieves a JTES of 0.9574, 0.8623, and 0.9752. The weighted F_1 results are 0.9881, 0.9547, and 0.9977. Refrigerator compressor results are overall very stable between the different parameters. The unfiltered and filtered results are not significantly different. Also changing the C hyperparameter does not change the result significantly. The chosen window sizes are between 16 and 128. In terms of steady-state profile complexity, the refrigerator's compressor is the most simple task and it is expected to be well detected by the thresholding algorithm (see Fig. 7.1). At the start of each cooling cycle, the compressor starts and the power demand jumps to a certain value. It then slowly decreases until the compressor stops. Both algorithms achieve very good results, with the thresholding being a little bit ahead of the SVM. The refrigerator light has a maximum unfiltered JTES of 0.4115, 0.2710, and 0.2586, which, overall, is better compared to the thresholding. The filtered results are 0.3999, 0.2734, and 0.2478, meaning a small improvement for appliance 9, but not for the rest. The C hyperparameter has also no significant impact on the resulting JTES. All three lights have the best result at the lowest tested window size 16. Since the light events are on average only 30, 13, and 6.5 seconds long, a window size of 16 is very large. Although the averages are low, the event filter lengths used are very different between the three refrigerators (see Table 7.2). A further test with window size 3 and step size one showed that the JTES can be increased significantly for appliances 1 and 9 to 0.5617, 0.5202, but did decrease for appliance 10 to 0.2397.

Overall the thresholding approach outperforms the SVM on the compressor task, and on the appliance 9 the difference is significant. For the light task, both algorithms have a medium performance. The lower bound thresholding is not capable of segmenting the light without filtering the events based on duration. A further weakness of the thresholding's simplicity: the light might switch on when the compressor is OFF or ON, which would require much more complex thresholding rules. A single threshold is simply not capable of differentiating between compressor and light. Using a lower and upper bound combined with the change in power usage might help to overcome this problem.

Washing Machines

Washing machines, with appliance id 2, 6, 12, 20, and 24 show medium-quality results based on the JTES for the thresholding algorithm. Results are in the range from 0.3675 to 0.7829 for unfiltered and 0.3831 to 0.9382 with filtering. Thus, the algorithm significantly benefits from duration-based filtering. There is a significant change in the result compared to the results published with the DEDDIAG dataset, where the range was 0.0385 to 0.6055. This vast performance increase, especially on the washing machine with id 6, was achieved by removing the data smoothing pre-processing step. In the first version of the algorithm used in the DEDDIAG publication, the data was pre-processed with a windowed average of size 3. The appliance has multiple sections where the power demand is very low and after smoothing the signal the power frequently fell below the chosen average. The chosen threshold by the algorithm range from 0.2861 to 1.9314 W (cf. Table 7.1). Unlike the refrigerator where a clear preference for small window sizes is indicated, there is no clear window sizes recommendation for the washing machine category. The best results have been found at windows ranging from 1–19. The algorithm works by far best on appliance 2 when using the event filter, resulting in a near-perfect JTES of 0.9382. Boxplots in Fig. 7.5 show that adding the event duration filtering reduces the variance of the thresholding results significantly, indicated by the much smaller whiskers. This trend also applies to the SVM results shown in Fig. 7.6.

The SVM results on the washing machine are very diverse. The C hyperparameter has a significant impact on the results. While for appliances 2 and 12 a higher C value is the better choice, for appliances 6, 20, and 24, the lowest tested $C = 0.001$ is the best choice. This indicates that appliances 6, 20, and 24 are the most difficult to separate. Training the SVM for appliance 20 does not converge when using $C = 0.1$, thus no results are provided. Unfiltered results range from 0.0 to 0.5460 and filtered from 0.1131 to 0.5915, where all results benefit from the event filtering. On Appliance 24, the difference between unfiltered and filtered is 0.0 to 0.5915, meaning without the filter, the number of false positive events is so high that the JTES is pushed to near zero by the number of false events, making the true events insignificant. This is not reflected in the weighted F_1-score of 0.9791. Thus for washing machines, the event filtering presents a vital step for the SVM approach. Overall the thresholding algorithm outperforms the SVM approach for washing machines, except for appliance 12 where both achieve comparable results. Event filtering improves both algorithms

Table 7.1: Lower bound thresholds calculated from the training split for each fold. While refrigerators use a high threshold, washing machines, and dishwashers require thresholds close to zero. For the refrigerator task, the two labels compressor (C) and light (L) have been evaluated separately.

Item	Category	Labels	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Variance
1	Refrigerator	33 (C)	6.1362	6.1362	6.1362	6.1362	14.4896	7.8069	11.1645
1	Refrigerator	34 (L)	2.3767	2.3767	3.7584	2.3767	2.3767	2.6531	0.3054
9	Refrigerator	31 (C)	0.3011	0.3149	0.3011	0.3011	0.3011	0.3039	0.0000
9	Refrigerator	32 (L)	1.0881	1.0881	1.0881	1.5188	1.0881	1.1742	0.0297
10	Refrigerator	29 (L)	6.3812	6.3812	6.3812	6.3812	6.8686	6.4787	0.0380
10	Refrigerator	30 (C)	0.2829	0.3194	0.2829	0.2829	0.2829	0.2902	0.0002
2	Washing Machine	35, 36	2.4481	2.3672	2.3672	2.3672	2.3672	2.3834	0.0010
6	Washing Machine	5, 21, 26	0.3108	0.2887	0.2887	0.2887	0.2887	0.2931	0.0001
12	Washing Machine	37	2.7089	0.7434	2.0546	4.0761	0.0740	1.9314	2.0161
20	Washing Machine	6, 7, 8, 11, 40, 41	0.3537	0.3537	0.3537	0.4159	0.3537	0.3662	0.0006
24	Washing Machine	14	0.2893	0.2853	0.2853	0.2853	0.2853	0.2861	0.0000
4	Dishwasher	17, 18	0.3212	0.3250	0.3250	0.3786	0.3188	0.3337	0.000
5	Dishwasher	9, 10, 15	0.2733	0.2733	0.2733	0.2733	0.2776	0.2741	0.0000
19	Dishwasher	2, 4, 19, 24	0.1537	0.1537	0.1537	0.1537	0.1689	0.1567	0.0000
26	Dishwasher	12, 22, 23	0.0675	0.0675	0.0675	0.0675	0.0675	0.0675	0.0000

significantly. In terms of window size, washing machines have a clear preference for larger window sizes of 128 and 256.

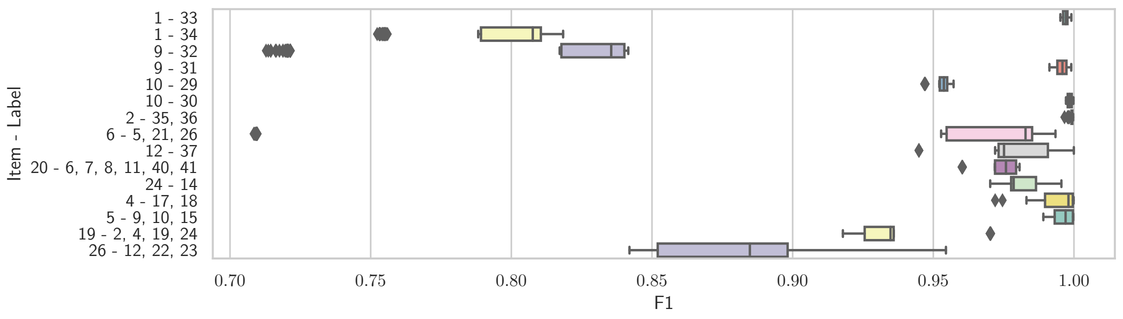
Dishwashers

The unfiltered thresholding dishwasher (4, 5 19, 26) JTES results are 0.8107, 0.7723, 0.5380, and 0.0788 (cf. 7.3). While dishwashers 4 and 5 show good results, 19 has only a medium performance and on dishwasher 26 the algorithm fails completely with a very low JTES. The filtered results are 0.8778, 0.9462, 0.3518, and 0.8341. Results for four out of five dishwashers improve significantly using the event duration filter. On appliance 26, the unfiltered result changes from 0.0788 to 0.8341, meaning that without a filter the thresholding algorithm does not segment the appliances in a useful manner, but it does. On the other hand, for appliance 19 the performance decreases when using the event filter. Compared to challenge results in Appendix C, both filtered and unfiltered results improved. The algorithm had to choose a very low threshold as there are many periods when the washing machine uses very little electricity. It then splits the event into multiple parts, which is heavily punished by the JTES, but not by the weighted F_1-score. Thresholds range from 0.0143 to 0.3541 W. The unfavorable JTES result on dishwasher 26 can be explained by a very high number of false positives of short length. These short-length false positives are successfully removed by duration filtering.

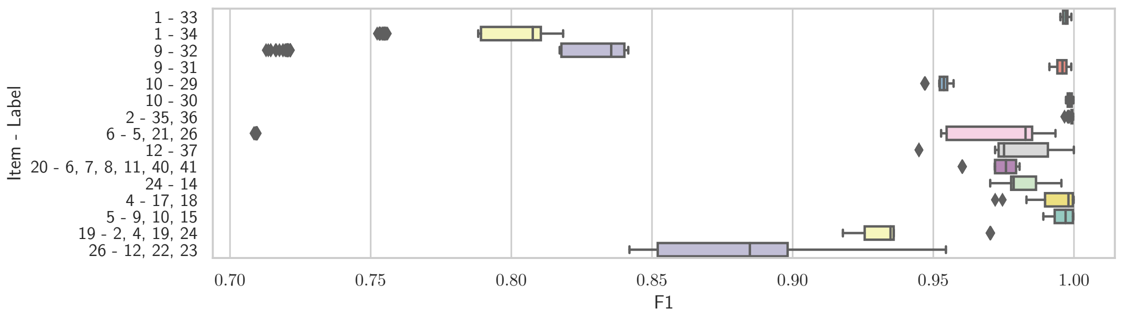
For the SVM approach, the best-unfiltered dishwasher (4, 5 19, 26) JTES results are 0.2371, 0.5839, 0.1288, and 0.5793. The best filtered JTES results are 0.2746, 0.8828, 0.4275, and 0.8030. Meaning, that similar to the washing machine results, the event filtering presents a significant difference. Dishwasher 4 performs best using $C = 0.001$, while appliances 5, 19, and 26 perform best using $C = 0.01$. Compared to the thresholding, the SVM results are in all cases not as good as the thresholding, but for appliances 5 and 26 comparably good. No clear preference for window sizes exists.

Table 7.2: Event length range used to filter both, thresholding and SVM results in seconds. Values are rounded to the nearest second. For the refrigerator task, the two labels compressor (C) and light (L) have been evaluated separately.

Item	Category	Labels	Fold 1		Fold 2		Fold 3		Fold 4		Fold 5	
1	Refrigerator	33 (C)	952	4140	952	4140	952	4140	968	4140	952	2172
1	Refrigerator	34 (L)	8	420	8	420	8	420	8	420	8	120
9	Refrigerator	31 (C)	614	1487	612	1487	612	1483	612	1487	612	1487
9	Refrigerator	32 (L)	2	124	2	124	2	124	2	59	2	124
10	Refrigerator	29 (L)	3	29	2	16	2	29	2	29	2	29
10	Refrigerator	30 (C)	413	937	418	955	413	955	413	955	413	955
2	Washing Machine	35, 36	1963	13933	1505	13933	1505	13933	1505	13933	1505	13660
6	Washing Machine	5, 21, 26	345	12485	345	12223	1680	12485	345	12485	345	12485
12	Washing Machine	37	2424	10936	2424	11280	2514	11280	2424	11280	2424	11280
20	Washing Machine	6, 7, 8, 11, 40, 41	541	24815	541	24815	541	24815	1437	16100	541	24815
24	Washing Machine	14	1387	24566	1387	24566	2388	23327	1387	24566	1387	24566
4	Dishwasher	17, 18	472	11759	472	11759	472	11759	472	11527	472	11759
5	Dishwasher	9, 10, 15	879	17966	879	15260	902	17966	879	17966	879	17966
19	Dishwasher	2, 4, 19, 24	1426	10883	890	10925	890	10925	890	10925	890	10925
26	Dishwasher	12, 22, 23	1660	8794	3450	7618	1660	8794	1660	8794	1660	8794



(a) Unfiltered thresholding



(b) Filtered thresholding

Figure 7.4: Boxplot showing F_1-Scores of thresholding segmentation method. Filtering does not have any effect on the thresholding score.

Table 7.3: Evaluation result of segmentation task using a thresholding method. While the task for dishwashers and washing machines does not distinguish between different labels per item, for refrigerators the two labels compressor (C) and light (L) are evaluated as two separate tasks. Results are evaluated using JTES and F_1-score per class as well as the weighted F_1-score. The result is shown for window sizes where JTES was highest in a trial with sizes ranging from 1 to 24.

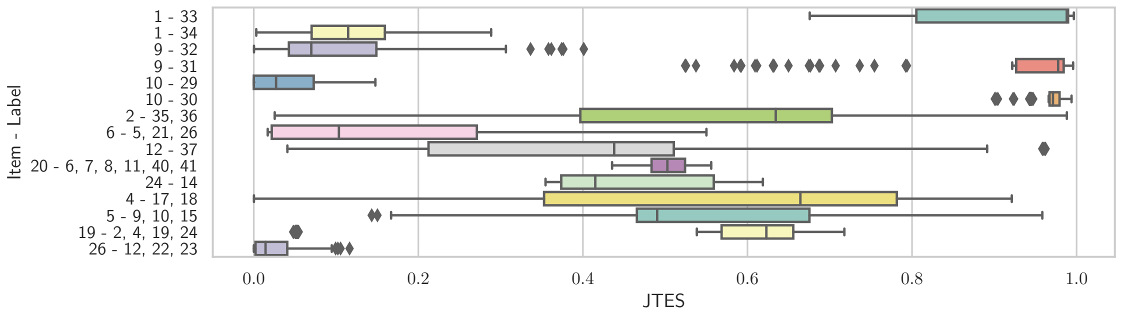
Item	**Category**	**Labels**	**ws**	**JTES**	**F_1 OFF**	**F_1 ON**	**Weigh. F_1**
Unfiltered							
1	Refrigerator	33 (C)	9	0.9908	0.9983	0.9967	0.9978
1	Refrigerator	34 (L)	2	0.2182	0.8022	0.0445	0.7963
9	Refrigerator	31 (C)	7	0.9888	0.9986	0.9968	0.9981
9	Refrigerator	32 (L)	3	0.2809	0.8166	0.0530	0.8098
10	Refrigerator	29 (L)	2	0.0950	0.9533	0.0030	0.9531
10	Refrigerator	30 (C)	5	0.9917	0.9996	0.9966	0.9994
2	Washing Machine	35, 36	24	0.7829	0.9996	0.9947	0.9993
6	Washing Machine	5, 21, 26	17	0.3675	0.9328	0.5636	0.9252
12	Washing Machine	37	12	0.5125	0.9836	0.6204	0.9768
20	Washing Machine	6, 7, 8, 11, 40, 41	1	0.5184	0.9842	0.7708	0.9737
24	Washing Machine	14	9	0.4863	0.9894	0.8482	0.9817
4	Dishwasher	17, 18	22	0.8107	0.9977	0.9458	0.9958
5	Dishwasher	9, 10, 15	23	0.7723	0.9976	0.9525	0.9957
19	Dishwasher	2, 4, 19, 24	23	0.5380	0.9543	0.6154	0.9368
26	Dishwasher	12, 22, 23	22	0.0788	0.8955	0.3059	0.8722
Filtered							
1	Refrigerator	33 (C)	1	0.9735	0.9987	0.9975	0.9983
1	Refrigerator	34 (L)	2	0.4081	0.8022	0.0445	0.7963
9	Refrigerator	31 (C)	1	0.9979	0.9980	0.9955	0.9972
9	Refrigerator	32 (L)	3	0.4126	0.8166	0.0530	0.8098
10	Refrigerator	29 (L)	2	0.6667	0.9533	0.0030	0.9531
10	Refrigerator	30 (C)	1	0.9970	0.9999	0.9988	0.9998
2	Washing Machine	35, 36	19	0.9382	0.9996	0.9946	0.9992
6	Washing Machine	5, 21, 26	1	0.5094	0.9332	0.5685	0.9256
12	Washing Machine	37	7	0.3831	0.9836	0.6208	0.9768
20	Washing Machine	6, 7, 8, 11, 40, 41	1	0.5283	0.9842	0.7708	0.9737
24	Washing Machine	14	1	0.6998	0.9896	0.8505	0.9821
4	Dishwasher	17, 18	12	0.8778	0.9977	0.9465	0.9959
5	Dishwasher	9, 10, 15	1	0.9462	0.9977	0.9546	0.9958
19	Dishwasher	2, 4, 19, 24	4	0.3518	0.9545	0.6166	0.9370
26	Dishwasher	12, 22, 23	5	0.8341	0.9121	0.3407	0.8894

Table 7.4: Evaluation result of segmentation task using a SVM method without event length filtering. Results are evaluated using JTES and F_1-score. A result is shown for window sizes where JTES was highest in a trial with window sizes ws $= 2^n$ with $n \in [3, 9]$. No result could be calculated for appliance 24, as the SVM did not converge.

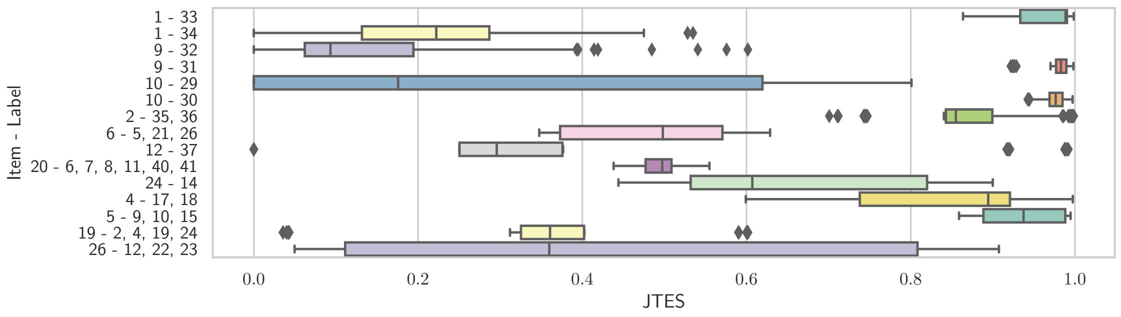
Item	**Category**	**Labels**	**ws**	**JTES**	**F_1 OFF**	**F_1 ON**	**Weigh. F_1**
Unfiltered							
$C = 0.1$							
1	Refrigerator	33 (C)	32	0.9574	0.9911	0.9816	0.9881
1	Refrigerator	34 (L)	16	0.4004	0.9200	0.2748	0.9184
9	Refrigerator	31 (C)	32	0.8469	0.9740	0.9218	0.9595
9	Refrigerator	32 (L)	16	0.2598	0.9197	0.1577	0.9166
10	Refrigerator	29 (L)	16	0.2541	0.9999	0.0000	0.9999
10	Refrigerator	30 (C)	16	0.9751	0.9987	0.9880	0.9977
2	Washing Machine	35, 36	512	0.5460	0.9880	0.8083	0.9761
6	Washing Machine	5, 21, 26	256	0.0440	0.1440	0.0194	0.1425
12	Washing Machine	37	256	0.4084	0.9922	0.7184	0.9877
20	Washing Machine	6, 7, 8, 11, 40, 41	–	–	–	–	–
24	Washing Machine	14	16	0.0343	0.9870	0.7915	0.9766
4	Dishwasher	17, 18	256	0.0839	0.9839	0.4386	0.9613
5	Dishwasher	9, 10, 15	32	0.3475	0.9981	0.9629	0.9965
19	Dishwasher	2, 4, 19, 24	16	0.0000	0.9813	0.6310	0.9641
26	Dishwasher	12, 22, 23	128	0.5793	0.9966	0.9126	0.9931
$C = 0.01$							
1	Refrigerator	33 (C)	32	0.9574	0.9911	0.9816	0.9881
1	Refrigerator	34 (L)	16	0.4044	0.9289	0.2753	0.9273
9	Refrigerator	31 (C)	128	0.8621	0.9679	0.9156	0.9543
9	Refrigerator	32 (L)	16	0.2710	0.9578	0.2135	0.9555
10	Refrigerator	29 (L)	16	0.2586	0.9999	0.0889	0.9999
10	Refrigerator	30 (C)	16	0.9751	0.9987	0.9880	0.9977
2	Washing Machine	35, 36	512	0.4334	0.9826	0.7143	0.9637
6	Washing Machine	5, 21, 26	256	0.0513	0.3214	0.0942	0.3184
12	Washing Machine	37	256	0.2747	0.9918	0.6774	0.9868
20	Washing Machine	6, 7, 8, 11, 40, 41	16	0.0415	0.9860	0.6140	0.9674
24	Washing Machine	14	16	0.0000	0.9880	0.8219	0.9791
4	Dishwasher	17, 18	128	0.0853	0.9885	0.6582	0.9750
5	Dishwasher	9, 10, 15	32	0.5839	0.9985	0.9668	0.9970
19	Dishwasher	2, 4, 19, 24	64	0.1288	0.9819	0.7066	0.9702
26	Dishwasher	12, 22, 23	64	0.3326	0.9951	0.8735	0.9903
$C = 0.001$							
1	Refrigerator	33 (C)	32	0.9574	0.9911	0.9816	0.9881
1	Refrigerator	34 (L)	16	0.4115	0.9516	0.3985	0.9505
9	Refrigerator	31 (C)	128	0.8623	0.9682	0.9158	0.9547
9	Refrigerator	32 (L)	16	0.2621	0.9573	0.0575	0.9537
10	Refrigerator	29 (L)	16	0.2586	0.9999	0.0889	0.9999
10	Refrigerator	30 (C)	16	0.9752	0.9988	0.9880	0.9978
2	Washing Machine	35, 36	256	0.1405	0.3941	0.2192	0.3806
6	Washing Machine	5, 21, 26	256	0.0511	0.3990	0.2457	0.3963
12	Washing Machine	37	128	0.0633	0.0165	0.0202	0.0165
20	Washing Machine	6, 7, 8, 11, 40, 41	16	0.0000	0.9860	0.6146	0.9676
24	Washing Machine	14	16	0.0000	0.9944	0.8947	0.9888
4	Dishwasher	17, 18	256	0.2371	0.9884	0.6338	0.9745
5	Dishwasher	9, 10, 15	16	0.0640	0.9934	0.8200	0.9862
19	Dishwasher	2, 4, 19, 24	64	0.0126	0.7846	0.5616	0.7773
26	Dishwasher	12, 22, 23	32	0.1403	0.9937	0.8201	0.9868

Table 7.5: Evaluation result of segmentation task using a SVM method with event length filtering. Results are evaluated using JTES and F_1-score. A result is shown for window sizes where JTES was highest in a trial with window sizes ws $= 2^n$ with $n \in [3, 9]$. No result could be calculated for appliance 24, as the SVM did not converge.

Item	**Category**	**Labels**	**ws**	**JTES**	**F_1 OFF**	**F_1 ON**	**Weigh. F_1**
Filtered							
$C = 0.1$							
1	Refrigerator	33 (C)	32	0.9383	0.9911	0.9816	0.9881
1	Refrigerator	34 (L)	16	0.3839	0.9200	0.2748	0.9184
9	Refrigerator	31 (C)	32	0.8566	0.9740	0.9218	0.9595
9	Refrigerator	32 (L)	16	0.2532	0.9197	0.1577	0.9166
10	Refrigerator	29 (L)	16	0.2478	0.9999	0.0000	0.9999
10	Refrigerator	30 (C)	16	0.9751	0.9987	0.9880	0.9977
2	Washing Machine	35, 36	512	0.5455	0.9880	0.8083	0.9761
6	Washing Machine	5, 21, 26	128	0.1131	0.9140	0.3738	0.9020
12	Washing Machine	37	256	0.4812	0.9922	0.7184	0.9877
20	Washing Machine	6, 7, 8, 11, 40, 41	–	–	–	–	–
24	Washing Machine	14	16	0.5070	0.9870	0.7915	0.9766
4	Dishwasher	17, 18	64	0.1879	0.9873	0.6073	0.9717
5	Dishwasher	9, 10, 15	64	0.8558	0.9967	0.9221	0.9938
19	Dishwasher	2, 4, 19, 24	16	0.4275	0.9813	0.6310	0.9641
26	Dishwasher	12, 22, 23	128	0.7529	0.9966	0.9126	0.9931
$C = 0.01$							
1	Refrigerator	33 (C)	32	0.9383	0.9911	0.9816	0.9881
1	Refrigerator	34 (L)	16	0.3880	0.9289	0.2753	0.9273
9	Refrigerator	31 (C)	128	0.8597	0.9679	0.9156	0.9543
9	Refrigerator	32 (L)	16	0.2688	0.9578	0.2135	0.9555
10	Refrigerator	29 (L)	16	0.2211	0.9999	0.0889	0.9999
10	Refrigerator	30 (C)	16	0.9751	0.9987	0.9880	0.9977
2	Washing Machine	35, 36	128	0.4701	0.9893	0.8353	0.9785
6	Washing Machine	5, 21, 26	128	0.1809	0.9932	0.5697	0.9840
12	Washing Machine	37	256	0.4229	0.9918	0.6774	0.9868
20	Washing Machine	6, 7, 8, 11, 40, 41	128	0.2861	0.9868	0.6437	0.9704
24	Washing Machine	14	16	0.5560	0.9880	0.8219	0.9791
4	Dishwasher	17, 18	128	0.1814	0.9885	0.6582	0.9750
5	Dishwasher	9, 10, 15	32	0.8828	0.9985	0.9668	0.9970
19	Dishwasher	2, 4, 19, 24	64	0.3574	0.9819	0.7066	0.9702
26	Dishwasher	12, 22, 23	128	0.8030	0.9963	0.9057	0.9929
$C = 0.001$							
1	Refrigerator	33 (C)	32	0.9383	0.9911	0.9816	0.9881
1	Refrigerator	34 (L)	16	0.3999	0.9516	0.3985	0.9505
9	Refrigerator	31 (C)	128	0.8599	0.9682	0.9158	0.9547
9	Refrigerator	32 (L)	16	0.2734	0.9573	0.0575	0.9537
10	Refrigerator	29 (L)	16	0.2211	0.9999	0.0889	0.9999
10	Refrigerator	30 (C)	16	0.9752	0.9988	0.9880	0.9978
2	Washing Machine	35, 36	16	0.3093	0.7890	0.6090	0.7777
6	Washing Machine	5, 21, 26	256	0.2204	0.3990	0.2457	0.3963
12	Washing Machine	37	16	0.2437	0.9966	0.7851	0.9925
20	Washing Machine	6, 7, 8, 11, 40, 41	128	0.3440	0.9874	0.6624	0.9717
24	Washing Machine	14	16	0.5915	0.9944	0.8947	0.9888
4	Dishwasher	17, 18	256	0.2746	0.9884	0.6338	0.9745
5	Dishwasher	9, 10, 15	16	0.4854	0.9934	0.8200	0.9862
19	Dishwasher	2, 4, 19, 24	32	0.3056	0.9793	0.6765	0.9646
26	Dishwasher	12, 22, 23	32	0.2403	0.9937	0.8201	0.9868



(a) Unfiltered thresholding



(b) Filtered thresholding

Figure 7.5: Boxplot showing JTES of thresholding segmentation method.

7.9 Summary

In this chapter two different appliance segmentation algorithms have been evaluated. The first is based on a lower bound threshold based on the assumption that an appliance will also stay above a certain threshold during usage. The second is based on classifying the power usage pattern at the start and end. The thresholding approach relies on the fact that only a single appliance is recorded (ILM 3). It cannot be easily extended or trained to work in aggregated data such as NILM scenarios. The thresholding results have been improved significantly compared to the previously published results on the DEDDIAG dataset. While the SVM approach is generally capable of finding the start and stop of an appliance, its performance based on the JTES is, despite the complexity of the algorithm, only good for some appliances. The JTES has been developed after the development of the SVM as a result of the poor relationship between the F_1-Score and the real-world application, that is, the actual events. While the SVM results looked promising using the F_1-Score, they show lower performance compared to the thresholding when evaluating with the new score. Thus the JTES is a very important step towards the development of appliance segmentation algorithms because it provides a real-world scenario score, where splitting and combining of events is punished. Since appliance segmentation is used to derive usage profiles, splitting and combining events will result in an incorrect number of usages, as well as usage time. Here the JTES shows its relevance and also the weakness of evaluating using the F_1-Score.

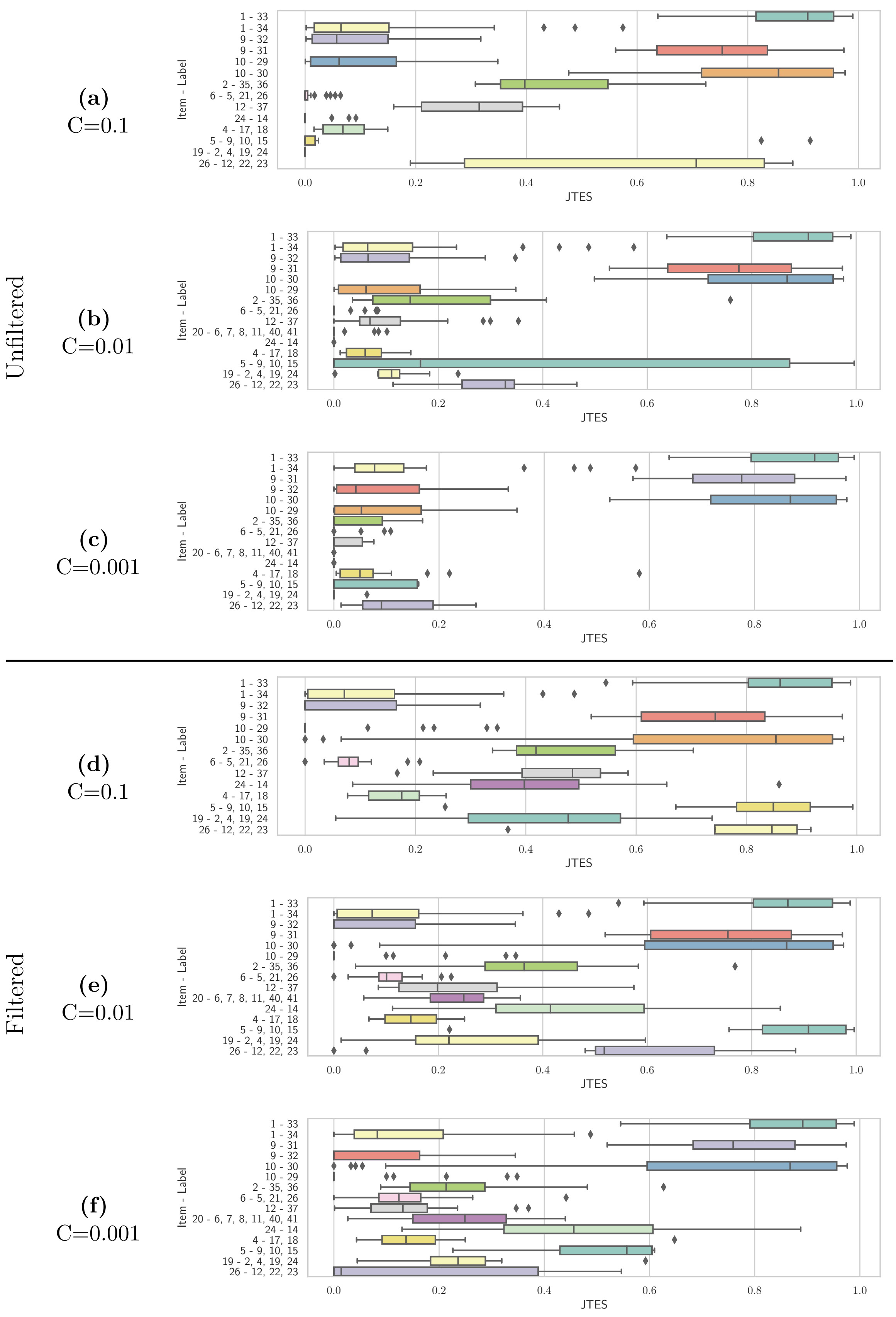


Figure 7.6: Boxplot showing JTES of SVM segmentation method. Event length filtering has a significant impact on the resulting scores.

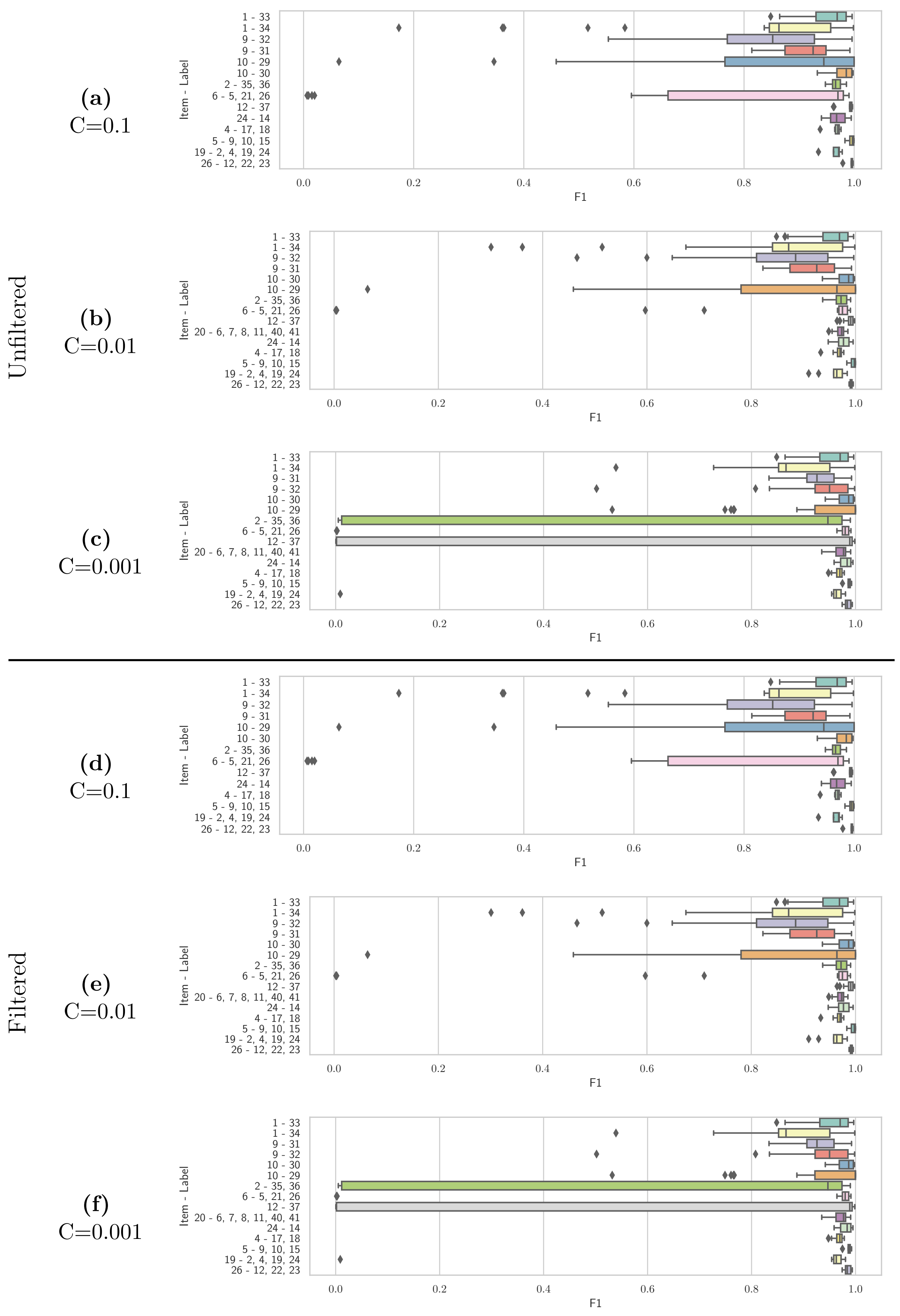


Figure 7.7: Boxplot showing F_1-Scores of SVM segmentation method.

Overall, using a simple thresholding algorithm will not provide a good and reliable event detection algorithm and the simple approach would benefit from merging small successive predictions as well as filtering based on event length. The pre-processing step used for the baseline results in the DEDDIAG publication was meant to help reduce false positives. It turns out that post-processing using an event duration filter is a much better approach as it significantly improves the overall results.

The SVM on the other hand has greater potential for more complex scenarios, but at the moment has a lower overall performance compared to the simple thresholding. Since many parameters need to be tuned for an SVM, choosing the correct parameters is difficult. For both, the thresholding and SVM approach, it will be challenging to find parameters that generalize for the category and not only a single appliance. There the SVM will have a benefit over the thresholding approach, as the SVM has the capability of finding a much more complex separation plane.

In general, appliance usage prediction and user behavior algorithms, as described in the next Chapter 8, will benefit from more advanced event annotations that are evaluated against manually annotated ground truth, as these algorithms require event annotations as their ground truth.

CHAPTER 8

Usage Prediction

Substantial parts of this chapter have been published in the following manuscript: [Wenn 17]. The majority of the manuscript has been authored by the author of this thesis. The main scientific contributions are based on his own work and thoughts.

8.1 Introduction

In this chapter, an approach to appliance usage prediction is described. The approach was designed for household appliances with regular usage such as dishwashers and washing machines. The prediction is the last step in the Machine Learning Demand Response Model (MLDR) (cf. Section 4.4). It provides the information required by automation systems to either control the appliances directly or to give recommendations to the household residents as to which time period using the appliance would result in lower energy costs. These recommendations are based on the users' normal behavior patterns, which are automatically learned from historical data. The desired information for Demand Response Management (DRM) are [Barb 11]:

- Which appliance will be used: Appliance,
- When will it run: Start time,
- How long will it take: Duration.

The answers to these questions can be derived from appliance usage prediction. Duration is part of what is commonly called load profile: the electricity load over time for a single use of an appliance. Usage prediction is sometimes seen as predicting the load and sometimes predicting precise usages. The electric load is a combination of predicting the start time of a certain profile. Thus, from a DRM point of view, there is no difference whether the appliance usage or the appliance's electricity load is predicted.

Predicting the usage is usually described as a classification task. Forecasting the load is usually described as a regression task. Short-term prediction commonly

only describes the next 1–24 h, while long-term predictions may describe up to several months ahead. The short-term prediction is the main focus in terms of the automated operation of appliances or recommendation systems, as the planning horizon of Real Time Pricings (RTPs) is in a similar range.

The here described usage prediction model is based on a household's historic appliance usage behavior. Thus requiring the extraction of appliance operation cycles (events) from its electricity metering data [Step 14]. These historic events are the result of the segmentation task as described in the previous Chapter 7.

Predicting appliance usage based on historical data follows two assumptions:

- The future usage behavior relates to past behavior,
- The unknown impact on the usage behavior is negligible.

Some appliance usage is closely related to the occupation of a household, thus predicting usage to some extent means predicting occupation. The complexity of the prediction task is therefore very dependent on the residents' daily routines. A household of a young single will have different routines compared to a retired couple or a family of 5 with children going to school. Thus, appliance usage prediction is associated with randomness and uncertainties and is therefore a nontrivial task [Basu 13].

In general, the problem can be described as finding usage $x_{m,t} \in \{0, 1\}$ of appliance m for the given time t, where 0 means OFF and 1 means ON. The time t is commonly modeled as time slots of a certain length such as 10 min or 1 h. Thus the task of prediction is to provide the ground for a decision if an appliance m will be switched on in a certain time slot t.

8.2 Related Work

Appliance usage profiling and prediction are already discussed in various publications.

Kang et al. use ON/OFF probabilities to build a non-homogeneous Markov Chain to model end-use energy profiles on appliance level [Kang 14]. Based on observations from N days, the state of an appliance m for each day n at time t is described as $x_{m,n,t}$. The authors describe a Rate of Use (ROU) statistic for each hour of the day. Although the term indicates real usage, the authors define the ROU based on the average energy consumption and not usages. ROU is defined as:

$$\mathrm{ROU}_{m,t} = \frac{1}{N} \sum_{n=1}^{N} x_{m,n,t} \quad . \tag{8.1}$$

For each appliance, a two-state Finite State Machine (FSM) models an appliance's switching probability. Monte Carlo is used to simulating ON/OFF sequences that capture the non-homogeneous stochasticity of appliance usage patterns for all appliances. The authors show that the Monte Carlo simulation converges to ROU_t. The authors claim the generalization of their model for buildings with similar characteristics.

Chang et al. propose a daily pattern-based probability model [Chan 13]. The model was created to predict the usage of printers in an office environment. The approach aims to find an optimal day-of-the-week representation and predict the printer usage within each hour of the day. They show that the probability model can be learned using a Bayesian network classifier.

Basu et al. compare the performance of different classifiers such as Bayesian networks, decision trees, and decision tables for predicting the future power consumption of an appliance [Basu 13]. The goal of the system is to predict the next hour's start/no start of an appliance. The authors describe a full home automation management system composed of an anticipative layer, a reactive layer, and a local layer that outputs user advice. The authors argue that predicting the consumption of a full house is simpler than predicting a single appliance due to the randomness. Derived from their dataset, the authors show that the difference in power usage between different weeks is higher when looking at a single washing machine compared to the full house consumption. The authors' prediction is based on historical data such as past consumption in binary form (0=OFF, 1=ON), hour of the day, day of the week, season (1-4), and month (1-12). The authors evaluate this based on accuracy, which is problematic as described in Section 8.4.1. The authors conclude their system improves with increasing historical data.

Chrysopoulos et al. and Holub et al. both describe a histogram-based approach that models usages in each time slot of a day [Chry 14, Holu 13]. The usages are modeled using a Gaussian weighting around the time of interest. An independent evaluation of the algorithm concluded that the approach results in Area under the Curve (AUC) values around 0.7 [Hube 18]. The difference between the appliances is large.

Barbato et al. describe a pattern search algorithm [Barb 11]. The data used for the pattern search is modeled as a vector of occurrences on each of N over the preceding days. The pattern search is performed on a daily basis (ON and OFF days) as well as on an hourly basis (ON and OFF hours). For each pattern, a probability expresses the likelihood that after a particular pattern, an ON or OFF day or hour follows. The evaluation by Huber et al. concludes that the approach has a similar AUC as the histogram approach, but with a smaller deviation between appliances [Hube 18].

Truong et al. propose a Bayesian inference based approach [Truo 13a]. The algorithm models the correlation between the use of appliances using a Markov chain, describing different day types k. The probability $P(x_{nmt})$ is described as:

$$P(x_{nmt}) \propto \sum_{k=1}^{K} P(k \mid n)\mu_{km}(t) \quad , \tag{8.2}$$

where k is the day type, m the appliance, t the time slot, and $P(k \mid n)$ the probability of day n being described by k. $\mu_{km}(t) \in [0, 1]$ describes the probability of the appliance m belonging to the day type k. Huber et al. find this to be the algorithm with the least deviations between the appliances, but of similar AUC [Hube 18].

8.3 A Combined Statistical Method

The prediction of an appliance usage $x_{m,t}$ can be modeled as the probability $P(x_{nmt})$. If the future usages are related to the past usages, this probability can be derived from historical usages $\boldsymbol{X}_{m,t}$. Based on this assumption, in the following a probabilistic model is described that will give a probability for each appliance m for each time slot t based on $\boldsymbol{X}_{m,t}$.

8.3.1 Probabilistic Model

The model takes two main factors into account when computing the probability that an appliance will be used:

- The time of day it is usually used,
- The time elapsed since it was used last.

For example, a dishwasher will typically be switched on in more or less regular intervals and only at certain times of day (e. g., normally not at 2 a. m.). This is modeled separately as Probability Distribution Functions (PDFs), wherein the following E will denote the event *elapsed time* and D the event *time of day*. To give recommendations to the user, the combined probability of these two events has to be calculated and must be above a defined threshold to initiate a recommendation:

$$\begin{aligned} P(E \cap D) &= P(E \mid D)P(D) \\ &\overset{\text{if independent}}{=} P(E)P(D) \quad . \end{aligned} \tag{8.3}$$

The statistical independence of E and D is assumed here; although strictly speaking not necessary, as the conditional probability $P(E \mid D)$ can be computed from the data if a sufficient amount of representative data are available. As this is quite often not the case (cf. Section 8.4 for details on typical data sets), the independence assumption results in more stable estimates of $P(E \cap D)$. That independence is valid can be checked on the data set by computing the product on the right and middle of (8.3), and checking that equality holds.

The probability $P(E)$ that the appliance is not used for a time period t (here measured in minutes) is modeled by an exponential distribution:

$$P(E) = P(E \leq t) = \begin{cases} 1 - e^{-\lambda t} & t \geq 0 \\ 0 & t < 0 \end{cases} \quad . \tag{8.4}$$

A maximum likelihood estimate of the parameter λ can be obtained from a sufficiently large data set by computing the mean value of the time periods between consecutive appliance-switch-on events. A good estimate of λ will be obtained when the time between usage is fairly regular, resulting in a small value for the variance of the time periods.

In contrast to using a continuous distribution for E, a discrete PDF for the event D is estimated from the sample data by computing relative frequencies of appliance

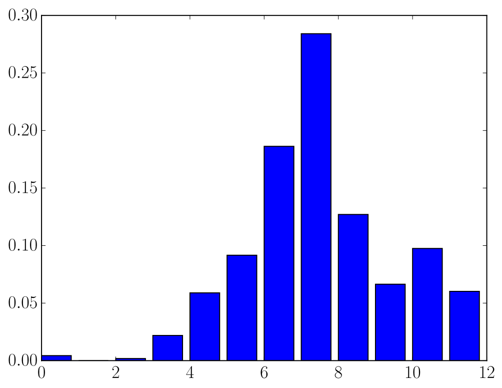


Figure 8.1: Example of estimates for $P(D)$ (discrete density of appliance usage throughout the day). Episode length is 2h, resulting in 12 episodes per day.

usage viewed over the 24-hour time period. This period has to be divided into discrete intervals, which will be called *episodes* in the following. An episode must be sufficiently large so that statistically valid relative frequencies (which are an estimate of the probability that the appliance is used in a particular episode) can be obtained. On the other hand, it has to be small enough to be of practical use. Episodes having a length of 1 h to 2 h have shown to be a good compromise. To avoid issues with episodes where no samples are contained in the data set, leading to density values of zero, the Parzen window approach [Parz 62, Duda 00] can be applied, which is basically an interpolation and smoothing method, typically using Gaussians. Figure 8.1 shows an example of a discrete $P(D)$, and Fig. 8.2 the combined cumulative distribution $P(E \cap D)$ computed from the GREEND data set [Mona 14].

8.3.2 Inactivity Detection

Most household appliances with non-homogeneous distribution in electricity consumption require the user's presence when starting the appliance. Thus, the prediction of household appliance usage is often accompanied by a prediction of the home's occupancy; in some cases, an appliance might even be directly linked to the presence of a specific person. Without ground truth on the occupancy, a strong indication that can be found in the historical electricity data is the usage of such appliances in recent history. Therefore, knowledge of the events that occurred in recent history is added to the prediction by lowering the probability in case no appliance was used in recent episodes. The previous 12 h to 24 h is of specific interest for this purpose and improves the prediction significantly. This approach could be further developed into a more sophisticated occupancy detection algorithm. For this purpose, a dataset with ground truth occupancy data is desirable.

8.3.3 Probability Threshold Estimation

Now that the usage probability can be computed for any time of day, a threshold has to be set that determines whether the appliance will be used or not. In combination

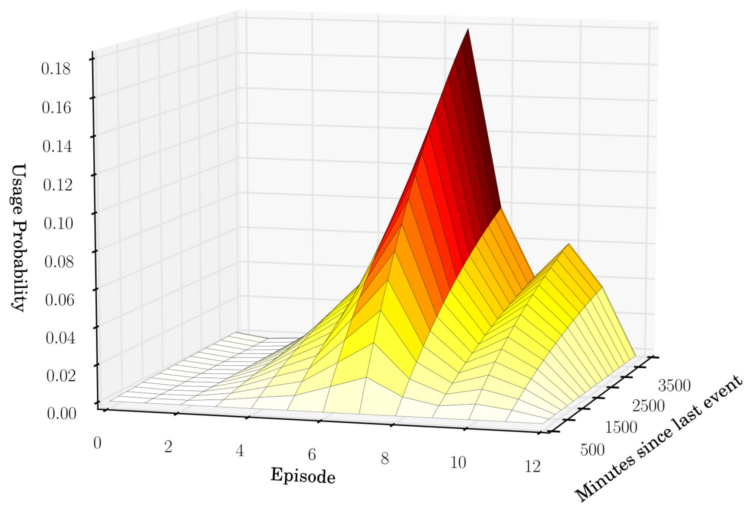


Figure 8.2: Example of the combined cumulative distribution $P(E \cap D)$. Episode length is 2h, resulting in 12 episodes per day.

with a dynamic electricity price, this may then either initiate an action on the home automation system or send a recommendation to the house residents. Note, that the absolute values of $P(E \cap D)$ depend heavily on the time interval chosen as episode duration. Obviously, longer episodes result in higher probabilities for appliance usage during this period; e. g., using 2h instead of 1h episodes would approximately double the probabilities (exactly for uniform distribution, less so for uni-/multi-modal densities). The threshold can be computed from the training data by calculating $P(E \cap D)$ for each episode of the training set. Let p_i be the predicted probability at the i-th turn-on event of a total of N that occurs in the data, the threshold θ_p is computed as the mean: $\theta_p = \frac{1}{N}\sum_{i=0}^{N-1} p_i$. If extreme outliers are expected or more control over the threshold is desired, the Median or any other quantile may be used instead.

8.3.4 Extended Model

The model described above works well when appliances are typically used in fairly regular intervals, such as dishwashers. There are scenarios, however, where the assumption fails; e. g., a household may use the washing machine every Saturday, not only for a single washing cycle but two or three times in a row. While the estimate for $P(D)$ will still be valid, the parameter λ of the exponential distribution will be invalid. This issue can be overcome by introducing an additional discrete random variable U, describing how many times an appliance has been used during the past

n_e episodes. The parameter n_e can be adjusted to the appliance at hand; e. g., for a washing machine a 10h period may suffice. Generalization of (8.3) gives:

$$\begin{aligned} P(E \cap D) &= \sum_{i=0}^{\infty} P(E \cap D \mid U = i) P(U = i) \\ &= \sum_{i=0}^{n_m} P(E \cap D \mid U = i) P(U = i) + \\ &\quad P(E \cap D \mid U > n_m) P(U > n_m) \quad , \end{aligned} \tag{8.5}$$

where n_m is an upper limit for the number of times an appliance is used that can be derived from the training data (a washing machine may be switched on 3 or 4 times in a 10h interval, but not 20 times); from a certain value of i onwards, all probabilities $P(U = i)$ will usually be zero.

8.4 Evaluation

8.4.1 Accuracy is not a Good Performance Metric

In many publications on appliance usage prediction or energy load disaggregation, the performance metric *accuracy* is chosen to evaluate the proposed classification methods. A similar problem has been addressed for the segmentation task in Chapter 7, which resulted in the development of the Jaccard-Time-Span-Event-Score (JTES) (cf. Section 7.4).

The accuracy A is defined as the proportion of data that has been classified correctly:

$$A = \frac{T_P + T_N}{n}, \tag{8.6}$$

where n is the total number of events, T_P is the number of positive events and T_N is the number of negative events that have been classified correctly (True Positives and True Negatives, respectively). For the problem at hand, we get a true positive if the algorithm predicts that an appliance is running during a given time period and the appliance is actually doing so. In the same manner, for a true negative, the prediction is that the appliance is off and this is truly the case. The denominator n is then the total number of episodes.

The main issue with this commonly chosen metric is that it is not meaningful for rare events; this is known as the *accuracy paradox* [Zhu 07, Valv 14]. Rare events, however, are in most cases the standard when looking at the problem of appliance usage prediction. The exceptions are usually appliances that do not require user interaction, such as fridges or freezers. Consider, for instance, a dishwasher: this appliance is normally used quite regularly, say every other day, and it takes about 2h to finish its cycle. This means, that in 96% of the total time the dishwasher is off. Even if it is used twice as often, i. e., every day, it will still be off 92% of the time. Publications making use of accuracy, such as [Heie 03, Barb 11, Basu 13, Lee 13, Lach 14], could therefore easily be outperformed for rare events by simply *always* predicting no occurrence (i. e., a Negative), which will result in an accuracy of 96% for the dishwasher example above.

The issue of selecting an appropriate metric has been addressed before by several authors from various fields [Cook 07, Hand 09, Powe 11a, Mako 15]. The overall performance of a binary classifier is usually captured using the Receiver Operating Characteristic (ROC), which is a plot of the true positive rate (TPR) also called sensitivity, recall, or detection rate) vs. false positive rate (FPR). These are given by:

$$\mathrm{TPR} = \frac{T_P}{P}\,, \quad \mathrm{FPR} = \frac{F_P}{N}\,, \tag{8.7}$$

where F_P is the number of negative events that have been classified incorrectly as positive ones, P is the total number of positive and N the total number of negative events, with $n = P + N$. Examples of ROC plots are shown in the evaluation Section 8.4 in Fig. 8.4. A perfect classifier would show a rectangular curve, while the main diagonal indicates complete randomness. Any point on the curve can be selected for classification by choosing the classifier's parameters appropriately. Every point results in a different value for the accuracy A calculated as shown in (8.6). In publications, where only accuracy is presented, this will usually be the point on the ROC curve where the maximum value is obtained.

This is similar for the weighted F_1-Score and the Matthews Correlation Coefficient (MCC). While the weighted F_1-Score also suffers from a bias when sample sizes for positive and negative data are different, the MCC balances these.

In contrast to all these metrics, which measure performance for a single point on the ROC, the AUC tries to capture the quality of the whole ROC in a single numerical value by computing the area under the ROC curve (cf. Section 3.5.6). For a correctly evaluated classifier, the AUC will range from 0.5 (total randomness) to 1 (perfect). Although reducing two dimensions to a single one without losing information is not feasible, AUC is still a valid metric for overall performance, and much better suited than accuracy. Extensions of AUC can be found in literature, e. g. [Hand 09], who suggests a weighted AUC.

Unfortunately, knowledge regarding evaluation metrics does not seem to be widely spread in the energy usage prediction and disaggregation community. This has been criticized before by several authors like [Kim 11, Mako 15], although with apparently little effect.

8.4.2 Evaluation on GREEND

The evaluation of the GREEND dataset presented here was used for the development of the approach described here. The results have previously been published [Wenn 17].

Data

Evaluating the proposed prediction method requires low-resolution power consumption measurements of individual appliances. A few publicly available datasets exist, usually containing the entire house and appliance-level energy consumption data. Public datasets have been addressed in Section 4.1.3 and an overview of available datasets is given in Table 4.1 Here, the GREEND dataset is used to evaluate the probabilistic prediction model. This set was chosen because it provides enough measurements to enable statistical analysis of events and is of sufficient data quality

[Mona 14]. Other datasets, such as REDD [Kolt 11] or ECO [Beck 14], do not provide enough data or the required quality (e. g., there are often long periods where data is missing). The GREEND data set provides measurements of eight homes with varying numbers of appliances and measuring periods ranging from 134 to 500 days. The data are sampled with a rate of 1 Hz and provide the power consumption in Watts per appliance.

Comparing the results to previous usage prediction publications proved to be infeasible as the results are not comparable due to the chosen data set or evaluation metric. Evaluations using artificially generated data [Heie 03, Barb 11] are not comparable as the amount of introduced randomness will dictate the result, especially for rare events such as dishwasher usage. Such evaluation should not be considered valid. Publications evaluating appliance usage prediction on short datasets, e. g. REDD [Truo 13a, Truo 13b], are also not comparable due to the insufficient amount of data in the set. REDD contains data for only up to 19 days, a duration that is considered totally inadequate for training and evaluation of rare events such as dishwasher usage.

Events

As the prediction is based on event occurrences, where an event is defined by the start and end of an appliance's usage cycle, in the first pre-processing step the events must be extracted from the continuous power consumption time series. As discussed in Chapter 7, ground truth for such segmentation task is not common among electricity datasets. The chosen GREEND dataset does not provide ground truth usage annotations, thus the events must be segmented using a segmentation algorithm. The segmentation is performed using a lower bound thresholding algorithm as described in Section 7.5 of the previous chapter. The thresholds are estimated using a window size of 60 seconds and the threshold is defined by the average power consumption within a window. In case an appliance's power consumption falls below the threshold during an event, this event will be partitioned, thus (incorrectly) generating multiple usage cycles instead of a single one. For the method proposed in this paper, the partitioning will not be an issue for computing the discrete estimate of $P(D)$ in (8.3); however, the estimate of λ of the exponential distribution $P(E)$ in (8.4) would be distorted. Since there is no ground truth to estimate a threshold for each appliance automatically, the thresholds have been estimated empirically. The thresholds are chosen so that partitioning is minimized. Further, the event length is only computed by removing outliers of unusual length. Table 8.1 gives a statistical overview of the extracted events for appliances selected from the GREEND data set.

Results

The extracted events are split into two disjoint parts, 60% for training and 40% for evaluation. The training set is used to estimate λ in (8.4) by calculating the mean value between consecutive appliance switch-on events. As an example, λ for the dishwasher in house 3 is 2274 minutes ($\approx$ 1.6 days). An episode length of 360 minutes was chosen, which divides the day into four partitions. A smaller episode length leads to a more time-precise prediction task, and vice versa. For DRM purposes, the exact

Table 8.1: Statistics of extracted events of the GREEND data set, providing the days between first and last events as well as the total events count for six different homes. Also shown are resulting performance metrics using an episode length of 360 minutes.

H#	**Appliance**	**Days**	**Events**	**AUC**	$\boldsymbol{F_1}$	**MCC**
0	coffee maker	308	676	0.703	0.708	0.498
0	dishwasher	306	143	0.684	0.389	0.348
0	fridge freezer	309	7353	0.999	0.999	0.972
0	lamp	307	215	0.701	0.524	0.344
0	television	117	445	0.567	0.852	0.251
0	washing mach.	309	256	0.591	0.378	0.190
1	bedside light	473	456	0.859	0.704	0.580
1	dishwasher	472	248	0.651	0.335	0.213
1	dryer	473	405	0.859	0.811	0.763
1	fridge	454	15	0.550	0.017	0.029
1	washing mach.	467	166	0.839	0.415	0.390
2	coffee maker	494	512	0.580	0.333	0.180
2	dishwasher	495	477	0.856	0.654	0.553
2	dryer	495	424	0.828	0.596	0.499
2	television	497	1446	0.770	0.786	0.618
2	washing mach.	497	794	0.809	0.728	0.634
3	coffee maker	456	250	0.719	0.397	0.349
3	dishwasher	456	211	0.843	0.414	0.394
3	fridge	460	1100	0.917	0.909	0.818
3	television	461	849	0.827	0.783	0.652
3	washing mach.	457	279	0.789	0.392	0.329
4	fridge freezer	282	137	0.802	0.471	0.451
4	television	280	2242	0.753	0.675	0.543
4	television 2	280	1657	0.657	0.706	0.332
4	washing mach.	263	81	0.641	0.198	0.168
5	fridge freezer	418	13054	0.734	0.985	0.665
5	lamp	416	322	0.627	0.420	0.193
5	television	417	1297	0.867	0.880	0.710
5	television 2	417	677	0.582	0.561	0.247
5	washing mach.	415	521	0.624	0.396	0.229

usage time is of minor relevance, thus four partitions per day are chosen and give a reasonable recommendation window.

The probability $P(D)$ from (8.3) for each episode is calculated by binning the appliance switch-on event duration into the corresponding episode-bin. Figure 8.3 shows $P(E \cap D)$ for the dishwasher in house 3. It clearly shows the characteristics of $P(D)$: The appliance is not likely to be used in the morning, very likely during midday, and medium in the evening. With this information, we can define the user's preferred usage window and only recommend load shifts within this window. It also shows the effect of $P(E)$ on the combined probability, as the probability drops immediately after the appliance is switched on. This respects the mean duration between events, hence no load shift recommendation must be made until a significant probability is reached in the successive episodes.

For usage prediction performance comparison between the appliances in different homes, the ROC and the AUC are calculated by changing the threshold probability at which the prediction will consider the appliances as being used; also the F_1-Score and MCC. The results are highly dependent on the house and appliance (cf. Table 8.1) The worst prediction result is obtained for the fridge in house 1, a MCC of 0.029, which is complete randomness. The reason is insufficient data: Table 8.1 shows that there were only 15 events available in the whole data set, therefore good performance cannot be expected.

Although very good results can typically be achieved for fridges, these are not ideal for load shifting; dishwashers, washing machines, and dryers on the other hand are of special interest, as they are well suited for this purpose. ROCs for these appliances are shown in Fig. 8.4. The best prediction results for this type of appliance were achieved for the dryer in house 1 with an AUC of 0.859 and MCC of 0.763. The home with the most predictable dishwasher and washing machine usage is house 2, with AUC 0.856, MCC 0.553 (dishwasher), and AUC 0.809, MCC 0.634 (washing machine). On the other hand, the results for house 0 are AUC 0.684, MCC 0.348 (dishwasher), and AUC 0.591, MCC 0.190 (washing machine). The reason for the performance difference compared to other homes lies in behavior changes of the inhabitants of house 0, which can be shown by comparing the probability distribution $P(D)$ for training and evaluation data (see Fig. 8.5). While in the dishwasher's training data, the first episode of a day has a probability of 0.37, in the evaluation data it is 0.08. The probability densities show that the events were moved to the last episode of the day, an episode with a low probability in the training data. The changes, represented as the Mean Square Error (MSE), between training and evaluation, are 0.0462 for house 0 and 0.0033 for house 2. Thus, the preconditioned behavior consistency is no longer given in house 0, a problem that could be overcome by analyzing recent behavior changes and adapting $P(D)$ accordingly.

8.4.3 Evaluation on DEDDIAG

The presented algorithm was developed and tested on the GREEND dataset. In the following, the algorithm is tested in a day-ahead scenario (24 h) using the Domestic Energy Demand Dataset of Individual Appliances in Germany (DEDDIAG) dataset.

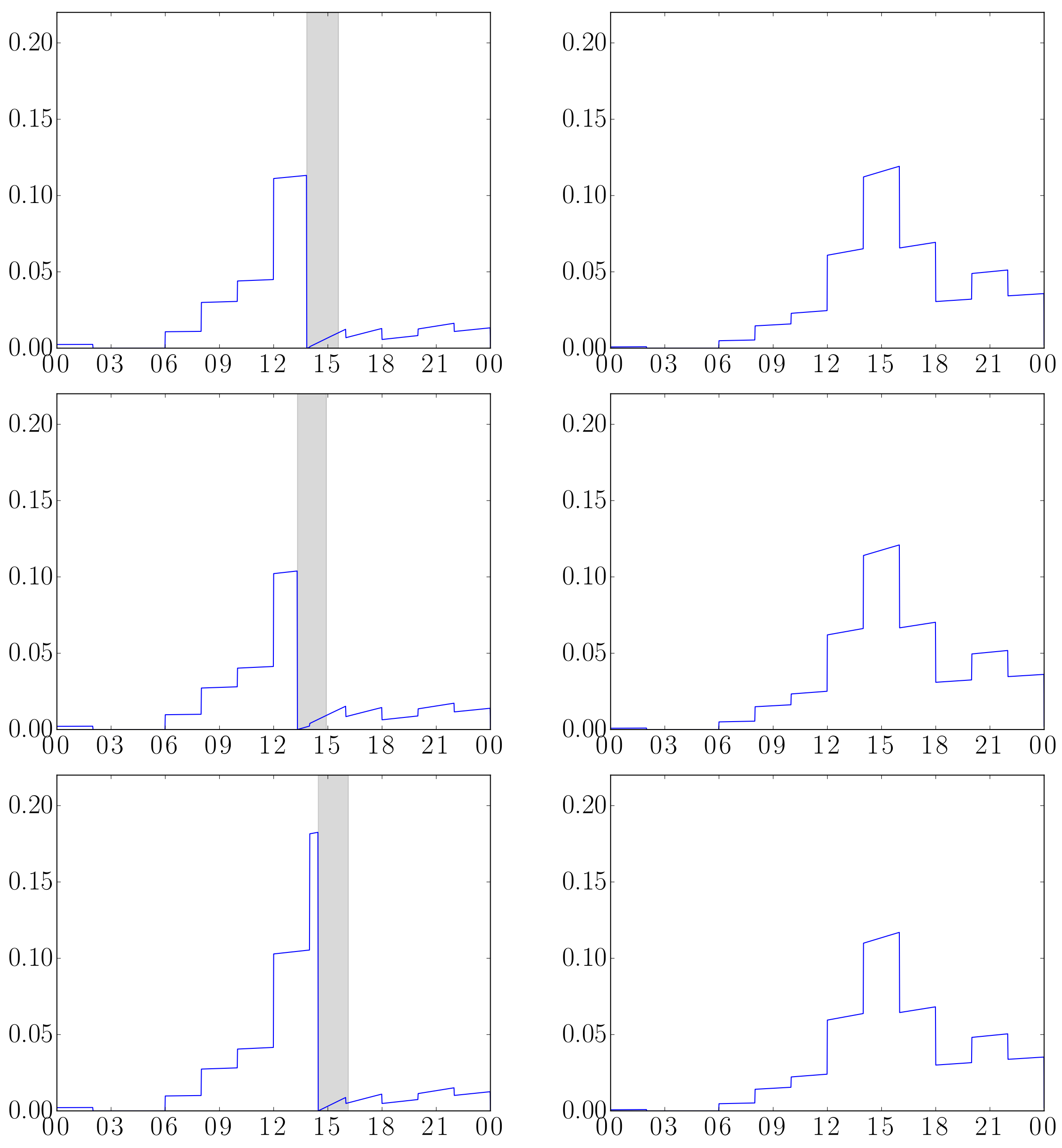
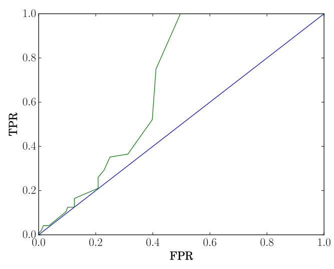
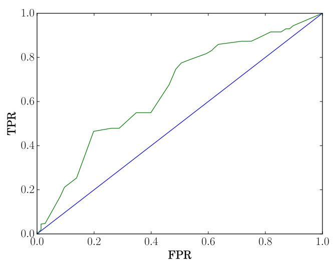


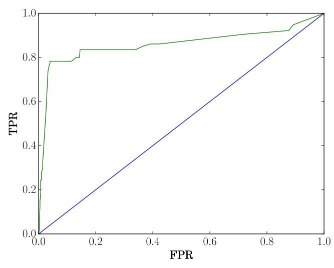
Figure 8.3: Usage probability prediction and real occurrence for each day of a dishwasher during 2015/2/1 to 2015/2/6 (l.r.t.b). The gray area marks the time the appliance is switched on, the curve the probability of the device being switched on at each minute. For a clearer demonstration, we chose an episode length of 120 minutes; the x-axis is labeled by the hour of the day.



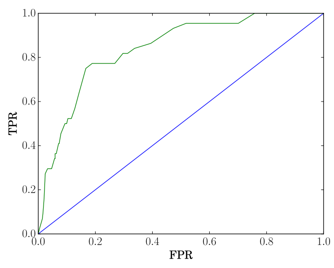
(a) H#0 – dishwasher



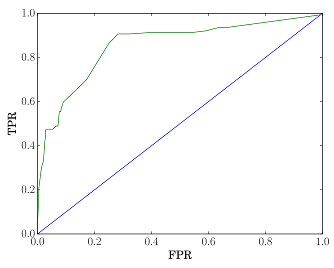
(b) H#1 – dishwasher



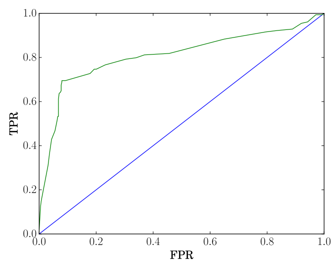
(c) H#1 – dryer



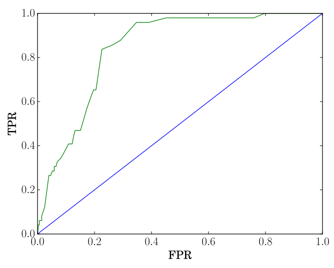
(d) H#1 – washing machine



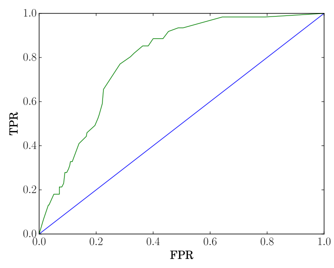
(e) H#2 – dishwasher



(f) H#2 – washing machine



(g) H#3 – dishwasher



(h) [H#3 – washing machine

Figure 8.4: Examples of ROCs of the prediction algorithm on the GREEND data set. Dishwasher, washing machine, and dryer were selected as these appliances are particularly suited for load shifting. A perfect classifier would show a rectangular curve, while the main diagonal indicates complete randomness.

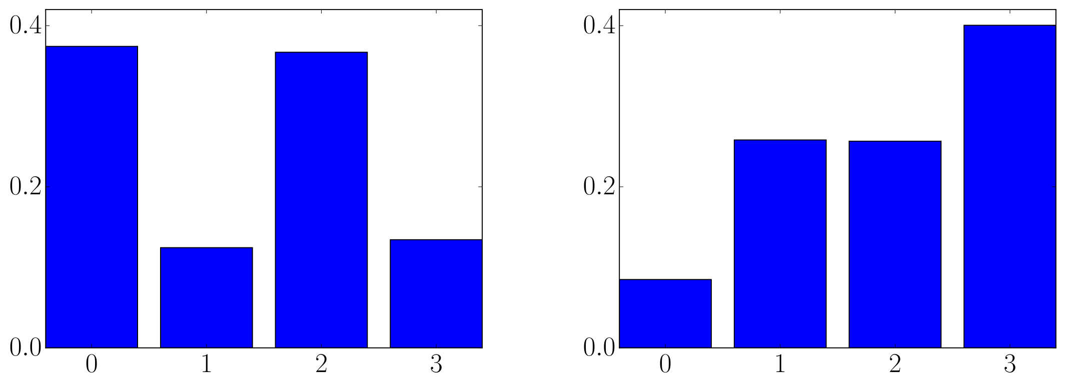


Figure 8.5: $P(D)$ of the dishwasher in house 0 shows a significant difference between (a) training and (b) evaluation.

So, unlike the GREEND scenario, where 60% of the data were used for training and 40% for testing, here all data preceding the predicted day are used for training.

Data

The DEDDIAG dataset is described in Chapter 5. For this experiment, the five washing machines and four dishwashers providing manual annotations are used. The appliances belong to 5 different houses. The measurements are extracted from the first fully metered day to the last fully metered day. Since there are missing measurements in the dataset, for some days, the prediction cannot be validated. To overcome this problem, days that are affected by a recording gap >24 h have been excluded from the prediction. This missing data may also affect the training, as we do not know the behavior on the days with missing data.

Since $P(D)$ and $P(E)$ need to be initialized, we cannot start the evaluation on day one. The initial training is performed using the first 4 weeks of available data. The evaluation is then proceeded by predicting starting at midnight, based on all available data preceding the predicted day. The initial last event required to determine $P(E)$ is saved from training. For each successive day, $P(D)$ and $P(E)$ are recalculated.

Events

The events are extracted using the thresholding algorithm as described in the previous Chapter 7 and as used in previous evaluation on GREEND. Since ground truth exists, the algorithm can be trained based on the available manual annotations. It was decided to not fully retrain the algorithm but to use the parameters found during evaluation in Section 7.8. Thus, the used parameters are derived from a 5-fold evaluation. The event ranges are taken as the minimum and maximum values of all folds (cf. Table 7.2). Thresholds are taken as the average overall folds (cf. Table 7.1). All parameters used in this evaluation are listed in Table 8.2. Using the parameters from all 5 folds also means that the segmentation algorithm uses all annotated data of each appliance. This annotated data is overlapping with this test data, thus not making this a full test of both segmentation and prediction but only prediction.

Table 8.2: Parameters used for the event segmentation using the thresholding algorithm. The parameters are based on the evaluation presented in Chapter 7.

Item	House	Category	Events	Threshold	Window size	Event range
2	1	Washing Machine	34	2.6167	19	1504 – 13933
4	1	Dishwasher	77	0.3337	12	472 – 11759
5	5	Dishwasher	307	0.2741	1	879 – 17966
6	5	Washing Machine	265	0.2931	1	345 – 12485
12	2	Washing Machine	76	1.9314	7	2424 – 11280
19	4	Dishwasher	673	0.1567	4	890 – 10925
20	4	Washing Machine	879	0.3662	1	541 – 24815
24	8	Washing Machine	168	0.2861	9	1387 – 24566
26	8	Dishwasher	242	0.0675	5	1660 – 8794

Results

The available data for each appliance in the DEDDIAG dataset vary, thus the direct comparison between appliances is difficult. The performance is measured using the same performance metrics as for the GREEND dataset. Also the same episode length of 360 minutes was used, splitting the day into 4 bins. An overview of the resulting AUC, F_1, and MCC are show in Table 8.3. Although the results cannot be directly compared with the evaluation approach performed on the GREEND dataset, it can be clearly shown that the F_1 and MCC are lower.

No appliance achieves a MCC above 0.5, but also only one achieves a MCC below 0.1. In terms of predicting human behavior that is always subject to randomness, everything above randomness can be seen as step towards usage prediction that aids a DRM recommendation system. The washing machines achieve MCCs of 0.4276, 0.2098, 0.0987, 0.3117 0.1797 where the corresponding AUCs are 0.7302, 0.6898, 0.6109, 0.7343, 0.6301. Dishwashers achieve MCCs of 0.3048, 0.1849, 0.2475, 0.2040 where the corresponding AUCs are 0.7631, 0.6377, 0.6924, 0.6191. The best-performing washing machine has item id 4 and also is the appliance with the least amount of available data of only 51 days resulting in 34 usages. The worst-performing appliance, on the other hand, was id 12. This poor performance is explained by the poorly performing segmentation algorithm. The segmentation using the parameters determined in training results in only 76 events for a recording period of 733 days. The appliance consumes electricity above the threshold until the resident manually switches off the washing machine. This in combination with the resident's choice of switching the washing machine on before going to bed means that the machine consumes energy above the threshold throughout the night, even when it is idle. Since the segmentation algorithm uses filtering based on event length, such usages are not seen as correct. This problem was also shown by the low JTES results of 0.3831 during evaluation of the segmentation. This highlights the close coupling of event segmentation and usage prediction, as the prediction can only be trained and evaluated correctly with a working segmentation algorithm. While splitting events into two does not influence the algorithm, detecting long stand-by times as ON times does heavily influence the performance of this approach. If the washing machine usually

Table 8.3: Result from day ahead prediction on DEDDIAG

Item	House	Category	AUC	F_1	MCC
2	1	Washing Machine	0.7302	0.5231	0.4276
4	1	Dishwasher	0.7631	0.3946	0.3048
5	5	Dishwasher	0.6377	0.3067	0.1849
6	5	Washing Machine	0.6898	0.2787	0.2098
12	2	Washing Machine	0.6109	0.0863	0.0974
19	4	Dishwasher	0.6924	0.3893	0.2475
20	4	Washing Machine	0.7343	0.4373	0.3117
24	8	Washing Machine	0.6301	0.2725	0.1797
26	8	Dishwasher	0.6191	0.3930	0.2040

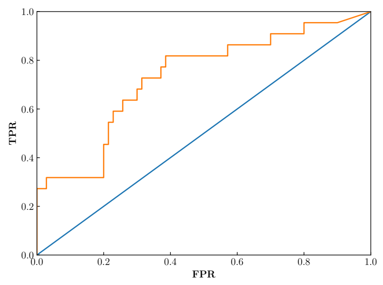
runs for 3 h, but the standby time of random length is seen as a usage, then the calculation of $P(D)$ will be wrong.

The best-performing dishwasher with a MCC of 0.3048 is again the one with the least number of detected events. A small number of events also means that the prediction was not tested against many behavioral situations that occur in a household. So, the good score may simply be statistically irrelevant. Additionally, it would be worth testing to adapt the approach to only using the most recent events when calculating $P(D)$ and $P(E)$ in order to adapt to behavioral changes over time.

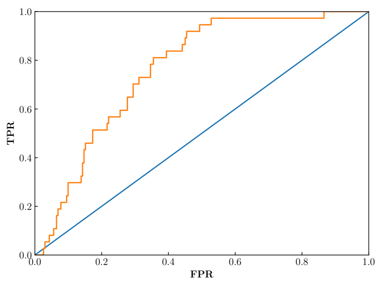
8.5 Summary

In this chapter, a probabilistic model for appliance usage prediction based on historical energy data was presented. The algorithm provides a solution to the last step of the MLDR. The results show that the performance of the algorithm varies between different households and appliances. The results on the GREEND dataset, tested using a simple train-test split approach, look promising. Results on the DEDDIAG dataset are less promising but are also better than random guessing.

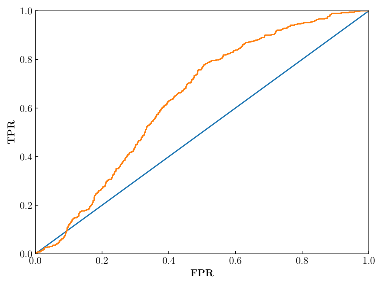
It is not yet clear what is the best and most meaningful performance evaluation metric for this sort of prediction problem. In contrast to usual classification problems, the goal is not to predict the *exact* time an appliance is used but rather to give a recommendation at convenient times. The results here have been evaluated using AUC, F_1, and MCC metrics and can act as baselines for future work. The most challenging elements of usage prediction are behavior changes. Evaluation on the DEDDIAG dataset also showed that insufficient segmentation will lead to insufficient predictions. While the approach presented is not influenced by effects such as splitting an event into two, it is heavily influenced by incorrect or inaccurate annotations of length. Future work will need to investigate how to handle long-term behavior changes such as those found in house 0 of the GREEND data set for the dishwasher. It will also be worth investigating ways to adapt the model over time.



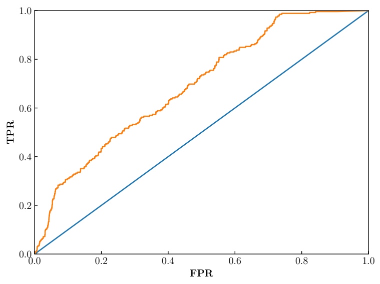
(a) H#1, Item 2 – Washing Machine



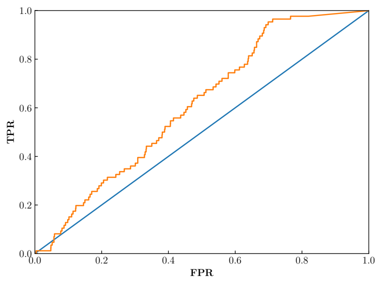
(b) H#1, Item 4 – Dishwasher



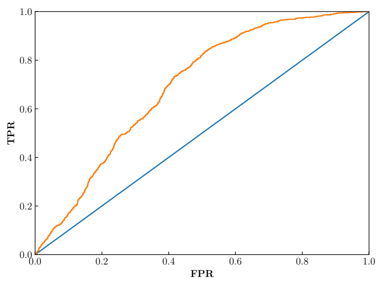
(c) H#5, Item 5 Dishwasher



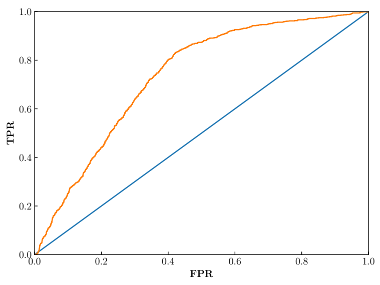
(d) H#5, Item 6 Washing Machine



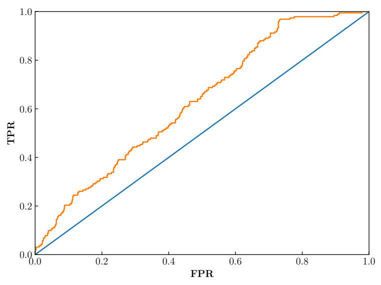
(e) H#2, Item 12 Washing Machine



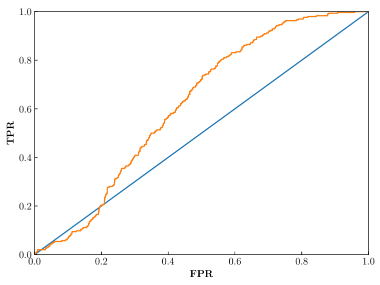
(f) H#4, Item 19 – Dishwasher



(g) H#4, Item 20 – Washing Machine



(h) H#8, Item 24 – Washing Machine



(i) H#8, Item 26 – Dishwasher

Figure 8.6: ROC plots of the day ahead prediction task evaluation on appliances in the DEDDIAG dataset.

CHAPTER 9

Outlook

All elements of the electricity grid will see significant changes over the next few decades; distributed production from photovoltaic systems will increase, fluctuations from wind and solar farms will increase, battery storage capacities in private households will increase, and demand will increase as electric cars become more widely used.

A well-maintained demand and supply equilibrium is and will continue to be a key factor for the electricity grid to maintain quality. In financially rich and technically well-developed countries, this is currently achieved by maintaining sufficient standby capacities. This results in a definition where keeping the discomfort away from customers is seen as the number one quality factor. This is and will remain the biggest obstacle in the road to distributing Demand Response (DR).

In the face of these upcoming changes, the development of intelligent assistance in handling the complexity increase seems inevitable. After 36 years of research in computer-aided DR, significant advances have been made. At the same time, only when all parts of the Machine Learning Demand Response Model (MLDR) are advanced and connected, will customers benefit from these systems. The main steps that the research needs to establish over the next few years in machine learning concerning DR are: (1) Overcoming research challenges posed at all steps of the model, (2) Introducing software frameworks that connect all the steps of the MLDR, (3) Performing comparable user studies using these frameworks.

Scientific challenges similar to MNIST have not been established, making most published research impossible to compare. Future researchers must move from isolated, insufficiently documented experiments to open-source challenges, allowing others to evaluate the many techniques and algorithms proposed. Further challenges for identification, segmentation, and prediction will professionalize the research area to a point where new ideas can be compared to previous work. Challenges for the identification and segmentation have been published as part of the Domestic Energy Demand Dataset of Individual Appliances in Germany (DEDDIAG) dataset publication. This thesis contributed to work that can be challenged for identification, segmentation, and prediction; all on one dataset, DEDDIAG.

By making the data collection system open source, the complexity of collecting data was significantly reduced and future research can benefit. Creating additional long-term datasets that span multiple years will allow researchers to evaluate behavior changes. In addition to the raw monitored energy data, information such as demographic data and manual annotations are needed for both new and established datasets. No dataset currently reflects user behavior changes that are introduced by DR. While behavior change does not affect identification or segmentation, it affects usage prediction. Thus, a future dataset that contains documented user behavior changes based on DR recommendation is needed to evaluate prediction algorithms in a scenario where they are applied.

Identifying appliances is the most advanced and least complex task in the MLDR. The current research state shows that high versus low sample rates are, to some extent, a question of response time when performing real-time analysis. Research shows that in Intrusive Load Monitoring (ILM) scenarios the appliances can be reliably identified using low sample rate data. In these scenarios, approaches for appliance category identification can, for real-world scenarios, be extended beyond the per-window classification task, which will increase the performance even further. No clear answer exists for Non-Intrusive Load Monitoring (NILM) scenarios, which seem to be the main reason for high sample rate approaches. Advancing appliance identification to work on NILM is of high interest considering the widespread installation of smart meters. At the same time, with the development of Internet of Things (IoT) devices, adding the metering capability to each of the relevant appliances seems feasible.

Finding appliance usage in monitored, historic data is an under-researched topic. The importance of the event segmentation task is underestimated: Load and usage patterns cannot be derived without finding the start and stop of cycle-based appliances. The common thresholding algorithm was shown to work effectively when fine-tuned for single appliances, and by developing the Support Vector Machine (SVM) based approach, the beginnings of a new kind of segmentation algorithm were made. However, its performance leaves space for improvements and needs to be advanced to a point where it functions on appliance categories, not only one specific appliance model. By developing the Jaccard-Time-Span-Event-Score (JTES) and publishing an annotated dataset, as well as providing challengeable results, a first step has been made toward comparable research. Fast improvements can be expected by adding annotations to existing datasets and evaluating the results based on performance metrics such as the JTES.

Predicting appliance usage also means predicting human behavior. Our everyday decisions depend on many factors, which are only partially observable in historic electricity data. Having the prediction rely on historical electricity data will therefore limit its capabilities. It can therefore be expected that other data sources that help to identify household occupancy and habits need to be incorporated. The prediction based on electricity data greatly depends on the performance of the segmentation task but their interconnection and dependency have not yet been part of scientific studies. As the last step in the MLDR, errors in the previous steps will impact the usage prediction in end-to-end scenarios. Manually annotated datasets and better segmentation algorithms are the basis for comparable usage prediction research. Of all the steps, usage prediction comes with the most open-ended questions. Statistical prediction models will have to be adapted to behavior change, especially considering behavior changes are the goal of DR.

As with any scientific field, evolution derives from repeatable and comparable research. It is time for research in machine learning DR to derive more comprehensive conclusions, establish challenges within the research community, and push towards comparable research, so that any individual piece of research may stand on the shoulders of giants.

CHAPTER 10

Summary

In the year 2023, the urgency to transform the world's resource usage into a sustainable system is an undeniable reality. In Germany, in 2020, half the country's electricity was already generated from renewable energy sources. By 2050, the share of electricity gained from renewable resources will be significantly higher than today in order to reach the goal to decrease greenhouse gas emissions by 80%. The under- and overproduction from wind and solar introduce high fluctuations on the supply side. The European grid stability is well managed on the grid level and the increasing problems arising from the increased share of renewable energy sources are mitigated using traditional incidence response activities. Some of these activities are simply taking wind turbines off the grid in times of overproduction. However, in times of underproduction, gas turbines, and other stand-by power plants are used to compensate for the increased demand. Adding the demand side to the control equation through Demand Response Management (DRM) is seen as a major step toward mitigating the problems introduced by fluctuations.

While the research in computer-aided Demand Response (DR) has come a long way since the first publications on the topic, the impact on our everyday life is barely visible. It is challenging to imagine that when the research in computer-aided DR began 36 years ago, the participation of private households in the grid stability remains in its infancy. Having private households participate in load balancing requires incentives and adjusting the demand to the supply creates discomfort for residents. Financial incentives, in the form of dynamic electricity prices, are starting to emerge. In an ideal and oversimplified scenario, the optimum DR policy is: run the appliance whenever the electricity price is lowest. The optimum policy to minimize the electricity cost must be operated by someone. Studies showed that especially Real Time Pricing (RTP) models require a complexity reduction to be accepted by the end-user. Thus, the use of dynamic pricing is still not widespread. The effectiveness of all domestic DR programs depends heavily on the end-users willingness to participate. Manual DRM is cumbersome, and electricity is, for most people, not a decisive variable in their budget. Thus, machine learning can be the enabler for DR by lowering the required involvement by providing detailed insights into an individual's electricity usage and providing recommendations or by directly taking automated actions such as delaying the start of an appliance.

The desired information for DRM is: Which appliance will be used, when will it run, and how long will it take? Appliance demand, profile, and shifting boundaries as well as the residents' disutility form a complex system to be optimized. This optimization requires data, the centerpiece of most machine learning research. Data can be sourced using a wide range of features, including, but not limited to geographic features, standard load profiles, weather data, and more direct measures such as high-resolution electricity monitoring using smart meters. In DR the data is described as macroscopic or microscopic. Macroscopic data do not directly relate to the individual household and rather provide insight into the residential sector the household is in. Microscopic data have a direct relation to the household and residents. With the primary source being smart meters, microscopic data have the potential to give direct electricity consumption information for a specific household. The required load monitoring to gather microscopic data is described as Intrusive Load Monitoring (ILM) and Non-Intrusive Load Monitoring (NILM). The main difference lies within the number of metering points used to monitor a household's electricity usage. In NILM only a single metering point is used, relying on decomposing the aggregated load into its components through mathematical algorithms. In ILM, multiple metering points are located throughout the household, providing direct metering of different areas or even appliances. Next, to load monitoring, secondary sources of data such as occupancy, location, and appliance-specific labels are used. While smart meters are being installed throughout the world, the need to extract knowledge from the monitored data is rising. In detail, ILM and NILM are closely related, as the goals in both cases are the same, and NILM only adds the complex task of disaggregation to an already complex task.

Over the years a number of datasets containing microscopic data of private households have been made available for research purposes. These datasets are the basis for research within the field of computer-aided DR. Knowledge is retrieved from data using a bottom-up or top-down approach, depending on whether the data source is microscopic or macroscopic. Most research is put into bottom-up approaches, which on the one hand are more intrusive as they provide more detailed information about a single household, but on the other hand, that information can be processed within the household without affecting privacy concerns. The focus of the techniques and algorithms in this thesis is bottom-up approaches. A new perspective on the steps required to convert collected data into actions or recommendations in bottom-up scenarios called Machine Learning Demand Response Model (MLDR) is presented. The model is organized in three levels: *data* $\rightarrow$ *knowledge* $\rightarrow$ *action*. Knowledge is a valuable asset derived from data that aids decision-making. A specific knowledge extraction model is presented, showing the different steps required for machine learning based DR using a bottom-up approach. These steps are data acquisition, identifying appliances, segmentation of usage events, creating of load profile and usage pattern, and finally predicting the usage. This thesis contributes work to all these required knowledge extraction steps and provides a critical view of the current state of scientific research. Each of the disciplines has received attention since DR emerged in the 1980s.

Data are the starting point for all research in this area. Datasets usually provide individual load monitoring of various household appliances as well as the total consumption in the form of monitoring of the three-phase mains. Over the years, multiple datasets have been released for research purposes. The datasets can in general be differentiated by: location, count, duration, sample frequency, level at which data were collected (whole house, rooms, plugs, appliances), and any additional information provided. Collecting load-monitoring datasets over long periods of time is a non-trivial task, especially when recorded over a long time in a real-world residential environment. Sample rates vary from 1/60 Hz to 250 kHz, and their duration from 1 h to multiple years. As part of a load-shifting research project, a new dataset called Domestic Energy Demand Dataset of Individual Appliances in Germany (DEDDIAG) was collected and released under an open-source license. The dataset focuses on appliances where the load can be shifted, such as dishwashers and washing machines. It provides data of 15 homes over a period of up to 3.5 years, containing, in total, 50 appliances monitored at 1 Hz. A new monitoring system was developed and the description and developed software was also released under an open source license. Unlike previously described monitoring systems for research purposes, the system uses hardware that is readily available for purchase at a very low cost. It was designed so that the per-appliance monitoring equipment could be installed by non-technical people, with the goal that the participating residents are able to install the system on their own. The collected data in each household is sent to a central server, creating the DEDDIAG dataset. Since no available dataset provides ground truth of appliance usages, manual annotations were added to the system. The manual annotations were created using a developed, collaborative web application.

The second step after data acquisition is appliance identification; the task of associating an appliance measurement with an appliance category or model. In machine learning terms this presents a classification problem. Approaches can be separated into two groups: low and high sample rates. A low and high sample rate approach is presented and developed for a sample rate of 1 Hz. The low sample rate approach was developed and published as a baseline for the DEDDIAG dataset. Sliding windows are used to convert the continuous time series into data vectors of event lengths. For each window, a feature vector is created using the window's mean value as well as the coefficients of a Discrete Wavelet Transformation (DWT). The classification is performed using the k-Nearest-Neighbor (kNN) classifier for nine different categories: Coffee Machine, Dishwasher, Dryer, Freezer, Office Desk, Other, Refrigerator, TV, and Washing Machine. K-fold cross-validation has been used to evaluate the performance using a sophisticated splitting approach. Cross-validation on long time-series data is not trivial, especially in a scenario where the data of each category originate from different appliances. The evaluation using different time window sizes reveals that above window sizes of 256 seconds, all appliances can be detected with a F_1 score greater than 0.8. Following this idea, a high sample rate approach called Recurrence Plot Spacial Pyramid Pooling (RPSPP) was developed. It uses features called voltage-current (V-I) trajectory, a 2-dimensional feature containing one normalized wave cycle of voltage and current. The V-I trajectory is transformed into two unthresholded Recurrence Plots (RPs), which are then classified using a deep neural network. A similar approach called Weighted Recurrence Plot (WRG) was previously proposed, relying on handcrafted parameters for each evaluated dataset. The RPSPP does not rely on such parameters and still outperforms the proposed method. During an evaluation of their work, it was found that the authors of WRG published false results, as their described evaluation differed from their implementation. Finding this deviation was possible because the authors published some of their source code. An evaluation of the new RPSPP algorithm, as well as a re-evaluation of the WRG with a corrected evaluation method, showed that RPSPP has similar or superior performance. The algorithms were evaluated in a 5-fold cross-validation as well as a leave-one-group-out evaluation. These were performed on three published high-sample rate datasets, namely: COOLL, WHITEDv1.1, and PLAID.

The knowledge extraction step that follows the appliance identification is segmentation; the forgotten task that has received little research attention. Segmentation requires extracting usage events from appliance measurements in the form of start and stop timestamp pairs. The most commonly used approach to find appliance usage is a lower bound threshold algorithm. The thresholding algorithm predicts the appliance being ON or OFF (steady state) for each window. The approach is frequently used and mentioned in publications but was never evaluated. Since event annotations are available in the DEDDIAG dataset, the algorithm was evaluated for the first time. As part of this test, a new performance metric called Jaccard-Time-Span-Event-Score (JTES) was proposed to evaluate the event as a whole instead of evaluating every data point in a time series. The JTES metric is designed to evaluate the segmentation results in relation to their real-world implications, punishing unwanted scenarios. Evaluating the thresholding approach using the JTES showed that for many appliances the thresholding algorithm works well, but also showed a major problem: The thresholding algorithm relies on the hypothesis that the start and stop of an appliance are separated by a single power usage threshold. This does not hold for many appliances, especially washing machines which may have a high stand-by electricity consumption until the residents unload the machine. To overcome this limitation, a Support Vector Machine (SVM) based classification approach was developed. The approach classifies the start and stop and combines both into a single event, overcoming the limitations of a thresholding approach. The SVM approach was developed before the JTES and showed great potential based on data point evaluation. Re-evaluating the algorithm based on the JTES showed that it also has limitations. For some appliances with a simple load profile it has similar performance as the thresholding, but in most the simple thresholding still outperforms the SVM approach. This forgotten task is a very important step and requires much more research attention. The thresholding evaluation was published as part of the DEDDIAG dataset, presenting a challenge baseline for other research.

The final step in knowledge extraction is appliance usage prediction. It is performed to predict future usage, thus providing the user with relevant recommendations based on their predicted future behavior. The development of usage prediction heavily depends on segmentation. Appliance usage is commonly used to train prediction algorithms and is absolutely necessary for evaluation. A proposed probabilistic model trained on historical appliance usage combines two probabilities. Firstly, the model calculates a resident's usage preference per time slot over a certain period by dividing each day into several episodes. Secondly, the time between usages is modeled as an exponential distribution. The events required to estimate the probabilities are segmented using the lower bound thresholding algorithm. Two evaluations are presented based on GREEND as well as the DEDDIAG dataset. Predicting appliance usage solely on historical usage is challenging because many factors influence a person's decision to use an appliance. Therefore, one cannot expect to predict the usage perfectly, but the results of this research show that the resident's preferences can be modeled using the proposed probabilistic model.

CHAPTER A
DEDDIAG

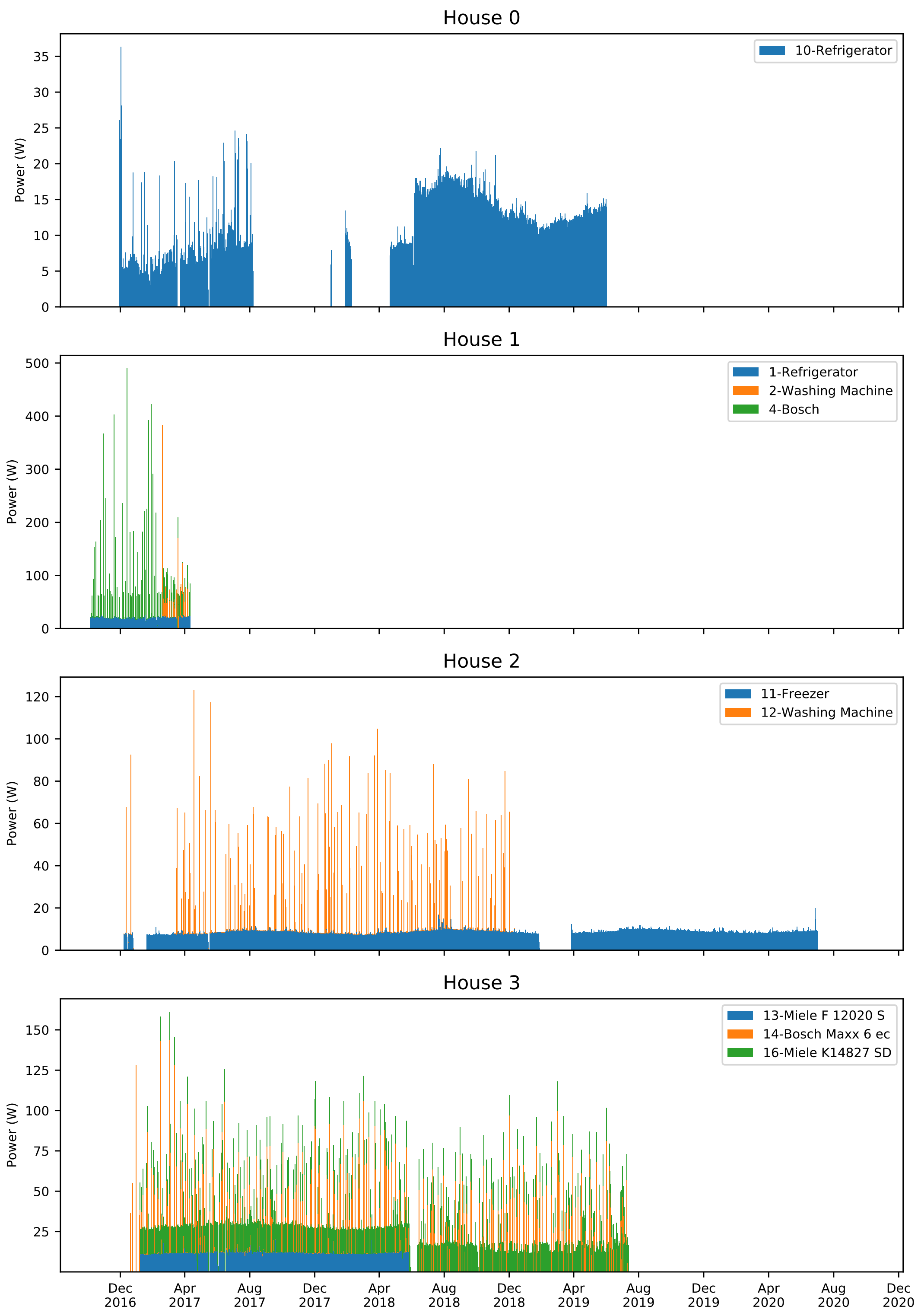


Figure A.1: Stacked average daily power demand of monitored appliances in Houses 0–3 of the Domestic Energy Demand Dataset of Individual Appliances in Germany (DEDDIAG) dataset. X-axes ranges are aligned, Y-axes are not aligned to increase per house readability. Each color block is stacked and shows daily average power demand of an appliance. [Wenn 21b], License: CC-BY 4.0.

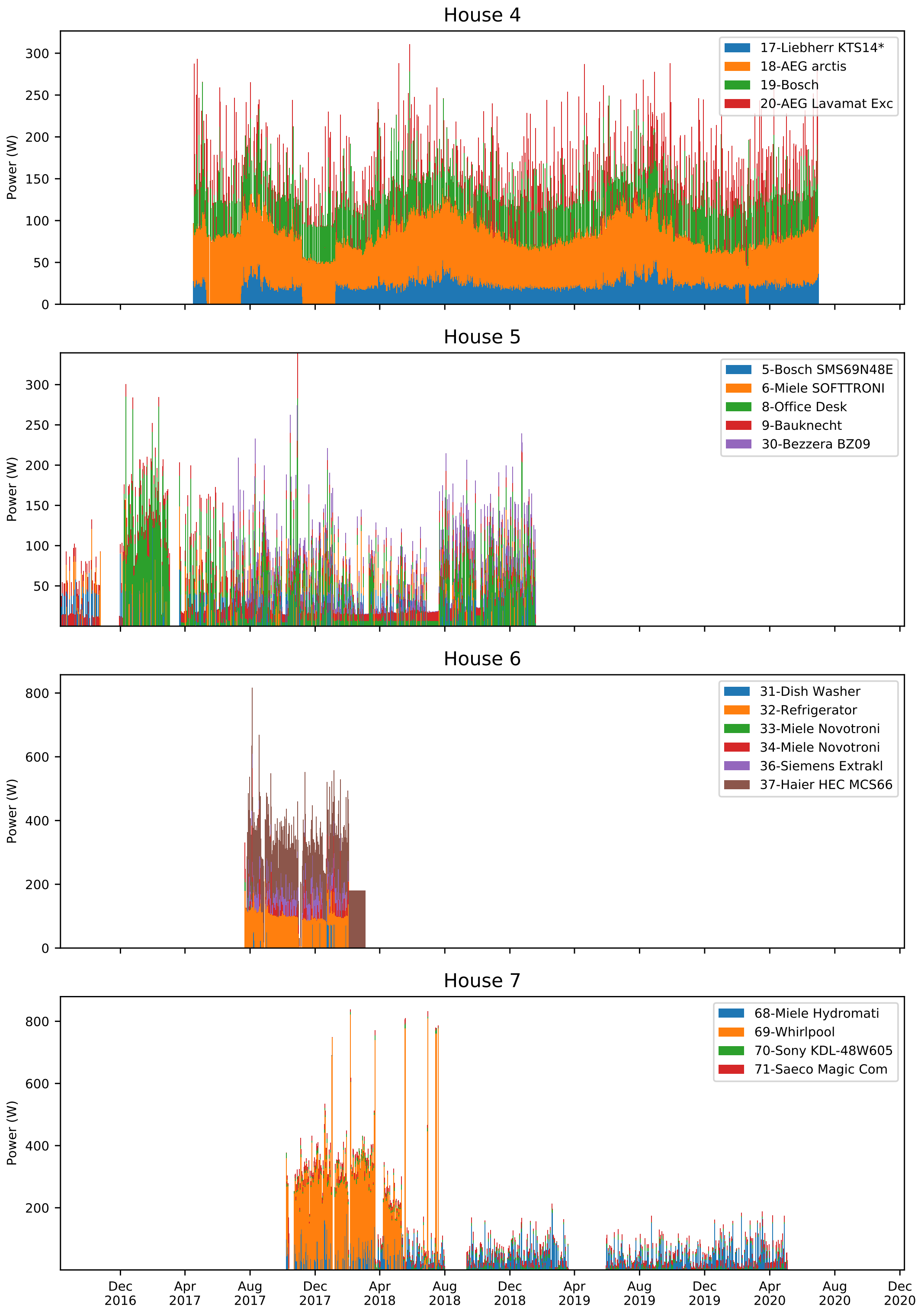


Figure A.2: Stacked average daily power demand of monitored appliances in Houses 4–7 of the DEDDIAG dataset [Wenn 21b]. X-axes ranges are aligned, Y-axes are not aligned to increase per house readability. Each color block is stacked and shows daily average power demand of an appliance. [Wenn 21b], License: CC-BY 4.0.

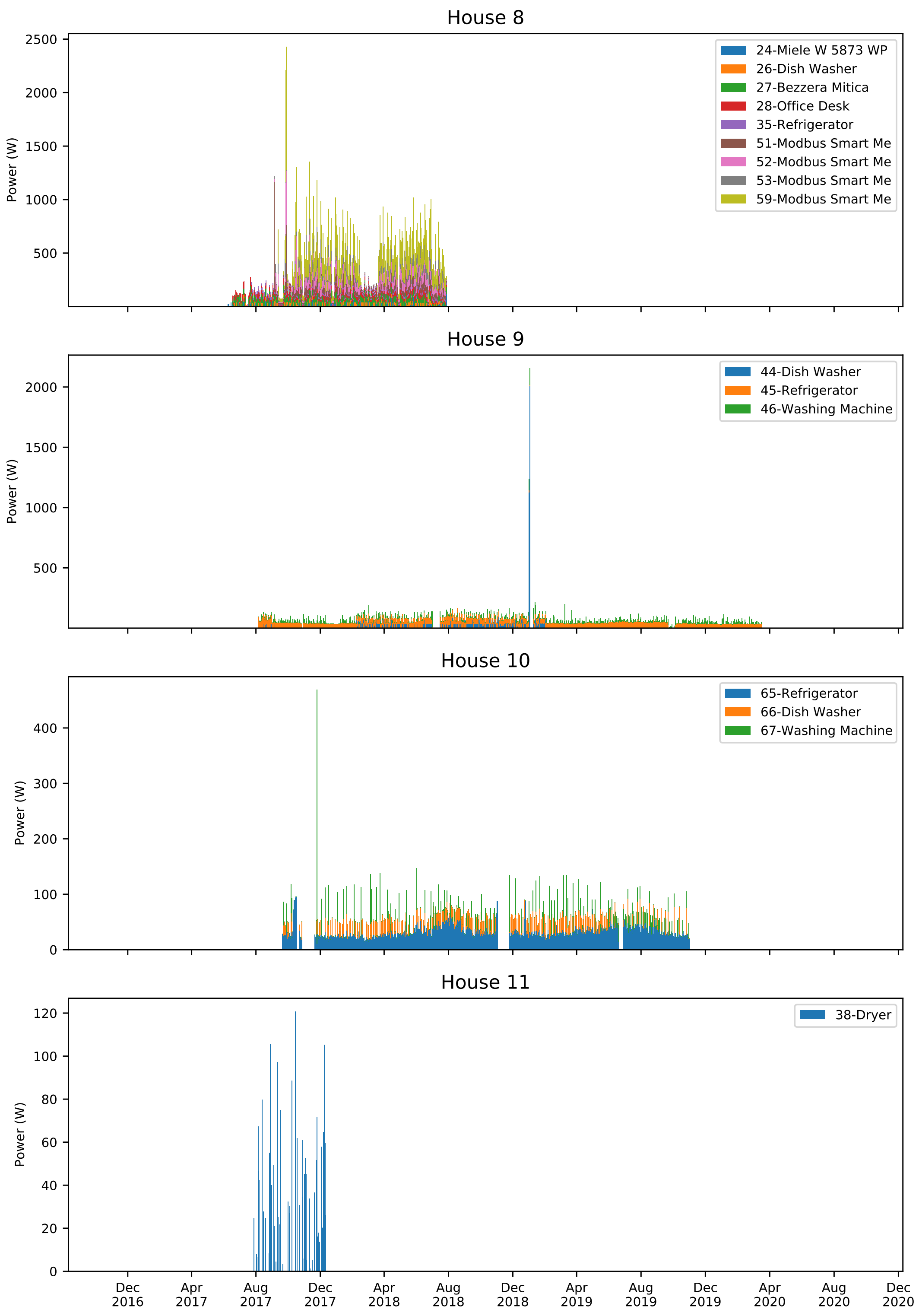


Figure A.3: Stacked average daily power demand of monitored appliances in Houses 8–11 DEDDIAG dataset. X-axes ranges are aligned, Y-axes are not aligned to increase per house readability. Each color block is stacked and shows daily average power demand of an appliance. [Wenn 21b], License: CC-BY 4.0.

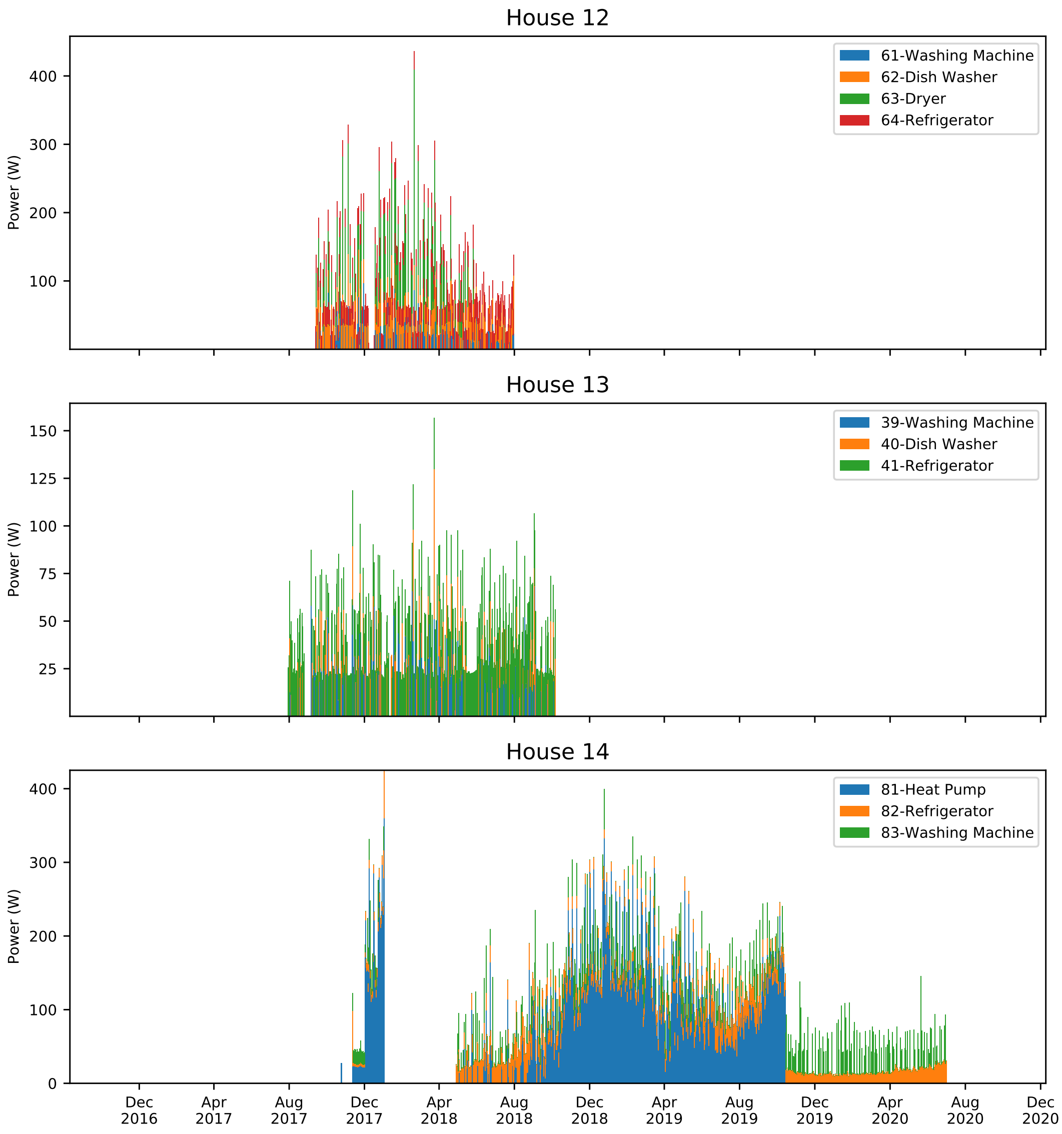


Figure A.4: Stacked average daily power demand of monitored appliances in Houses 12–14 DEDDIAG dataset. X-axes ranges are aligned, Y-axes are not aligned to increase per house readability. Each color block is stacked and shows daily average power demand of an appliance. [Wenn 21b], License: CC-BY 4.0.

Table A.1: List of houses and appliances observed in the DEDDIAG dataset. The columns "Missing > 1h5 sec" and "Missing > 1day" provide an indicator for missing data, defined as the sum of gaps in relation to the duration.

Item ID	Category	Type	First date	Last date	Duration	Missing > 1h5sec	Missing > 1day
House 0							
10	Refrigerator		2016-11-30 20:24:05	2019-06-02 17:56:17	913 days	19.29%	7.85%
House 1							
1	Refrigerator		2016-10-06 19:25:07	2017-04-11 16:00:06	186 days	7.44%	2.91%
2	Washing Machine		2017-02-18 15:01:05	2017-04-11 15:00:00	51 days	22.17%	2.80%
4	Dish Washer	Bosch	2016-10-06 20:18:07	2017-04-11 16:00:04	186 days	59.40%	49.26%
House 2							
11	Freezer		2016-12-08 15:21:13	2020-08-20 22:00:07	1351 days	6.85%	6.76%
12	Washing Machine		2016-12-08 15:24:12	2018-12-12 14:43:51	733 days	14.73%	11.30%
House 3							
13	Freezer	Miele F 12020 S-3	2017-01-07 09:57:15	2018-05-27 07:59:33	504 days	2.50%	2.07%
14	Washing Machine	Bosch Maxx 6 ecoSpar	2016-12-20 09:18:54	2019-07-13 10:00:00	935 days	6.57%	3.34%
16	Refrigerator	Miele K14827 SD ED/CS	2017-01-07 09:57:15	2019-07-13 10:00:07	917 days	3.56%	3.27%
House 4							
17	Refrigerator	Liebherr KTS14*	2017-04-17 19:33:30	2020-11-16 09:32:07	1308 days	1.02%	0.78%
18	Freezer	AEG arctis	2017-04-17 19:15:04	2020-12-09 09:59:13	1331 days	0.68%	0.44%
19	Dish Washer	Bosch	2017-04-17 19:51:27	2020-12-09 09:00:00	1331 days	0.46%	0.22%
20	Washing Machine	AEG Lavamat Exclusive 54569	2017-04-17 19:17:14	2020-12-09 09:00:00	1331 days	0.78%	0.31%
House 5							
5	Dish Washer	Bosch SMS69N48EU	2016-08-10 23:26:28	2019-01-18 08:00:00	890 days	32.59%	12.78%
6	Washing Machine	Miele SOFTTRONIC W2241	2016-08-16 10:32:18	2019-01-17 17:08:06	884 days	21.39%	10.25%
8	Office Desk		2016-12-06 12:58:02	2019-01-18 08:00:08	772 days	4.81%	4.44%
9	Refrigerator	Bauknecht	2016-08-10 14:10:32	2019-01-18 08:00:00	890 days	10.40%	8.23%
30	Coffee Machine	Bezzera BZ09	2017-06-28 17:26:00	2019-01-18 08:00:00	568 days	6.19%	1.06%
House 6							
31	Dish Washer		2017-07-22 09:00:03	2018-02-01 23:00:00	194 days	5.27%	5.24%
32	Refrigerator		2017-07-22 08:54:03	2018-02-01 23:00:00	194 days	5.24%	5.21%
33	Washing Machine	Miele Novotronic W1514	2017-07-22 09:16:03	2017-08-12 17:00:00	21 days	0.00%	0.00%
34	Dryer	Miele Novotronic T7644C	2017-07-22 09:13:02	2018-02-01 16:00:00	194 days	5.34%	5.27%
36	Dish Washer	Siemens Extrakl. Festival Spuler	2017-07-26 21:20:02	2018-02-01 23:00:00	190 days	3.65%	3.62%
37	Refrigerator	Haier HEC MCS662FIX	2017-07-26 21:11:03	2018-03-05 10:00:00	221 days	4.63%	4.60%
House 7							
68	Washing Machine	Miele Hydromatic W701	2017-10-08 15:01:04	2020-12-09 09:40:09	1157 days	29.35%	29.12%
69	Other	Whirlpool	2017-10-08 15:01:04	2018-07-20 08:05:50	284 days	35.06%	34.14%
70	TV	Sony KDL-48W605B	2017-10-08 15:01:04	2020-05-04 11:00:07	938 days	14.13%	13.88%
71	Coffee Machine	Saeco Magic Comfort+	2017-10-08 16:00:00	2020-12-09 10:00:00	1157 days	29.38%	29.12%
House 8							
24	Washing Machine	Miele W 5873 WPS Edition 111	2017-06-06 15:29:23	2018-07-28 08:00:12	416 days	1.13%	1.01%
26	Dish Washer		2017-06-18 13:47:12	2018-07-28 08:00:00	404 days	1.13%	1.04%
27	Coffee Machine	Bezzera Mitica Top MN	2017-06-18 13:37:34	2018-07-28 08:00:12	404 days	1.04%	1.03%
28	Office Desk		2017-06-18 14:13:03	2018-07-28 08:00:13	404 days	1.09%	1.03%
35	Refrigerator		2017-07-23 20:26:03	2018-07-28 08:00:13	369 days	0.01%	0.00%
51	Smart Meter Phase	Modbus Smart Meter Phase 1	2017-09-05 15:45:29	2018-07-28 08:00:00	325 days	0.28%	0.00%
52	Smart Meter Phase	Modbus Smart Meter Phase 2	2017-09-05 17:53:03	2018-07-28 08:00:00	325 days	1.83%	1.38%
53	Smart Meter Phase	Modbus Smart Meter Phase 3	2017-09-05 17:55:03	2018-07-28 08:00:00	325 days	9.71%	9.38%
59	Smart Meter Total	Modbus Smart Meter Total	2017-09-12 14:10:03	2018-07-28 08:00:00	318 days	13.97%	13.58%
House 9							
44	Dish Washer		2017-08-05 18:13:18	2019-02-03 10:09:47	546 days	5.86%	5.52%
45	Refrigerator		2017-08-05 18:07:03	2020-03-17 08:00:07	954 days	2.98%	2.92%
46	Washing Machine		2017-08-05 18:13:30	2020-03-17 08:00:00	954 days	2.99%	2.93%
House 10							
65	Refrigerator		2017-09-20 16:04:04	2019-11-01 13:00:00	771 days	8.13%	7.62%
66	Dish Washer		2017-09-20 17:00:00	2019-11-01 13:00:00	771 days	8.18%	7.63%
67	Washing Machine		2017-09-20 16:07:55	2019-11-01 13:00:00	771 days	8.14%	7.62%
House 11							
38	Dryer		2017-07-27 20:37:35	2017-12-15 09:00:00	140 days	0.28%	0.00%
House 12							
61	Washing Machine		2017-09-13 16:55:35	2018-07-31 11:00:00	320 days	3.29%	3.29%
62	Dish Washer		2017-09-13 16:50:13	2018-07-31 11:00:00	320 days	3.27%	3.27%
63	Dryer		2017-09-13 17:00:00	2018-07-31 11:00:00	320 days	3.30%	3.30%
64	Refrigerator		2017-09-13 16:50:04	2018-07-31 11:00:10	320 days	3.28%	3.28%
House 13							
39	Washing Machine		2017-07-30 10:56:08	2018-10-06 11:00:05	433 days	7.59%	7.56%
40	Dish Washer		2017-07-30 10:30:38	2018-10-06 11:00:09	433 days	3.00%	3.00%
41	Refrigerator		2017-07-30 10:58:59	2018-10-06 11:00:10	433 days	4.73%	4.73%
House 14							
81	Other	Heat Pump	2017-10-24 11:44:22	2019-10-14 16:06:02	720 days	19.08%	19.05%
82	Refrigerator		2017-11-12 14:58:03	2020-12-09 09:59:59	1122 days	10.87%	10.51%
83	Washing Machine		2017-11-12 15:49:05	2020-12-09 09:40:16	1122 days	10.54%	10.52%

Table A.2: List of appliance usage annotation labels as provided in the DEDDIAG dataset.

Label ID	Item ID	Label Name	Item Name	Annotation Count
House 0				
29	10	Light	Refrigerator	17
30	10	Compressor	Refrigerator	191
House 1				
33	1	Compressor	Refrigerator	83
34	1	Light	Refrigerator	64
35	2	Normal	Washing Machine	33
36	2	Short	Washing Machine	4
17	4	PreWash	Bosch	2
18	4	Normal	Bosch	67
House 2				
37	12	Normal	Washing Machine	69
House 4				
2	19	High	Bosch	28
4	19	Low	Bosch	119
19	19	PreWash	Bosch	1
24	19	Error	Bosch	1
6	20	Short	AEG Lavamat Exclusive 54569 Electronic	55
7	20	40 degrees	AEG Lavamat Exclusive 54569 Electronic	65
8	20	60 degrees	AEG Lavamat Exclusive 54569 Electronic	37
11	20	90 degrees	AEG Lavamat Exclusive 54569 Electronic	2
40	20	PreWash	AEG Lavamat Exclusive 54569 Electronic	2
41	20	Unkown	AEG Lavamat Exclusive 54569 Electronic	1
House 5				
9	5	ECO	Bosch SMS69N48EU	201
10	5	Normal	Bosch SMS69N48EU	70
15	5	PreWash	Bosch SMS69N48EU	4
5	6	Normal	Miele SOFTTRONIC W2241	73
21	6	Short	Miele SOFTTRONIC W2241	4
25	6	Error	Miele SOFTTRONIC W2241	1
26	6	PreWash	Miele SOFTTRONIC W2241	3
31	9	Compressor	Bauknecht	97
32	9	Light	Bauknecht	113
House 8				
14	24	Normal	Miele W 5873 WPS Edition 111	100
12	26	Normal	Dish Washer	225
22	26	Medium	Dish Washer	1
23	26	Short	Dish Washer	1

CHAPTER B

Appliance Identification

Table B.1: Original baseline result of appliance category identification using k-nearest neighbors classification as published in [Wenn 21b]. Results show F_1-score for each category at tested window sizes as well as the weighted average over all classes based on a simple 5-fold cross-validation. The values have be reevaluated in this thesis using a corrected cross validation approach (cf. Table 6.1).

Window Size	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**	**2048**
Coffee Machine	0.9137	0.9193	0.9160	0.9214	0.9404	0.9412	0.9304	0.9348	0.9434	0.9552
Dish Washer	0.5163	0.5735	0.6452	0.7193	0.7878	0.8391	0.8634	0.8782	0.8930	0.9113
Dryer	0.8457	0.8542	0.8571	0.8714	0.8901	0.9083	0.9129	0.9293	0.9413	0.9433
Freezer	0.8002	0.8198	0.8391	0.8576	0.8803	0.9042	0.9052	0.9194	0.9468	0.9767
Office Desk	0.7295	0.7567	0.7824	0.8024	0.8219	0.8473	0.8549	0.8844	0.9274	0.9500
Other	0.8517	0.8575	0.8639	0.8685	0.8776	0.8893	0.9041	0.9198	0.9358	0.9556
Refrigerator	0.7010	0.7239	0.7534	0.7857	0.8178	0.8466	0.8675	0.8987	0.9293	0.9504
TV	0.7581	0.8222	0.8725	0.9020	0.9189	0.9335	0.9415	0.9465	0.9556	0.9663
Washing Machine	0.6232	0.6736	0.7295	0.7801	0.7876	0.7904	0.8003	0.8194	0.8420	0.8702
Weighted Average	0.7443	0.7743	0.8043	0.8326	0.8566	0.8764	0.8855	0.9023	0.9231	0.9421

CHAPTER C

Appliance Segmentation

Table C.1: Original baseline result of appliance segmentation using thresholding as published in [Wenn 21b]. While the task for dish washers and washing machines does not distinguish between different labels per item, for refrigerators the two labels compressor (C) and light (L) are evaluated as two separate tasks. Results are evaluated using Jaccard-Time-Span-Event-Score (JTES) and F_1-score. Result is shown for window sizes where JTES was highest in a trial with sizes ranging from 1 to 24.

Item	**Category**	**Labels**	**Window Size**	**JTES**	$\boldsymbol{F_1}$
1	Refrigerator	33 (C)	13	0.8817	0.9967
1	Refrigerator	34 (L)	1	0.1589	0.7960
2	Washing Machine	35, 36	24	0.6055	0.9994
4	Dish Washer	17, 18	22	0.7167	0.9959
5	Dish Washer	9, 10, 15	1	0.2370	0.9958
6	Washing Machine	5, 21, 26, 25	24	0.0385	0.9196
9	Refrigerator	31 (C)	7	0.8855	0.9967
9	Refrigerator	32 (L)	1	0.2713	0.8089
10	Refrigerator	29 (L)	1	0.0403	0.9530
10	Refrigerator	30 (C)	5	0.9348	0.9989
12	Washing Machine	37	24	0.4587	0.9768
19	Dish Washer	2, 4, 19, 24	1	0.4607	0.9370
20	Washing Machine	6, 7, 8, 11, 40, 41	1	0.4974	0.9737
24	Washing Machine	14	1	0.3796	0.9818
26	Dish Washer	12, 22, 23	24	0.0817	0.8722

List of Acronyms

ACC

accuracy 30–32

ALM

Appliance Load Monitoring 38

AUC

Area under the Curve 32, 33, 115, 120, 123, 127, 128

BayWISS

Bayerisches Wissenschaftsforum VII

BMBF

German Federal Ministry of Education and Research VII

CNN

Convolutional Neural Network 4, 28, 70, 80, 81, 83, 155, III, V

CPP

Critical Peak Pricing 14

CUDA

Compute Unified Device Architecture 80

db1

Daubechies wavelet 1 72, 73

DBSCAN

density-based spatial clustering of applications with noise 91

DEDDIAG

Domestic Energy Demand Dataset of Individual Appliances in Germany 4–6, 39, 41, 49, 53, 59, 60, 65, 71, 73, 75, 88, 90, 94, 96, 99, 101, 102, 108, 111, 123, 126–129, 131, 135, 136, 138–143, 156, 157, 159, III–VII

DER

Distributed Energy Resources 13

DFT

Discrete Fourier transformation 68

GOF

goodness-of-fit 91

GPU

Graphics Processing Unit 80

GUI

Graphical User Interface 50, 56, 57, 59, 156

HAR

harmonic distortion 47, 156

HMM

Hidden Markov Model 19, 20, 68

IEA

International Energy Agency 10

ILM

Intrusive Load Monitoring 38, 48, 51, 53, 65, 67, 69, 71, 78, 84, 94, 108, 132, 134

IoT

Internet of Things 16, 84, 132

IoU

Intersection-over-Union 92, 93

IQR

interquartile range 33, 75, 155

JTES

Jaccard-Time-Span-Event-Score 5, 92–94, 100–103, 105–109, 119, 127, 132, 136, 147, 157, 161, IV

kNN

k-Nearest-Neighbor 19–21, 24, 69, 70, 73, 75, 76, 88, 91, 135, 155, 156, 159, III, V

LSTM

Long Short-Term Memory 96

MCC

Matthews Correlation Coefficient 31, 32, 120, 123, 127, 128

MLDR

Machine Learning Demand Response Model 3, 50–52, 65, 67, 68, 88, 89, 113, 128, 131, 132, 134, 156, III, V

MLP

Multilayer Perceptron 25, 26, 155

MSE

Mean Square Error 123

MVP

Minimum Viable Product 55

NALM

Non-Intrusive Appliance Load Monitor 38, 51

NB

Naïve Bayes 19, 20

NILM

Non-Intrusive Load Monitoring 38, 48, 51–53, 59, 64, 67, 68, 84, 92, 108, 132, 134

NILMTK

Non-Intrusive Load Monitoring Toolkit 64

NN

Neural Network 19, 20, 24, 28, 80, 84

PAA

Piecewise Aggregate Approximation 78–80

PDF

Probability Distribution Function 116

ReLU

Rectified Linear Unit 25, 80

RMS

root-mean-square 45, 46, 69–71, 88

RNN

Recurrent Neural Network 96

ROC	Receiver Operating Characteristic 32, 120, 123, 125, 129, 155, 157
ROU	Rate of Use 114
RP	Recurrence Plot 4, 70, 78–80, 88, 135, III, V
RPSPP	Recurrence Plot Spacial Pyramid Pooling 78, 83–88, 135
RTP	Real Time Pricing 14, 15, 114, 133
SGD	stochastic gradient descent 27
SLP	Standard Load Profile 8
SPP	Spacial Pyramid Pooling 80, 81, 88, III, V
SQL	Structured Query Language 54, 56, 59, 62, 64
SVM	Support Vector Machine 4, 19, 20, 22–24, 70, 89–91, 96–98, 100–104, 106–111, 132, 136, 155, 157, 159, IV
tanh	hyperbolic tangent 25
TN	true negatives 29
TOU	Time of Use 14
TP	true positives 29
TPR	true positive rate 32, 120, 155

TSV

tab-separated value 61

V-I

voltage-current 4, 48, 70, 78, 84, 88, 135, 156, III, V

WRG

Weighted Recurrence Plot 80, 82–88, 135

List of Figures

List of Tables

List of Algorithms

List of Self-Citations

Some of the here presented work has been published before. The majority of these publications has been authored by the author of this thesis. The main scientific contributions are based on his own work and thoughts. Most parts of this work were written from scratch and all experiments were repeated. It is therefore possible that some chapters or sections contain previously published contents. Details on the mentioned works are given in the following table.

Chapter	**Reference**
Chapter 5	[Wenn 21b]
Chapter 6, Section 6.3	[Wenn 21b]
Chapter 7, Section 7.6	[Wenn 19b]
Chapter 8	[Wenn 17]

Bibliography

[Abub 16] I. Abubakar, S. Khalid, M. Mustafa, H. Shareef, and M. Mustapha. "Recent approaches and applications of non-intrusive load monitoring". *ARPN Journal of Engineering and Applied Sciences*, Vol. 11, No. 7, pp. 4609–4618, Apr. 2016.

[Adam 84] D. Adams. *The hitch hiker's guide to the galaxy*. Pan Books, London, Nov. 1984.

[Adol 22] J. F. Adolfsen, F. Kuik, E. M. Lis, and T. Schuler. "The impact of the war in Ukraine on euro area energy markets". 2022. https://www.ecb.europa.eu/pub/economic-bulletin/focus/2022/html/ecb.ebbox202204_01~68ef3c3dc6.en.html.

[Alca 17] J. Alcalá, J. Ureña, A. Hernández, and D. Gualda. "Event-Based Energy Disaggregation Algorithm for Activity Monitoring From a Single-Point Sensor". *IEEE Transactions on Instrumentation and Measurement*, Vol. 66, No. 10, pp. 2615–2626, 2017.

[Ande 12a] K. Anderson, M. Berges, A. Ocneanu, D. Benitez, and J. Moura. "Event detection for Non Intrusive load monitoring". pp. 3312–3317, Oct. 2012.

[Ande 12b] K. Anderson, A. Ocneanu, D. Benitez, D. Carlson, A. Rowe, and M. Berges. "BLUED: A Fully Labeled Public Dataset for Event-Based Non-Intrusive Load Monitoring Research". In: *Proceedings of the 2nd KDD Workshop on Data Mining Applications in Sustainability (SustKDD)*, Beijing, China, Aug. 2012.

[Bara 03] M. Baranski and Voss J. "Nonintrusive appliance load monitoring based on an optical sensor". In: *2003 IEEE Bologna Power Tech Conference Proceedings*, 2003.

[Barb 11] A. Barbato, A. Capone, M. Rodolfi, and D. Tagliaferri. "Forecasting the usage of household appliances through power meter sensors for demand management in the smart grid". In: *IEEE Intern. Conf. SmartGridComm, 2011*, pp. 404–409, Oct. 2011.

[Bark 12] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht. "Smart*: An Open Data Set and Tools for Enabling Research in Sustainable Homes". In: *Proceedings of SustKDD*, Beijing, China, Aug. 2012.

[Basu 13] K. Basu, L. Hawarah, N. Arghira, H. Joumaa, and S. Ploix. "A prediction system for home appliance usage". *Energy and Buildings*, Vol. 67, pp. 668–679, 2013.

[Batr 13] N. Batra, M. Gulati, A. Singh, and M. B. Srivastava. "It's Different: Insights into Home Energy Consumption in India". In: *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pp. 1–8, ACM, New York, NY, USA, 2013.

[Batr 14a] N. Batra, J. Kelly, O. Parson, H. Dutta, W. Knottenbelt, A. Rogers, A. Singh, and M. Srivastava. "NILMTK". In: *Proceedings of the 5th International Conference on Future Energy Systems*, ACM, June 2014.

[Batr 14b] N. Batra, O. Parson, M. Berges, A. Singh, and A. Rogers. "A comparison of non-intrusive load monitoring methods for commercial and residential buildings". *arXiv:1408.6595*, 2014.

[Batr 19] N. Batra, R. Kukunuri, A. Pandey, R. Malakar, R. Kumar, O. Krystalakos, M. Zhong, P. Meira, and O. Parson. "Towards reproducible state-of-the-art energy disaggregation". In: *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ACM, Nov. 2019.

[BDEW 19] BDEW Bundesverband der Energie-und Wasserwirtschaft e.V. "Wie heizt Deutschland? - Studie zum Heizungsmarkt". https://www.bdew.de/media/documents/BDEW_Heizungsmarkt_final_30.09.2019_3ihF1yL.pdf, 2019.

[Beck 14] C. Beckel, W. Kleiminger, and R. Cicchetti. "The ECO Data Set and the Performance of Non-Intrusive Load Monitoring Algorithms". In: *Proceedings of the 1st ACM Intern. Conf. BuildSys 2014*, pp. 80–89, 2014.

[Berg 10] M. Berges, E. Goldman, H. Matthews, and L. Soibelman. "Enhancing Electricity Audits in Residential Buildings with Nonintrusive Load Monitoring". *Journal of Industrial Ecology*, Vol. 14, Nov. 2010.

[Berg 11] M. Berges, E. Goldman, H. Matthews, L. Soibelman, and K. Anderson. "User-Centered Nonintrusive Electricity Load Monitoring for Residential Buildings". *Journal of Computing in Civil Engineering*, Vol. 25, pp. 471–480, Nov. 2011.

[Bish 06] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[Bitt 99] R. Bitterer and B. Schieferdecker. "Repräsentative VDEW-Lastprofile, Aktionsplan Wettbewerb". *VDEW Tech. Rep. M32-99*, 1999. https://www.bdew.de/media/documents/1999_Repraesentative-VDEW-Lastprofile.pdf.

[Bose 92] B. E. Boser, I. M. Guyon, and V. N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144–152, ACM, New York, NY, USA, 1992.

[bp p 20] bp p.l.c. "bp Statistical Review of World Energy 2020". https://www.bp.com/content/dam/bp/business-sites/en/global/corporate/pdfs/energy-economics/statistical-review/bp-stats-review-2020-full-report.pdf, 2020.

[Chaf 11] D. Chaffey and G. White. *Business Information Management: Improving Performance Using Information Systems*. Financial Times/Prentice Hall, 2011.

[Chan 13] C. Chang, P.-A. Verhaegen, J. R. Duflou, M. M. Drugan, and A. Nowe. "Finding Days-of-week Representation for Intelligent Machine Usage Profiling". *Journ. of Industrial and Intelligent Information*, Vol. 1, No. 3, pp. 148–154, 2013.

[Chic 20] D. Chicco and G. Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". *BMC Genomics*, Vol. 21, No. 1, Jan. 2020.

[Chry 14] A. Chrysopoulos, C. Diou, A. Symeonidis, and P. Mitkas. "Bottom-up modeling of small-scale energy consumers for effective Demand Response Applications". *Engineering Applications of Artificial Intelligence*, Vol. 35, pp. 299–315, Oct. 2014.

[Chua 15] V. V. Y. Chuan Choong YANG, Chit Siang SOH. "A systematic approach to ON-OFF event detection and clustering analysis of non-intrusive appliance load monitoring". *Frontiers in Energy*, Vol. 9, No. 2, p. 231, 2015.

[Cook 07] N. R. Cook. "Use and Misuse of the Receiver Operating Characteristic Curve in Risk Prediction". *Circulation*, Vol. 115, No. 7, pp. 928–935, 2007.

[Cort 95] C. Cortes and V. Vapnik. "Support-vector networks". Sep 1995.

[Cree 90] J. Creedy. *Foundations of Economic Thought*. Blackwell Publishers, 1990.

[Curt 08] G. Curtis and D. Cobham. *Business information systems: Analysis, design and practice*. Pearson Education, 2008.

[Dau 18] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. "The UCR Time Series Classification Archive". October 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[Daub 88] I. Daubechies. "Orthonormal bases of compactly supported wavelets". *Communications on Pure and Applied Mathematics*, Vol. 41, No. 7, pp. 909–996, Oct. 1988.

[De B 18a] L. De Baets, T. Dhaene, D. Deschrijver, C. Develder, and M. Berges. "VI-Based Appliance Classification Using Aggregated Power Consumption Data". In: *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 179–186, 2018.

[De B 18b] L. De Baets, J. Ruyssinck, C. Develder, T. Dhaene, and D. Deschrijver. "Appliance classification using VI trajectories and convolutional neural networks". *Energy and Buildings*, Vol. 158, pp. 32–36, 2018.

[De B 20] L. De Baets, M. Berges, T. Dhaene, C. Develder, J. Gao, and D. Deschrijver. "PLAID 2017". Jan. 2020.

[DEDD 21] "DEDDIAG, a domestic electricity demand dataset of individual appliances in Germany". Jan. 2021.

[Dekk 05] F. Dekking, C. Kraaikamp, H. Lopuhaä, and L. Meester. *A Modern Introduction to Probability and Statistics: Understanding Why and How*. *Springer Texts in Statistics*, Springer, 2005.

[Dolu 20] G. Doluweera, F. Hahn, J. Bergerson, and M. Pruckner. "A scenario-based study on the impacts of electric vehicles on energy consumption and sustainability in Alberta". *Applied Energy*, Vol. 268, p. 114961, 2020.

[Du 16] L. Du, D. He, R. G. Harley, and T. G. Habetler. "Electric Load Classification by Binary Voltage–Current Trajectory Mapping". *IEEE Transactions on Smart Grid*, Vol. 7, No. 1, pp. 358–365, 2016.

[Duda 00] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2 Ed., 2000.

[Eid 16] C. Eid, E. Koliou, M. Valles, J. Reneses, and R. Hakvoort. “Time-based pricing and electricity demand response: Existing barriers and next steps”. *Utilities Policy*, Vol. 40, pp. 15–25, 2016.

[EU D 19] “EU Directive 2009/72/EG”. `https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=celex%3A32009L0072`, 2019.

[Ever 10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. “The Pascal Visual Object Classes (VOC) Challenge”. *International Journal of Computer Vision*, Vol. 88, No. 2, pp. 303–338, Jun 2010.

[Faou 20] J. Faouzi and H. Janati. “pyts: A Python Package for Time Series Classification”. *Journal of Machine Learning Research*, Vol. 21, No. 46, pp. 1–6, 2020.

[Farh 10] H. Farhangi. “The path of the smart grid”. *IEEE Power and Energy Magazine*, Vol. 8, No. 1, pp. 18–28, 2010.

[Fari 99] L. Farinaccio and R. Zmeureanu. “Using a pattern recognition approach to disaggregate the total electricity consumption in a house into the major end-uses”. *Energy and Buildings*, Vol. 30, No. 3, pp. 245–259, 1999.

[Faus 20] A. Faustine and L. Pereira. “Improved Appliance Classification in Non-Intrusive Load Monitoring Using Weighted Recurrence Graph and Convolutional Neural Networks”. *Energies*, Vol. 13, No. 13, 2020.

[Faus 21] A. Faustine, L. Pereira, and C. Klemenjak. “Adaptive Weighted Recurrence Graphs for Appliance Recognition in Non-Intrusive Load Monitoring”. *IEEE Transactions on Smart Grid*, Vol. 12, No. 1, pp. 398–406, 2021.

[Feng 13] C. Feng, H. Hoe, M. P. Abdullah, M. Y. Hassan, and F. Hussin. “Tracing of energy consumption by using harmonic current”. *Proceeding - IEEE Student Conference on Research and Development, SCOReD*, pp. 444–449, Jan. 2013.

[Figu 11] M. B. Figueiredo, A. de Almeida, and B. Ribeiro. “An Experimental Study on Electrical Signature Identification of Non-Intrusive Load Monitoring (NILM) Systems”. In: *Adaptive and Natural Computing Algorithms*, pp. 31–40, Springer, Jan. 2011.

[Fisc 13] J. Fischer, S. Ramchurn, M. Osborne, O. Parson, T. D. Huynh, M. Alam, N. Pantidi, S. Moran, K. Bachour, S. Reece, E. Costanza, T. Rodden, and N. Jennings. “Recommending Energy Tariffs and Load Shifting Based on Smart Household Usage Profiling”. pp. 383–394, March 2013.

[Fitt 10] M. Fitta, S. Biza, M. Lehtonen, T. Nieminen, and G. Jacucci. “Exploring Techniques for Monitoring Electric Power Consumption in Households”. *UBICOMM 2010 - 4th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pp. 471–477, Jan. 2010.

[Fix 51] E. Fix and J. L. Hodges. “Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties”. Tech. Rep., USAF School of Aviation Medicine, Randolph Field, Texas., Feb. 1951.

[Flee 08] P. J. V. Fleet. *Discrete Wavelet Transformations.* John Wiley & Sons, Inc., Jan. 2008.

[Foru 12] Forum Netztechnik / Netzbertrieb im VDE. “Technische Anforderungen an die automatische Frequenzentlastung”. 2012. https://www.regelleistung.net/ext/download/ablaAnforderungenFnn.

[Froe 11] J. Froehlich, E. Larson, S. Gupta, G. Cohn, M. Reynolds, and S. Patel. “Disaggregated End-Use Energy Sensing for the Smart Grid”. *Pervasive Computing, IEEE*, Vol. 10, pp. 28–39, Apr. 2011.

[Gao 14] J. Gao, S. Giri, E. C. Kara, and M. Bergés. “PLAID: A Public Dataset of High-Resoultion Electrical Appliance Measurements for Load Identification Research: Demo Abstract”. In: *Proc. of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, p. 198–199, Association for Computing Machinery, New York, NY, USA, 2014.

[Gao 15] J. Gao, E. C. Kara, S. Giri, and M. Bergés. “A feasibility study of automated plug-load identification from high-frequency measurements”. In: *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 220–224, 2015.

[Gell 85] C. W. Gellings. “The concept of demand-side management for electric utilities”. *Proceedings of the IEEE*, Vol. 73, No. 10, pp. 1468–1470, 1985.

[Gell 96] C. W. Gellings. “Then and now”. *Energy Policy*, Vol. 24, No. 4, pp. 285–288, Apr. 1996.

[Germ 20] German Federal Environment Agency. “Anteile der Anwendungsbereiche am Netto-Stromverbrauch der privaten Haushalte 2008 und 2018”. https://www.umweltbundesamt.de/sites/default/files/medien/384/bilder/dateien/5_abb_anteile-anwendungsbereiche-am-netto-sv-ph_2020-07-01.xlsx, 2020.

[Germ 22] T. F. G. of Germany. “Relief for electricity consumers”. 2022. https://www.bundesregierung.de/breg-en/news/renewable-energy-sources-act-levy-abolished-2011854.

[Gils 14] H. C. Gils. “Assessment of the theoretical demand response potential in Europe”. *Energy*, Vol. 67, pp. 1–18, Apr. 2014.

[Giri 13] S. Giri, P.-H. Lai, and M. Bergés. “Novel Techniques for the Detection of ON and OFF States of Appliances for Power Estimation in Non-Intrusive Load Monitoring”. In: F. Hassani, O. Moselhi, and C. Haas, Eds., *Proceedings of the 30th International Symposium on Automation and Robotics in Construction and Mining (ISARC 2013): Building the Future in Automation and Robotics*, pp. 522–530, International Association for Automation and Robotics in Construction (IAARC), Montreal, Canada, Aug. 2013.

[Girm 16] A. A. Girmay and C. Camarda. “Simple Event Detection and Disaggregation Approach for Residential Energy Estimation”. In: *3rd International Workshop on Non-Intrusive Load Monitoring*, 2016.

[Gisl 13] C. Gisler, A. Ridi, D. Zufferey, O. A. Khaled, and J. Hennebert. “Appliance consumption signature database and recognition test protocols”. In: *2013 8th International Workshop on Systems, Signal Processing and their Applications (WoSSPA)*, pp. 336–341, 2013.

[glob 23] globalpetrolprices.com. "Electricity Prices Worldwide - June 2023". 2023. https://www.globalpetrolprices.com/Germany/electricity_prices/.

[Goel 18] T. Goeller, M. Wenninger, and J. Schmidt. "Towards Cost-Effective Utility Business Models - Selecting a Communication Architecture for the Rollout of New Smart Energy Services". In: *Proceedings of the 7th International Conference on Smart Cities and Green ICT Systems - Volume 1: SMARTGREENS,*, pp. 231–237, INSTICC, SciTePress, 2018.

[Good 16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[Gros 84] A. Grossmann and J. Morlet. "Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape". *SIAM Journal on Mathematical Analysis*, Vol. 15, pp. 723–736, July 1984.

[Gutw 09] T. Gutwin. "Smart Grid at BChydro: Current Status". 2009. https://grouper.ieee.org/groups/td/dist/da/doc/2009-07%20BC%20Hydro%20%20SmartGRID-Status.pdf.

[Hail 96] Haili Ma and A. A. Girgis. "Identification and tracking of harmonic sources in a power system using a Kalman filter". *IEEE Transactions on Power Delivery*, Vol. 11, No. 3, pp. 1659–1665, 1996.

[Hand 09] D. J. Hand. "Measuring classifier performance: a coherent alternative to the area under the ROC curve". *Machine Learning*, Vol. 77, No. 1, pp. 103–123, 2009.

[Hand 18] D. Hand and P. Christen. "A note on using the F-measure for evaluating record linkage algorithms". *Statistics and Computing*, Vol. 28, No. 3, pp. 539–547, 2018.

[Hart 85] G. W. Hart. "Prototype nonintrusive appliance load monitor". Tech. Rep., MIT Energy Lab. and Electric Power Research Institute, 1985.

[Hart 92] G. W. Hart. "Nonintrusive appliance load monitoring". *Proceedings of the IEEE*, Vol. 80, No. 12, pp. 1870–1891, 1992.

[He 14] K. He, X. Zhang, S. Ren, and J. Sun. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". *Lecture Notes in Computer Science*, p. 346–361, 2014.

[Heie 03] E. O. Heierman, III and D. J. Cook. "Improving Home Automation by Discovering Regularly Occurring Device Usage Patterns". In: *Proc. of the Third IEEE Intern. Conf. on Data Mining*, pp. 537–540, IEEE Computer Society, Washington, DC, USA, 2003.

[Henn 98] J. Hennebert. *Hidden Markov models and artificial neural networks for speech and speaker recognition*. PhD thesis, Jan. 1998.

[Hoch 97] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". *Neural Comput.*, Vol. 9, No. 8, p. 1735–1780, nov 1997.

[Holu 13] O. Holub and M. Sikora. "End user models for residential demand response". In: *IEEE PES ISGT Europe 2013*, pp. 1–4, 2013.

[Hube 18] P. Huber, M. Gerber, A. Rumsch, and A. Paice. "Prediction of domestic appliances usage based on electrical consumption". *Energy Informatics*, Vol. 1, No. S1, Oct. 2018.

[IEA 19] IEA – International Energy Agency. "World Energy Outlook 2019". 2019.

[IEA 20] IEA – International Energy Agency. "Germany 2020 - Energy Policy Review". 2020. https://www.bmwi.de/Redaktion/DE/Downloads/G/germany-2020-energy-policy-review.pdf?__blob=publicationFile&v=4.

[Jana 13] K. Janani and S. Himavathi. "Non-intrusive harmonic source identification using neural networks". In: *Proc. of Int. Conf. on Computation of Power, Energy, Inform. and Commun., ICCPEIC*, pp. 59–64, Apr. 2013.

[Jian 12] L. Jiang, S. Luo, and J. Li. "An Approach of Household Power Appliance Monitoring Based on Machine Learning". In: *5th International Conference on Intelligent Computation Technology and Automation*, pp. 577–580, 2012.

[Jian 13] L. Jiang, S. Luo, and J. Li. "Automatic power load event detection and appliance classification based on power harmonic features in nonintrusive appliance load monitoring". pp. 1083–1088, June 2013.

[John 13] M. J. Johnson and A. S. Willsky. "Bayesian Nonparametric Hidden Semi-Markov Models". *J. Mach. Learn. Res.*, Vol. 14, No. 1, p. 673–701, Feb. 2013.

[John 14] B. J. Johnson, M. R. Starke, O. A. Abdelaziz, R. K. Jackson, and L. M. Tolbert. "A MATLAB based occupant driven dynamic model for predicting residential power demand". In: *2014 IEEE PES T D Conference and Exposition*, pp. 1–5, 2014.

[Kahl 16] M. Kahl, A. Haq, T. Kriechbaumer, and H. Jacobsen. "WHITED-A Worldwide Household and Industry Transient Energy Data Set". 2016.

[Kahl 19] M. Kahl, T. Kriechbaumer, D. Jorde, A. Ul Haq, and H.-A. Jacobsen. "Appliance Event Detection - A Multivariate, Supervised Classification Approach". In: *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, p. 373–375, Association for Computing Machinery, New York, NY, USA, 2019.

[Kang 14] Z. Kang, M. Jin, and C. J. Spanos. "Modeling of end-use energy profile: An appliance-data-driven stochastic approach". In: *The 40th Annual Conf. of the IEEE Industrial Electronics Society, Dallas, TX, USA*, pp. 5382–5388, 2014.

[Kato 09] T. Kato, H. S. Cho, D. Lee, T. Toyomura, and T. Yamazaki. "Appliance Recognition from Electric Current Signals for Information-Energy Integrated Network in Home Environments". In: M. Mokhtari, I. Khalil, J. Bauchet, D. Zhang, and C. Nugent, Eds., *Ambient Assistive Health and Wellness Management in the Heart of the City*, pp. 150–157, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[Kell 14] J. Kelly, N. Batra, O. Parson, H. Dutta, W. Knottenbelt, A. Rogers, A. Singh, and M. Srivastava. "NILMTK v0.2: a non-intrusive load monitoring toolkit for large scale data sets". In: *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, ACM, Nov. 2014.

[Kell 15] J. Kelly and W. Knottenbelt. “The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes”. *Scientific Data*, Vol. 2, No. 1, March 2015.

[Keog 01] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. “Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases”. *Knowledge and Information Systems*, Vol. 3, No. 3, pp. 263–286, Aug. 2001.

[Kim 11] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han. “Unsupervised Disaggregation of Low Frequency Power Measurements”. In: *Proceedings of the 2011 SIAM International Conference on Data Mining*, Society for Industrial and Applied Mathematics, Apr. 2011.

[Kim 19] J.-G. Kim and B. Lee. “Appliance Classification by Power Signal Analysis Based on Multi-Feature Combination Multi-Layer LSTM”. *Energies*, Vol. 12, No. 14, p. 2804, July 2019.

[Klem 19] C. Klemenjak, A. Reinhardt, L. Pereira, S. Makonin, M. Bergés, and W. Elmenreich. “Electricity Consumption Data Sets: Pitfalls and Opportunities”. In: *Proc. of the 6th ACM Int. Conf. on Systems for Energy-Efficient Buildings, Cities, and Transportation*, p. 159–162, Association for Computing Machinery, New York, NY, USA, 2019.

[Kolt 11] Z. Kolter and M. J. Johnson. “REDD: A public data set for energy disaggregation research”. In: *Proc. of SustKDD*, 2011.

[Krie 17] T. Kriechbaumer and H.-A. Jacobsen. “BLOND: Building-Level Office eNvironment Dataset”. 2017.

[Lach 14] D. Lachut, N. Banerjee, and S. Rollins. “Predictability of energy use in homes”. In: *International Green Computing Conference*, pp. 1–10, Nov. 2014.

[Lai 13] Y.-X. Lai, C.-F. Lai, Y.-M. Huang, and H.-C. Chao. “Multi-Appliance Recognition System with Hybrid SVM/GMM Classifier in Ubiquitous Smart Home”. *Inf. Sci.*, Vol. 230, p. 39–55, May 2013.

[Lam 07] H. Y. Lam, G. S. K. Fung, and W. K. Lee. “A Novel Method to Construct Taxonomy Electrical Appliances Based on Load Signatures”. *IEEE Transactions on Consumer Electronics*, Vol. 53, No. 2, pp. 653–660, 2007.

[LeCu 89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. “Backpropagation Applied to Handwritten Zip Code Recognition”. *Neural Computation*, Vol. 1, No. 4, pp. 541–551, 1989.

[Lecu 98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278–2324, Nov. 1998.

[Lee 10] S.-C. Lee, G.-Y. Lin, W.-R. Jih, and J. Y.-J. Hsu. “Appliance Recognition and Unattended Appliance Detection for Energy Conservation”. In: *Proceedings of the 5th AAAI Conference on Plan, Activity, and Intent Recognition*, p. 37–44, 2010.

[Lee 13] S. Lee, G. Ryu, Y. Chon, R. Ha, and H. Cha. “Automatic Standby Power Management Using Usage Profiling and Prediction.”. *IEEE Trans. Hum. Mach. Syst.*, Vol. 43, No. 6, pp. 535–546, 2013.

[Mako 15] S. Makonin and F. Popowich. “Nonintrusive load monitoring (NILM) performance evaluation”. *Energy Efficiency*, Vol. 8, No. 4, pp. 809–814, 2015.

[Mako 16] S. Makonin, B. Ellert, I. V. Bajic, and F. Popowich. “Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014”. *Scientific Data*, Vol. 3, No. 160037, pp. 1–12, 2016.

[Mako 17] S. Makonin, Z. J. Wang, and C. Tumpach. “RAE: The Rainforest Automation Energy Dataset for Smart Grid Meter Data Analysis”. *arXiv:1705.05767*, May 2017.

[Marc 11] M. Marcu and A. Stancovici. “Systems classification based on power signatures”. In: *2011 2nd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies*, Dec. 2011.

[Marw 07] N. Marwan, M. Carmenromano, M. Thiel, and J. Kurths. “Recurrence plots for the analysis of complex systems”. *Physics Reports*, Vol. 438, No. 5-6, pp. 237–329, jan 2007.

[McCu 43] W. S. McCulloch and W. Pitts. “A logical calculus of the ideas immanent in nervous activity”. *The bulletin of mathematical biophysics*, Vol. 5, No. 4, pp. 115–133, 1943.

[Meeh 12] P. Meehan, S. Phelan, C. McArdle, and S. Daniels. “Temporal and frequency analysis of power signatures for common household appliances”. In: *Symposium on ICT and Energy Efficiency and Workshop on Information Theory and Security*, pp. 22–27, Institution of Engineering and Technology, July 2012.

[Meie 06] A. von Meier. *Electric power systems: a conceptual introduction*. Wiley-IEEE Press, 2006.

[MITN 20] MITNETZ STROM Mitteldeutsche Netzgesellschaft Strom mbH. “Berichte zu durchgeführten Entlastungsmaßnahmen der MITNETZ STROM 2019”. =https://nsm-portal.mitnetz-strom.de/MNS/NSM/#/teilnahme/public, 2020.

[Mona 14] A. Monacchi, D. Egarter, W. Elmenreich, S. D’Alessandro, and A. M. Tonello. “GREEND: An energy consumption dataset of households in Italy and Austria”. In: *IEEE Int. Conf. SmartGridComm*, pp. 511–516, Nov. 2014.

[Murr 15] D. Murray, J. Liao, L. Stankovic, V. Stankovic, R. Hauxwell-Baldwin, C. Wilson, M. Coleman, T. Kane, and S. Firth. “A data management platform for personalised real-time energy feedback”. In: *Proc. of the 8th Int. Conf. on Energy Efficiency in Domestic Appliances and Lighting*, pp. 1–15, Aug. 2015.

[Niem 03] H. Niemann. *Klassifikation von Mustern*. Springer Verlag, 2003.

[ONei 10] D. O’Neill, M. Levorato, A. Goldsmith, and U. Mitra. “Residential Demand Response Using Reinforcement Learning”. In: *1st IEEE International Conference on Smart Grid Communications*, pp. 409–414, 2010.

[Ozak 18] R. Ozaki. “Follow the price signal: People’s willingness to shift household practices in a dynamic time-of-use tariff trial in the United Kingdom”. *Energy Research & Social Science*, Vol. 46, pp. 10–18, Dec. 2018.

[Parz 62] E. Parzen. “On Estimation of a Probability Density Function and Mode”. *The Annals of Mathematical Statistics*, Vol. 33, No. 3, pp. 1065–1076, 1962.

[Pate 07] S. Patel, T. Robertson, J. Kientz, M. Reynolds, and G. Abowd. “At the flick of a switch: detecting and classifying unique electrical events on the residential power line”. pp. 271–288, Sep. 2007.

[Patt 12] S. Pattem. “Unsupervised Disaggregation for Non-intrusive Load Monitoring”. In: *2012 11th International Conference on Machine Learning and Applications*, pp. 515–520, 2012.

[Peca 14] Pecan Street Inc. “Dataport”. https://dataport.pecanstreet.org/, 2014.

[Pere 15] L. Pereira and N. J. Nunes. “Semi-automatic labeling for public non-intrusive load monitoring datasets”. In: *2015 Sustainable Internet and ICT for Sustainability (SustainIT)*, pp. 1–4, 2015.

[Pere 19] L. Pereira. “NILMPEds: A Performance Evaluation Dataset for Event Detection Algorithms in Non-Intrusive Load Monitoring”. *Data*, Vol. 4, No. 3, p. 127, Aug. 2019.

[Pico 16] T. Picon, M. N. Meziane, P. Ravier, G. Lamarque, C. Novello, J.-C. L. Bunetel, and Y. Raingeaud. “COOLL: Controlled On/Off Loads Library, a Public Dataset of High-Sampled Electrical Signals for Appliance Identification”. 2016.

[Powe 11a] D. M. W. Powers. “Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation”. *Journal of Machine Learning Technologies*, Vol. 2, No. 1, pp. 37–63, 2011.

[Powe 11b] D. M. W. Powers. “Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation”. *Journal of Machine Learning Technologies*, Vol. 2, No. 1, pp. 37–63, 2011.

[Proe 21] E. Proedrou. “A Comprehensive Review of Residential Electricity Load Profile Models”. *IEEE Access*, Vol. 9, No. , pp. 12114–12133, 2021.

[Pull 21] M. Pullinger, J. Kilgour, N. Goddard, N. Berliner, L. Webb, M. Dzikovska, H. Lovell, J. Mann, C. Sutton, J. Webb, and M. Zhong. “The IDEAL household energy dataset, electricity, gas, contextual sensor data and survey data for 255 UK homes”. Vol. 8, No. 146, May 2021.

[Qayy 15] F. Qayyum, N. Muhammad, A. Khwaja, L. Guan, and B. Venkatesh. “Appliance Scheduling Optimization in Smart Home Networks”. *IEEE Access*, Vol. 3, pp. 2176–2190, Jan. 2015.

[Quan 05] Quantum Consulting Inc. and Summit Blue Consulting, LLC. “Demand response program evaluation - Final report”. Working Group 2 Measurement and Evaluation Committee, California Edison Company, 2005.

[Rein 12] A. Reinhardt, P. Baumann, D. Burgstahler, M. Hollick, H. Chonov, M. Werner, and R. Steinmetz. “On the accuracy of appliance identification based on distributed load metering data”. pp. 1–9, Jan. 2012.

[Ridi 13] A. Ridi, C. Gisler, and J. Hennebert. “Automatic identification of electrical appliances using smart plugs”. In: *8th International Workshop on Systems, Signal Processing and their Applications (WoSSPA)*, pp. 301–305, 2013.

[Ridi 14] A. Ridi, C. Gisler, and J. Hennebert. “A Survey on Intrusive Load Monitoring for Appliance Recognition”. In: *2014 22nd International Conference on Pattern Recognition*, pp. 3702–3707, 2014.

[Roll 14] S. Rollins, N. Banerjee, L. Choudhury, and D. Lachut. “A system for collecting activity annotations for home energy management”. *Pervasive and Mobile Computing*, Vol. 15, pp. 153–165, Dec. 2014.

[Rowl 07] J. Rowley. “The wisdom hierarchy: representations of the DIKW hierarchy”. *Journal of Information Science*, Vol. 33, No. 2, pp. 163–180, 2007.

[Rume 86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. *Nature*, Vol. 323, No. 6088, pp. 533–536, Oct. 1986.

[Sait 10] T. Saitoh, T. Osaki, R. Konishi, and K. Sugahara. “Current Sensor Based Home Appliance and State of Appliance Recognition”. *SICE Journal of Control, Measurement, and System Integration*, Vol. 3, pp. 86–93, Apr. 2010.

[Scha 17] A. P. Schaffarczyk. *Einführung in die Windenergietechnik*. Carl Hanser Verlag GmbH Co KG, 2 Ed., 2017.

[Scho 10] A. Schoofs, A. Guerrieri, D. Delaney, G. O’Hare, and A. Ruzzelli. “ANNOT: Automated Electricity Data Annotation Using Wireless Sensor Networks”. pp. 1–9, July 2010.

[Schw 09] A. Schwab. *Elektroenergiesysteme: Erzeugung, Transport, Übertragung und Verteilung elektrischer Energie*. Springer, Dordrecht New York, 2009.

[SMAR 24] SMARD. “Net public electricity generation in Germany in 2023”. `https://www.smard.de/page/home/topic-article/444/211756`, 2024. Last access: 29.02.2024.

[Srin 06] D. Srinivasan, W. Ng, and A. Liew. “Neural-Network-Based Signature Recognition for Harmonic Source Identification”. *IEEE Transactions on Power Delivery*, Vol. 21, No. 1, pp. 398–405, Jan. 2006.

[Stec 20] D. Stecher. “Machine Learning: Event Detection in Time Series Using Support Vector Machines”. 2020.

[Step 14] B. Stephen, S. Galloway, and G. Burt. “Self-Learning Load Characteristic Models for Smart Appliances”. *IEEE Transactions on Smart Grid*, Vol. 5, No. 5, pp. 2432–2439, Sep. 2014.

[Stro 22] L. Strobel, J. Schlund, and M. Pruckner. “Joint analysis of regional and national power system impacts of electric vehicles—A case study for Germany on the county level in 2030”. *Applied Energy*, Vol. 315, p. 118945, 2022.

[Swan 09] L. G. Swan and V. I. Ugursal. “Modeling of end-use energy consumption in the residential sector: A review of modeling techniques”. *Renewable and Sustainable Energy Reviews*, Vol. 13, No. 8, pp. 1819–1835, 2009.

[Tasc 18] A. Tascikaraoglu. “On Data-Driven Approaches for Demand Response”. In: R. Arghandeh and Y. Zhou, Eds., *Big Data Application in Power Systems*, pp. 243–259, Elsevier, 2018.

[Truo 13a] N. C. Truong, J. McInerney, L. Tran-Thanh, E. Costanza, and S. D. Ramchurn. “Forecasting Multi-appliance Usage for Smart Home Energy Management”. In: *Proc. of the Twenty-Third Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 2908–2914, AAAI, 2013.

[Truo 13b] N. C. Truong, L. Tran-Thanh, E. Costanza, and S. D. Ramchurn. “Towards Appliance Usage Prediction for Home Energy Management”. In: *Proc. of the Fourth Intern. Conf. on Future Energy Systems*, pp. 287–288, ACM, New York, NY, USA, 2013.

[US D 06] U.S. Department of Energy. “Benefits of Demand Response in Electricity Markets and Recommendations for Achieving Them: A Report to the United States Congress Pursuant to Section 1252 of The Energy Policy Act of 2005”. pp. 11–12, 2006.

[Utta 15] A. S. Uttama Nambi, A. Reyes Lua, and V. R. Prasad. “LocED: Location-aware Energy Disaggregation Framework”. In: *Proc. of the 2nd ACM Int. Conf. on Embedded Systems for Energy-Efficient Built Environments*, pp. 45–54, New York, USA, 2015.

[Valv 14] F. J. Valverde-Albacete and C. Peláez-Moreno. “100% Classification Accuracy Considered Harmful: The Normalized Information Transfer Factor Explains the Accuracy Paradox”. *PLOS ONE*, Vol. 9, No. 1, pp. 1–10, Jan. 2014.

[Weis 12] M. Weiss, A. Helfenstein, F. Mattern, and T. Staake. “Leveraging smart meter data to recognize home appliances”. In: *2012 IEEE International Conference on Pervasive Computing and Communications*, pp. 190–197, 2012.

[Wenn 17] M. Wenninger, J. Schmidt, and T. Goeller. “Appliance Usage Prediction for the Smart Home with an Application to Energy Demand Side Management - And Why Accuracy is not a Good Performance Metric for this Problem”. In: *Proceedings of the 6th International Conference on Smart Cities and Green ICT Systems - Volume 1: SMARTGREENS,*, pp. 143–150, INSTICC, SciTePress, 2017.

[Wenn 19a] M. Wenninger, S. P. Bayerl, J. Schmidt, and K. Riedhammer. “Timage – A Robust Time Series Classification Pipeline”. In: *Artificial Neural Networks and Machine Learning - ICANN 2019: Text and Time Series*, pp. 450–461, Springer International Publishing, Cham, 2019.

[Wenn 19b] M. Wenninger, D. Stecher, and J. Schmidt. “SVM-Based Segmentation of Home Appliance Energy Measurements”. In: *18th IEEE International Conference on Machine Learning and Applications - ICMLA 2019*, pp. 1666–1670, 2019.

[Wenn 21a] M. Wenninger, S. P. Bayerl, A. Maier, and J. Schmidt. “Recurrence Plot Spacial Pyramid Pooling Network for Appliance Identification in Non-Intrusive Load Monitoring”. In: *20th IEEE International Conference on Machine Learning and Applications - ICMLA 2021*, 2021.

[Wenn 21b] M. Wenninger, A. Maier, and J. Schmidt. “DEDDIAG, a domestic electricity demand dataset of individual appliances in Germany”. *Scientific Data*, Vol. 8, No. 176, July 2021.

[Widr 90] B. Widrow and M. Lehr. “30 years of adaptive neural networks: perceptron, Madaline, and backpropagation”. *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1415–1442, 1990.

[Wild 15] B. Wild, K. S. Barsim, and B. Yang. “A new unsupervised event detector for non-intrusive load monitoring”. In: *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 73–77, 2015.

[Zaid 10] A. Zaidi, F. Kupzog, T. Zia, and P. Palensky. “Load recognition for automated demand response in microgrids”. pp. 2442–2447, Dec. 2010.

[Zhan 14] K. Zhang, L. Xu, M. Ouyang, H. Wang, L. Lu, J. Li, and Z. Li. “Optimal decentralized valley-filling charging strategy for electric vehicles”. *Energy Conversion and Management*, Vol. 78, pp. 537–550, 2014.

[Zhan 15] D. Zhang, J. Wang, and X. Zhao. “Estimating the Uncertainty of Average F1 Scores”. In: *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, p. 317–320, Association for Computing Machinery, New York, NY, USA, 2015.

[Zhon 16] J. Zhong, J. Wang, X. Yu, Q. Wang, M. Combariza, and G. Holmes. “Hybrid Load Profile Clustering for identifying patterns of electricity consumers”. pp. 708–713, June 2016.

[Zhu 07] X. Zhu and I. Davidson. *Knowledge Discovery and Data Mining: Challenges and Realities.* Information Science Reference, Hershey, New York, 2007.