ULB UNIVERSITÉ LIBRE DE BRUXELLES

DEMANDFLEX

Deliverable 4.1a:

An Overview of Non-Intrusive Load Monitoring

Jean-Luc Timmermans
Pierre Henneaux

Brussels, May 2025

Contents

1 Introduction **4**
- 1.1 Definition . . . 4
- 1.2 Motivations . . . 4
- 1.3 Objectives . . . 5

2 Core principles and methodologies **6**
- 2.1 Problem description . . . 6
- 2.2 Load categorization . . . 7
- 2.3 Taxonomy of NILM Methods . . . 11
  - 2.3.1 Event-based and event-less methods . . . 11
  - 2.3.2 Supervised and unsupervised methods . . . 12
  - 2.3.3 Classification and regression methods . . . 13
- 2.4 Pre-processing . . . 13

3 Algorithms review **15**
- 3.1 Hart algorithm . . . 15
- 3.2 Optimization algorithms . . . 15
- 3.3 Hidden Markov Models . . . 17
- 3.4 Machine Learning . . . 24
  - 3.4.1 K-means clustering . . . 24
  - 3.4.2 k-nearest neighbors . . . 25
  - 3.4.3 Support Vector Machine . . . 25
  - 3.4.4 Decision Tree . . . 26
  - 3.4.5 Random Forest . . . 27
  - 3.4.6 Naive Bayesian Classifier . . . 28
  - 3.4.7 Ensemble model . . . 29
- 3.5 Deep learning . . . 29
  - 3.5.1 Multilayer Perceptron . . . 30
  - 3.5.2 Recurrent neural network . . . 32
  - 3.5.3 Convolutional neural network . . . 34
  - 3.5.4 Autoencoder . . . 37
  - 3.5.5 Generative adversarial network . . . 39
  - 3.5.6 Domain Adversarial Neural Networks . . . 41
  - 3.5.7 Transformers and attention mechanism . . . 43
  - 3.5.8 Transfer Learning . . . 44
- 3.6 VI-trajectory . . . 44
- 3.7 Others . . . 46
  - 3.7.1 Dynamic Time warping . . . 46
  - 3.7.2 Principal Component Analysis . . . 49
  - 3.7.3 Sparse Dictionary Learning or sparse coding . . . 50
  - 3.7.4 Graph signal processing . . . 51
  - 3.7.5 Recurrence Graph . . . 52
  - 3.7.6 Fourier and Wavelet Transform . . . 53
- 3.8 Review conclusion . . . 55

4 Data source and metrics **57**
- 4.1 Public datasets . . . 57
- 4.2 Metrics . . . 59
  - 4.2.1 Regression metrics . . . 59
  - 4.2.2 Classification metrics . . . 60

5 Belgium case **61**

6 Algorithms implementation **64**
6.1 Motivations 64
6.2 Proposed method 64
6.3 Experiments 67
6.4 Results 68
6.5 Conclusion of the implementation 71

7 Conclusion **72**

1 Introduction

1.1 Definition

Non-Intrusive Load Monitoring (NILM), also called energy disaggregation, is a method to estimate the energy consumption of individual appliances within a building using a single point measurement, the smart meter or digital meter. This method is called non-intrusive because it only requires one measurement of the aggregated consumption at the base of the building. In comparison, Intrusive Load Monitoring (ILM) requires a measurement device for each load to monitor, which can be expensive, inconvenient, and laborious to maintain.

1.2 Motivations

Today, the electric power system must be adapted to integrate an increasing proportion of Renewable Energy Sources (RES) into the energy mix and to handle the evolution of electricity consumption. Traditional electrical grids are evolving into smart grids, which are advanced electricity networks capable of automatically monitoring and adjusting energy flows to optimize the production, transportation, and consumption of electrical energy. Enhancements at the residential level are essential for optimizing the power system as residential consumption represents 29.6 % of final energy consumption in Europe in 2021 [1]. In this context, deploying a successful NILM algorithm could have many benefits. The main ones are listed below.

One of the key benefits frequently highlighted in the literature is NILM's ability to provide residential consumers with detailed insights into their energy usage. Specifically, NILM can inform residential consumers about their energy consumption and the exact contribution of each appliance to it without the need for a sub-metering device for each appliance. In general, energy consumption feedback has been shown to positively influence consumer behavior, leading to reductions in both overall energy use and electricity bills. These reductions benefit not only consumers but also the environment. In [2], the authors review studies that evaluated the impact of energy feedback on residential consumption. The review shows that savings range typically between 5% and 20%. Unlike total consumption feedback, NILM provides appliance-specific feedback, which can enable more personalized recommendations, as suggested by [3], potentially leading to even greater reductions in energy use. However, there remains a lack of studies that fairly compare the impact of aggregated versus disaggregated feedback, as discussed in [4] and [5]. Additionally, more research is needed to clarify the long-term effects of energy feedback on the general population (not just energy enthusiasts) as discussed in [2]. With the recent rise in energy prices across Europe, growing public awareness of climate change, and the ease of transmitting energy data from smart meters to smartphones via apps, disaggregated feedback has the potential to lead to great energy savings. Moreover, the medium and presentation format of energy feedback can influence its effectiveness. For example, in [6], the authors propose feedback incorporating normative comparisons, showing users their energy consumption in relation to similar households, to enhance engagement. Following their systematic review, the authors in [5] recommend further studies on the impact of energy feedback delivered through smartphones and tablets to better integrate these practices into consumers' daily routines.

Energy disaggregation can also help develop residential flexibility. Residential flexibility is the adjustment of residential consumer consumption in response to an external signal (electricity price, operator request, etc.). This flexibility is more and more needed with the increasing proportion of renewable energy (RE) in the electricity mix. Indeed, as the production of RE assets cannot be controlled, the consumption must be adjusted to satisfy the adequacy requirement of equality of production and consumption. Residential flexibility is also necessary at a local level; with the electrification (e.g., increase of Electric Vehicle (EV) and Heat-Pumps (HP)) and the deployment of Distributed Energy Resources (DER) (like PV panels), the local electric grid cannot support the new peak in

power consumption or injection. These peaks in power injection lead, for example, to the well-known photovoltaic panel disconnections due to overvoltage. To avoid such consequences, these peaks must be shaved until a reinforcement of the grid. In this context, energy disaggregation can be useful in estimating the potential of the demand response. By decomposing the known total residential consumption into appliance consumption, the proportion of flexible loads such as EV charging, boiler, washing machine, dishwasher, and less flexible or non-flexible loads such as TV, computer, and cooking appliances can be estimated. Still, for the development of flexibility, energy disaggregation can be useful to analyse the impact of external signals like requests for consumption reduction or implicit incentives like Time-of-Use (ToU) tariffs[1].

NILM can also be useful for improving the consumer load profile, detecting faulty appliances, and monitoring elderly people.

1.3 Objectives

The objectives of this document are threefold. The first objective is to provide a clear and comprehensive overview of most of the existing NILM methods. By reading this document, one should have a general understanding of the energy disaggregation problem and its associated challenges. It is designed to be a helpful start for anyone interested in NILM, no matter the context. The second objective is to present disaggregation algorithms that appear to be the most adapted for residential energy disaggregation in Belgium. In particular, the algorithms must be adapted to the Belgian smart meter in deployment by the Distributed System Operators (DSOs) over all three regions of Belgium. The third objective is to identify the barriers to deploying NILM methods in Belgium and propose actions and future research orientation to tackle these barriers.

The remainder of this document is structured as follows. Section 2 explores the core principles and main categories of NILM methodologies, laying the foundation for understanding the various approaches to energy disaggregation. This section provides the theoretical introduction necessary for grasping the nuances of subsequent sections. Section 3 offers a comprehensive review of NILM methods found in the literature. This detailed review aims to give readers a thorough understanding of the current state-of-the-art in NILM research. Section 4 describes popular metrics and datasets from the literature. This section helps to understand how different NILM methods are assessed and compared, providing the tools to evaluate NILM implementations critically. Section 5 outlines the characteristics of the Belgian smart meter, explaining its specific features and limitations. Section 6 details our personal implementation of NILM deep learning models tailored for residential energy disaggregation in Belgium. It describes the technical approaches used and the results obtained. Finally, Section 7 presents the conclusions of this report. It includes our recommendations for overcoming the barriers to the deployment of NILM in Belgium and suggestions for future research directions.

[1] "Time-of-use network tariffs (or tariff time elements) mean charges for network service(s) that vary according to when the service is used (e.g. by peak/off-peak, season, month, weekdays/weekends, hour). They could take different forms depending on the basis used for charging: Energy-based: EUR/kWh in period t or Power-based: EUR/kW in period t. Time-of-use charges give signals to network users to use the network less in some periods in the day, week, or year and use it more in other periods. The charges should be higher in periods when network utilisation is closer to the technical limits and lower otherwise." from ACER's report on Electricity Transmission and Distribution Tariff Methodologies in Europe and Distribution Tariff Methodologies in Europe (2023) [7].

2 Core principles and methodologies

This section outlines the core principles and methodologies of NILM. It begins with a description of the energy disaggregation problem, followed by a discussion on load categorization. A taxonomy of NILM methods is then presented, covering event-based and event-less methods, supervised and unsupervised methods, and regression and classification approaches. The section concludes with an overview of the pre-processing steps necessary for preparing data for NILM algorithms.

2.1 Problem description

NILM can be described as the task of estimating the energy consumption of individual appliances within a household by analyzing an aggregated energy signal measured by a single sensor, typically a smart meter. The smart meter records the total active power consumption at regular intervals (time steps). The total energy consumption over a period is the product of power consumption and time[2]. At any given time t, the aggregate active power $P_{total}(t)$ recorded by the smart meter is the sum of the power consumed by each appliance $P_n(t)$, plus an error term $e(t)$, which accounts for measurement noise and line losses. Mathematically, this relationship is expressed in equation (1), with m the total number of appliances in the household.

$$P_{total}(t) = \sum_{n=1}^{m} P_n(t) + e(t) \tag{1}$$

Thus, the NILM problem can be framed as recovering the individual appliance power consumption $P_n(t)$ from the aggregate power $P_{total}(t)$ measured at each time step. Despite the simplicity of this formulation, energy disaggregation is a highly challenging problem. Even the most advanced algorithms fail to perfectly estimate the power of each appliance. To better understand how an algorithm estimates appliance consumption, it is crucial to first comprehend how appliances consume power and contribute to the aggregate signal. Figure 1 illustrates the decomposition of the aggregate power into the individual appliance power components.

[2]The relationship between power and energy is given by $E = P \times T$, where E is the energy in joules, P is the power in watts, and T is time in seconds. One watt-hour equals 3600 joules. For example, if a household has an average power consumption of 5 kW over the course of one hour, the total energy consumption would be 5 kWh.

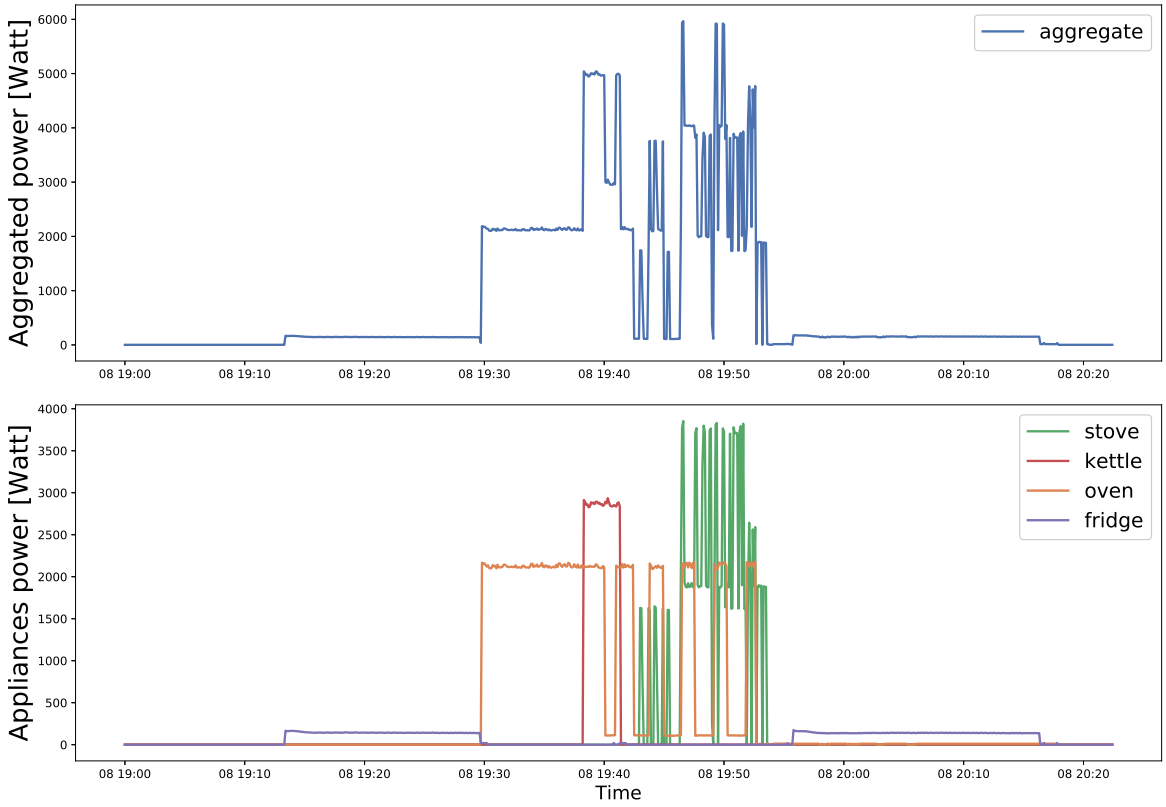

Figure 1: Illustration of the energy disaggregation concept.

2.2 Load categorization

In the literature, most authors split appliances into three categories according to their operating cycles. The operating cycle of an appliance is the variation of the power drawn by the appliance during its running time. Those categories consider the steady state power of the appliance, not the transient (i.e., the short period following the switching of an appliance) or the instantaneous small variation of power. The three categories are the following:

1. Two-state appliance or ON/OFF appliance. The two-state appliances are the appliances that can only be in two states: ON or OFF. When these appliances are ON, they draw a fixed amount of power. An example of a two-state appliance is a kettle.

2. Multi-State (MS) appliance. The multi-state appliances are the appliances that can be in multiple ON states. Each ON state draws a different amount of power. Examples of MS appliances are ovens, dishwashers, and washing machines.

3. Continuously Variable (CV) appliance. The continuously variable appliances are the appliances whose power consumption varies continuously when running. A Variable Speed Drive (VSD) is an example of a CV appliance. VSD is a device that can control the speed and torque of an electric motor. VSD can be used for elevators.

The three categories of appliances are illustrated in Figure 2.

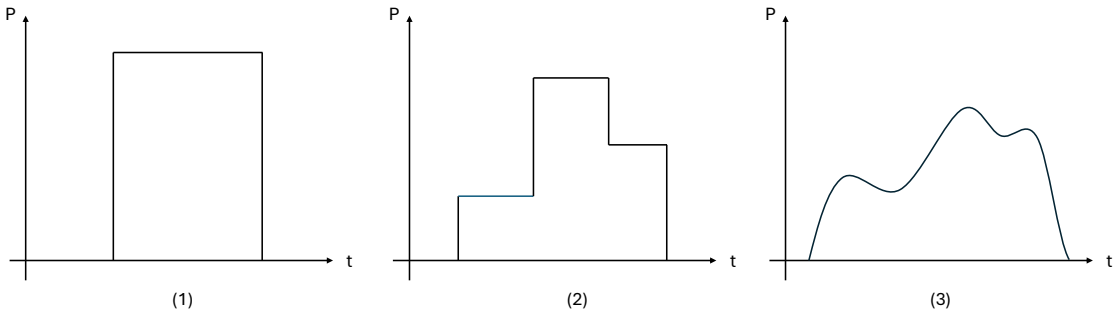

Figure 2: Load categorization with operating cycle. (1) two-state appliance, (2) multi-state appliance, (3) continuously variable appliance.

Figure 3 and Figure 4 display the actual measurement of two two-state appliances, a kettle and a fridge. In Figure 4, the power spikes observed just before each ON state of the fridge are clear examples of transients. As previously discussed, transients are typically not considered in load categorization. The small power spikes (below 20 watts) are disregarded due to their minimal impact on total energy consumption. If these spikes were more significant, the fridge could be classified as a multi-state appliance. Figure 5 displays the actual measurement of a multi-state appliance, an oven. Obviously, appliances do not consume perfectly constant power due to various physical factors related to both the grid and the appliance itself. For this reason, small instantaneous variations in power consumption are typically not accounted for in appliance categorization.

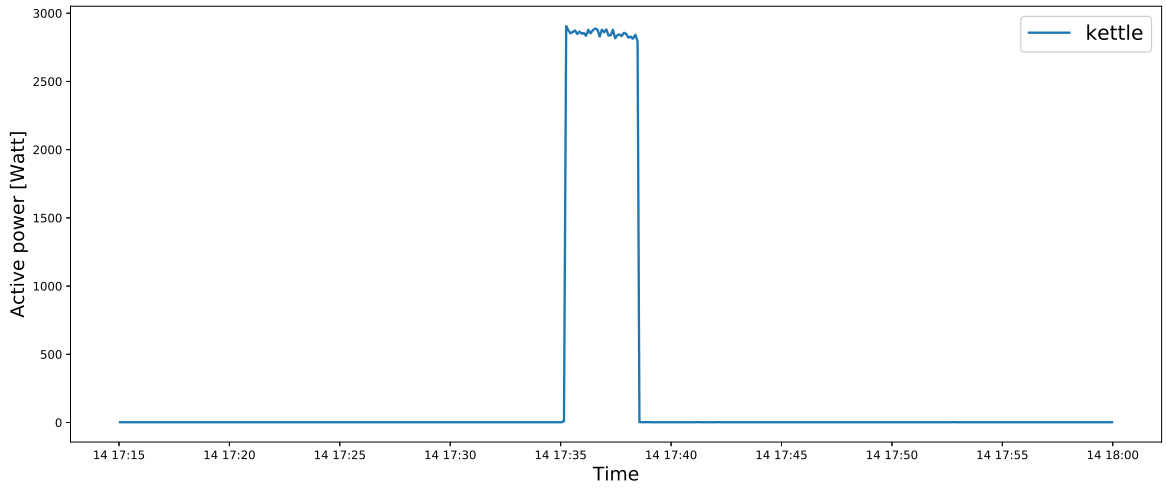

Figure 3: Measurement of a kettle operating cycle from UK-DALE dataset.

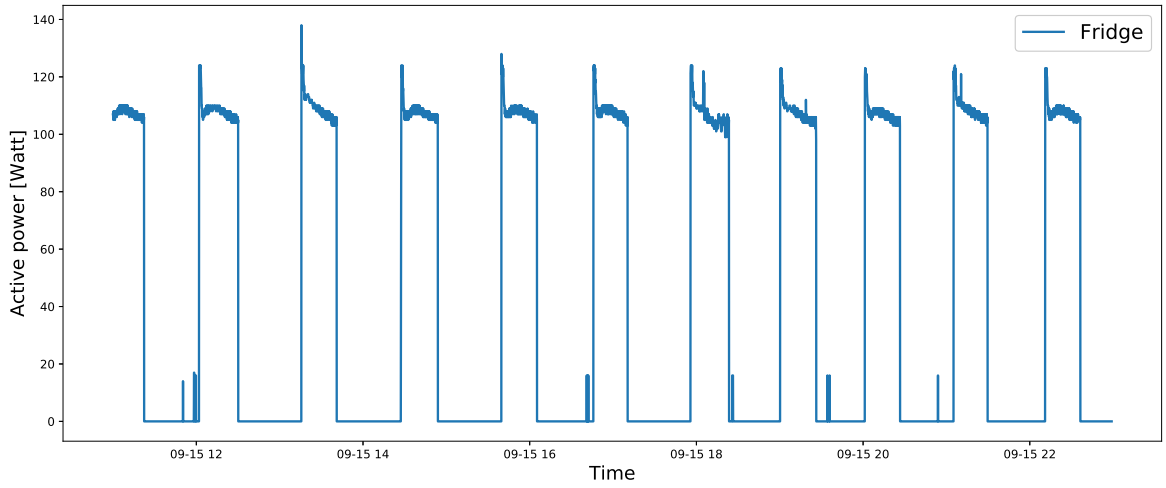

Figure 4: Measurement of a fridge operating cycle from UK-DALE dataset.

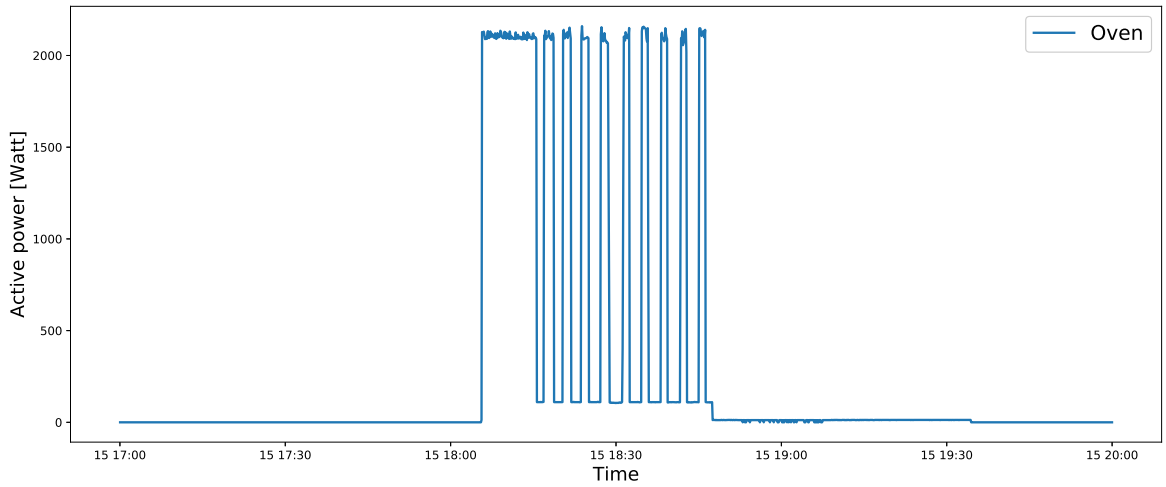

Figure 5: Measurement of an oven operating cycle from UK-DALE dataset.

From these examples of operating cycles, it is already evident that different appliances exhibit different characteristics. The power of the states, the duration of the states, the succession of the states, and the transient are all different. Thanks to these specific characteristics, NILM methods can detect and disaggregate individual appliances from the aggregate energy consumption. In NILM, the distinctive pattern and characteristics of consumption of an appliance are called the "load signature". These load signatures are essential for NILM methods. They are often based on active power measurement, but not exclusively. Load signatures can be based on different kinds of measurements. Some of them are listed below.

- Power consumption. Signature can be based on the active, reactive, and apparent power signals. The active power (P) measured in Watts (W) is the power consumed by an electrical device to perform work. This power will be transformed into heat, light, or movement. The reactive power (Q) measured in volt-amperes reactive (var) quantifies the energy alternately stored in and released from magnetic fields (within inductors) or electric fields (within capacitors). This

storage and release of energy induces a phase lag between voltage and current. The apparent power (S) measured in volt-amperes (VA) is the product of the current and voltage in an AC circuit. It represents the total power flowing through the circuit, including active and reactive power. Figure 6 represents on a (P, Q) plane 5000 switching events from the REDD dataset (see Section 4.1). As illustrated in Figure 6, certain appliances solely consume active power, while others consume both active and reactive power. Appliances consuming only active power are considered purely resistive. Resistive appliances include bread toasters, electric boilers, and electric heaters. Conversely, appliances that consume reactive power are typically inductive and characterized by internal windings. Common examples are microwaves and washing machines. Capacitive appliances, which generate reactive power, are relatively rare. The mathematical formulations of active, reactive, and apparent power (S) are given in equations (2), (3), (4). In these equations, ϕ indicates the phase lag between voltage and current. Where ϕ is positive when the current lags behind the voltage (as in inductive circuits) and negative when the current leads the voltage (as in capacitive circuits). This phase lag is due to inductive or capacitive components in the electric circuit. V_{rms} and I_{rms} are the rms voltage and current.

$$P = V_{rms} I_{rms} \cos\phi \tag{2}$$

$$Q = V_{rms} I_{rms} \sin\phi \tag{3}$$

$$S = \sqrt{P^2 + Q^2} \tag{4}$$

- Current and voltage characteristics. The current and voltage waveform, magnitude, and phase angle generated by an appliance when it is in use can be used as a signature.
- Transient events. The transient is the short period following the switching of an appliance. During this period, the quick changes in energy consumption are unique for every appliance and depend on the physical characteristics of the appliance. Figure 4 shows an example of a transient. Indeed, when the fridge is switched ON, a power spike occurs before the power reaches steady state.
- Harmonic components. Harmonics in an electrical signal are voltage or current waveforms whose frequencies are integer multiples of the fundamental frequency (50 Hz in Europe and 60 Hz in North America). The harmonics appear when the relationship between voltage and current in the load is not linear. The harmonics are potentially unique for each appliance and can thus be a way to distinguish appliances in the aggregated signal.

The accuracy of a NILM method depends heavily on the kind of signature and the algorithm used to detect and disaggregate them. The selection of a specific load signature for NILM applications is determined by the capabilities of the smart meter. For instance, signatures that rely on harmonics or the current waveform require a smart meter capable of very high-frequency sampling. At 50 Hz (i.e., 50 cycles of current per second), the sampling frequency must be at the kHz level to capture the current waveform. Conversely, signatures focusing on steady-state power consumption can be identified with smart meters that sample at lower frequencies with a sampling period ranging from approximately one second to fifteen minutes. Additionally, using signatures based on specific signals like reactive power or current necessitates the availability of these signals in the digital meter. Today, modern smart meters like the Belgian smart meters *s211* (see Section 5) measure active power and other signals at a frequency of 1 Hz [8].

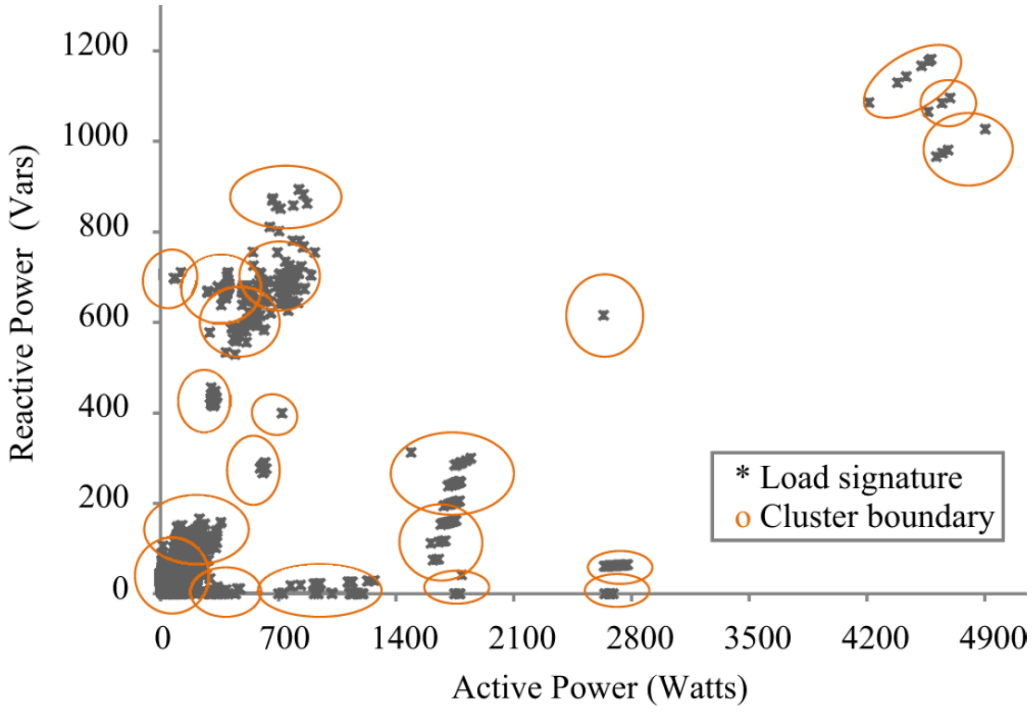

Figure 6: Switching event from REDD dataset represented on a (P, Q) plane from [9].

2.3 Taxonomy of NILM Methods

The literature presents a wide variety of algorithms and approaches to address the challenge of energy disaggregation. These methods can be categorized in several ways, and no single taxonomy can fully capture all the characteristics of NILM techniques. In this document, NILM methods are classified according to three dimensions: whether they are event-based or non-event-based, whether they use supervised or unsupervised learning, and whether they are designed for classification or regression tasks. In theory, all combinations of these categories are possible, and a specific algorithm may fall into different categories depending on its implementation. For instance, an algorithm could be used in an event-based classification context in one study, and in an event-less regression context in another. These categories are further detailed in the following sections, with representative examples discussed in the algorithm review in Section 3.

2.3.1 Event-based and event-less methods

Event-based NILM methods focus on detecting and identifying events. An event is a significant change in the aggregate power (or current) signal. A change (increase or decrease) above a certain threshold in the power signal indicates that an appliance has been switched ON or OFF or has changed its power state (for multi-state appliances). Once an event is detected, the signature linked to this event is extracted. After that, an appliance identification algorithm uses the signature extracted to identify the appliance responsible for the event. Finally, with knowledge of the appliance and its operational state, the appliance's power (and thus consumption with operation time) can be estimated. The main steps of an event-based NILM method are summarized below:

1. Data acquisition

2. Data pre-processing
3. Event detection
4. Event signature extraction
5. Appliance identification
6. Energy disaggregation

On the other hand, eventless methods do not depend on the detection of events. Instead, they analyze the entire consumption signal. The overall aggregated data is usually divided into smaller sections to reduce the computational requirement. These sequences of aggregated power are disaggregated one by one by the algorithm. The steps of an event-less NILM method are summarized below:

1. Data acquisition
2. Data pre-processing
3. Energy disaggregation

The first two steps, data acquisition and pre-processing, are common to event-based and eventless methods and, thus, to every NILM method. In the data acquisition step, the digital meter measures and records the aggregated consumption of the building. In the data pre-processing step, the data is prepared in an optimal format for the next steps of the methods. Further details about the pre-processing step are provided in Section 2.4.

2.3.2 Supervised and unsupervised methods

Supervised and unsupervised methods differ from each other in the kind of data needed for training and selecting the model's parameters. In supervised methods, a labeled dataset is required for training. In the context of NILM, a labeled dataset is a dataset that contains the aggregated consumption of the building as well as the consumption of the appliances present in the building for the same period. A labeled dataset contains, at the same time, the input of any disaggregation algorithm (the aggregated data) and the target (the consumption of the appliances). These datasets are the main limitation of the development and implementation of supervised methods, as they are complex to obtain. To obtain a labeled dataset, the consumption of each appliance must be measured by a sensor while at the same time, the total consumption of the building is also measured at the smart meter level. Open-access labeled datasets exist for research purposes, but due to the complexity of the measurement campaign necessary to obtain the labeled datasets, their number, size, and diversity are limited. Some of them are detailed in section 4.1.

On the other hand, unsupervised methods do not require labeled datasets to adjust their parameters, which is their main advantage. Nevertheless, information about consumption is still needed to tune the algorithms. The minimum information necessary is the number of states of each appliance and the mean power level associated with each state. More advanced algorithms may need separate measurements of each appliance's consumption. Those appliance consumptions can be measured separately, in contrast with labeled datasets where the appliance and the aggregated consumptions must be measured during the same period at the same timestep. Data for unsupervised methods are easier to obtain, but measurement campaigns are still necessary.

Semi-supervised methods use labeled and non-labeled datasets to train their parameters. Thus, they can use all the available data for their training to increase their performance and generalization. Semi-supervised methods are rare in the literature. An example of semi-supervised methods is detailed

in section 3.5.6.

Finally, supervised, unsupervised, or semi-supervised methods require labeled datasets for testing. Indeed, the only way to assess an algorithm's performance is to test it on aggregated data from which the actual consumption of individual appliances is known. This true consumption is often called the ground truth. Thus, a labeled dataset is necessary no matter the method, and new measurement campaigns to create a labeled dataset are necessary, as discussed in this document.

2.3.3 Classification and regression methods

In the introduction, we defined NILM as a method to estimate the consumption of individual appliances from a single-point measurement, the smart meter. However, in non-intrusive load monitoring, the monitoring of appliances can be limited to determine whether an appliance is operating or not. Identifying if an appliance is ON or OFF is a classification problem, while estimating the appliance's energy consumption in kWh is a regression problem. The objectives, implementation strategies, and performance metrics for these two approaches differ significantly. Some of these metrics are detailed in section 4.2. Both event-based and eventless methods can do classification and regression. For event-based methods, the appliance identification step is a classification problem. After classifying an event, the consumption can be estimated by considering the power of the appliance to be constant and equal to a known value during the time of use of the appliance. In the literature, most event-based methods focus only on the classification problem, as the performance for regression tends to be lower. Choosing a regression or classification method depends only on the goal behind the NILM implementation.

2.4 Pre-processing

As discussed in Section 2.4, the pre-processing step is a crucial component common to all NILM methods. The primary objective of this stage is to transform raw measurements obtained from smart meters into a format that is optimal for the subsequent stages of the NILM process. The specific operations performed during pre-processing depend on both the quality and characteristics of the data as well as the requirements of the disaggregation algorithm. Consequently, the extent of pre-processing may vary, ranging from basic data cleaning to more complex transformations. Below is a non-exhaustive list of typical operations that may be performed during the pre-processing phase.

- Handling of missing data and handling of outliers. Missing data and outliers (aberrant values of consumption) can be filled in by interpolation methods, by a copy of the previous data, or can be ignored.
- Noise reduction. Filters and moving averages can be used to keep, for example, only the steady-state features.
- Re-sampling. The sampling frequency of the data measured by the smart meter can be reduced to a frequency adapted to the method used.
- Feature Scaling or Data Normalization. Machine learning and deep learning models are widely used to address the NILM problem, as discussed in Sections 3.4 and 3.5. Feature scaling adjusts the data values to ensure consistent input for these models. Common methods include z-score normalization, min-max scaling, and robust scaling. For example, z-score normalization adjusts the data to have a mean of zero and a standard deviation of one. These techniques help ensure that different features contribute equally to the model training process [10], [11], [12].
- Feature engineering. Feature engineering is the creation or extraction of new features from existing ones. For instance, the aggregated current signal can be transformed from the time domain to the frequency domain with the Fourier transform for harmonics analysis. Another example can be calculating apparent power from the voltage and current.

- Balancing. Machine learning models tend to perform poorly on unbalanced datasets [13]. In the case of NILM, if an appliance is ON a tiny portion of the time in the training data, the model will tend to predict this appliance as always OFF. The solution to this problem can be to increase the proportion of the running appliance in the dataset. It can be done by oversampling the operating period of the appliance or undersampling its OFF period.
- Data segmentation. As explained before, event-less methods cannot process the entirety of the sequence of aggregated consumption at one time. The long sequence can be reduced into shorter sequences or windows of aggregated consumption. These sequences are then treated sequentially by the algorithm.

3 Algorithms review

This section presents a comprehensive overview of most of the existing NILM algorithms. Compared to other reviews like [14] and [15], this review expands on the functioning of these algorithms in greater detail. A better comprehension of how the NILM methods work allows the reader to make better and more justified choices of algorithms given a particular situation. This detailed review is also more comprehensive for less advanced readers in NILM.

This section is structured as follows: Sections 3.1 to 3.7 each present a distinct class of NILM algorithms. Section 3.1 introduces Hart's pioneering algorithm, which laid the foundation for NILM research. Section 3.2 explores optimization-based algorithms, which frame NILM as a mathematical optimization problem. Section 3.3 delves into Hidden Markov Models (HMMs) and their extensions, which leverage probabilistic modeling to infer appliance states. Section 3.4 reviews traditional machine learning approaches, followed by Section 3.5, which focuses on deep learning techniques, including neural networks and advanced architectures. Section 3.6 discusses methods based on voltage-current (V-I) trajectory analysis, which exploit the electrical characteristics of appliances for disaggregation. Finally, Section 3.7 covers less common or emerging NILM approaches and tools that do not fit into the previous categories. For each class of algorithms, this review first outlines the general methodology and governing equations applied to NILM. Then, key implementations from the literature are briefly described as illustrative examples. As will be demonstrated, each algorithm class allows for a variety of implementations, exhibiting diverse levels of complexity and performance.

3.1 Hart algorithm

In 1992, George W Hart introduced the concept of NILM [16]. In his paper, Hart described most of the NILM concepts like a signature (steady state, transient, harmonics), appliance type (ON-OFF, MS, continuous), etc., that are still discussed today in state-of-the-art papers. In addition, Hart proposed a simple NILM algorithm based on step-change detection in active and reactive power signals. The algorithm detects power change between steady-state periods. If the change is below a certain threshold, it is discarded. If not, the algorithm tries to match it to a known appliance model. If there is no close match, the change is ignored. After a step-change is successfully associated with an appliance, the appliance's energy consumption is calculated based on the running time and the power level. This first implementation has many limitations. For instance, it is limited to two-state appliances and cannot efficiently detect different appliances with the same power step-change.

3.2 Optimization algorithms

The methodology behind optimization-based NILM is to represent the energy disaggregation problem as an optimization problem, where the objective is to minimize the difference between the aggregated measured power and the sum of the estimated power consumptions of individual appliances. This method does not need a labeled dataset. The only information needed is the number of appliances in buildings, the different possible states of each appliance, and the power associated with these states. This information can be extracted by taking measures on the appliances only, or it can be asked from the manufacturer. It is easy to add new appliances to the algorithm. Often, the power states of an appliance are extracted by applying clustering algorithms like K-means on appliance consumption measurements (section 3.4.1).

The optimization problem can be represented as follows. Consider a building with a finite number of appliances. These appliances are type 1 (two-state appliances) or type 2 (multistate appliances). The multistate appliances are divided into multiple virtual two-state appliances, one virtual appliance for each running state of the MS appliance. The building is now composed of m two-state appliances,

each with a power consumption $p_i > 0$. The aggregated power at the time step n, $P[n]$, can be represented with the equation (5).

$$P[n] = \sum_{i=1}^{m} x_i[n]p_i + e[n] \tag{5}$$

Where $x_i[n]$ is the status of the ith appliance at the time step n: $x_i = 0$ if the ith appliance is OFF and $x_i = 1$ if the ith appliance is ON. The objective function of the optimization problem can then be formulated with equation (6): For each time step n, the disaggregation is done by finding the vector of status x that minimizes the difference between the aggregated power and the sum of the appliances' power.

$$\underset{x[n]=x_1[n],x_2[n],\ldots,x_m[n]}{\operatorname{argmin}} \left| P[n] - \sum_{i=1}^{m} x_i[n]p_i[n] \right| \tag{6}$$

Most of the time, the objective function is formulated as the least-squares error, which penalizes larger errors:

$$\underset{x[n]=x_1[n],x_2[n],\ldots,x_m[n]}{\operatorname{argmin}} \left(P[n] - \sum_{i=1}^{m} x_i[n]p_i[n] \right)^2 \tag{7}$$

This optimization problem is a combinatorial optimization problem because it consists of finding the best solution from a finite set of possible solutions. Of course, it is supposed that one appliance cannot be in two different states simultaneously. If the number m of appliances is very small, the optimization problem can be solved by testing all possible solutions and selecting the one that minimizes equation (6) or (7). As the number of appliances increases, the search space becomes too large, and this method becomes intractable. Optimization algorithms are then used to try to find the best combination of appliances. The methods to find the solution of the optimization are, for instance, Linear Programming (LP), Integer Linear Programming (ILP), Dynamic Programming (DP), Genetic Algorithms (GAs), or Evolutionary Algorithms. This basic implementation has two main weaknesses. First, the non-uniqueness of solutions: multiple appliance combinations can produce the same aggregated power, making it difficult to uniquely disaggregate the contributions of individual appliances. Second, unrealistic switching behavior: the optimization objectives in equations (6) and (7) do not inherently enforce realistic appliance operation constraints. As a result, the solution may exhibit high-frequency switching, where an appliance turns ON and OFF at every time step, which contradicts real-world behavior. In practice, most appliances have a minimum operating duration once switched ON, a constraint that is not captured in this formulation. This initial simple implementation can be improved by modifying the objective function, adding other objectives (Multi-Objective optimization), or adding constraints to the optimization (constrained optimization). These implementations can be classed generally as unsupervised and eventless implementations (see section 2.3). In this kind of implementation, the output can be the energy consumption or the operational state of each appliance. Thus, regression and classification metrics (section 4.2) can be used to estimate the algorithm's performance.

Implementations from literature

This box summarizes key studies from the literature on the application of optimization methods for NILM.

In [17], the authors compared two multi-objective NILM optimization methods. In the first

method, the two objectives are the minimization of the active power difference (6) and the minimization of the soft clustering distance (SCD) based on the Hamming distance. The Hamming distance gives the number of elements in which two vectors differ. The minimization of the SCD helps avoid an unrealistic number of appliance switchings. In the second multi-objective optimization method, the two objectives are the minimization of the active power difference (equation 6) and the minimization of the reactive power difference. To solve these optimization problems, the authors used the Non-dominated Sorting Genetic Algorithm II (NSGA-II). In multi-objective optimization, the acceptable solutions form a Pareto front, and a decision-making function is necessary to select one of these solutions. In the paper, adding a second objective improved the performance of the algorithms.

In [18], the authors proposed a multi-objective NILM optimization method with five objective functions using active power, reactive power, apparent power, currents, and harmonics. The Moth Flame optimization algorithm (MFO) is used to solve the optimization problem. The MFO is categorized as swarm intelligence or nature-inspired algorithms. In addition, the Factorial Hidden Markov Model (see section 3.3) is used to determine the next possible state of the appliances and reduce the complexity of the optimization. Indeed, some of the states can follow a specific order for multi-state appliances. For a washing machine, the spin-drying never precedes the washing step.

In [19], the authors proposed a NILM optimization method that uses the amplitude and phase of the fundamental current. The amplitude and phase are obtained by applying a fast Fourier transform on the current signal (see section 3.7.6. The objective function is created knowing that the sum of the real part of the appliance's current must be equal (with respect to error) to the real part of the aggregated current and that the sum of the imaginary part of the appliance's current must be equal to the imaginary part of the aggregated current. The optimization problem is solved with an Artificial Bee Colony algorithm.

In [20], the authors compared different state-of-the-art optimization-based algorithms on the same dataset and with the same metrics. In this comparison, none of the algorithms outperformed the others on all appliances.

In [21], the authors proposed an optimization-based algorithm for the industrial context. In the industrial context, Variable Frequency Drives (VFD) are often used. VFDs are continuously variable loads and are especially challenging for classical optimization-based algorithms, which are more adapted to steady-state appliances. To tackle this challenge, the authors propose a mixed-integer program that separately models CV load (type 3) and steady-state loads (type 1 and 2). In addition, temporal dependencies between the appliance and the state duration of the appliance are used as constraints to improve the algorithm's performance.

In [22], the authors formulated the disaggregation problem as a Constrained Multi-Objective Problem (CMOP). The two objectives are the sparsity and the disaggregation error. The constraints are appliance-specific, like the number of state switches per operation. The optimization problem is solved with a Constrained Multi-Objective Evolutionary Algorithm (CMOEA). As it is a multi-objective problem, the authors select the solution with the lowest disaggregation error from the set of solutions.

3.3 Hidden Markov Models

Hidden Markov models are widely used in the literature for energy disaggregation. To understand how they work, it is first essential to understand what a Markov chain is. A Markov chain can be defined as

a probabilistic model that represents a series of events, where the likelihood of each event is determined solely by the state reached in the event before it. This memoryless property of the sequence is called the Markov property. Most of the mathematical development in this section comes from [23].

Figure 7 represents an example of a Markov process or Markov chain with three states (x_1, x_2, and x_3). In this process, the probability of each event depends only on the state of the previous event. The Markov process can be represented by the transition matrix A in equation (8). A possible sequence of events for this sequence can be $x_2 \rightarrow x_1 \rightarrow x_1 \rightarrow x_3 \rightarrow x_1 \rightarrow x_2$ etc. With this Markov process, it is possible to have a sequence with two consecutive states x_1 because the probability of x_1 to x_1 is 0.4. However, having two consecutive states x_2 or x_3 is impossible.

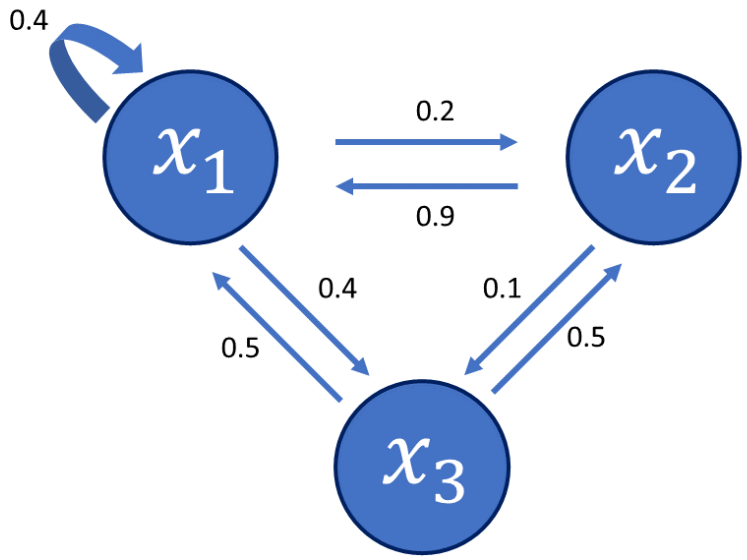

Figure 7: Representation of a three-state Markov process.

$$A = \begin{bmatrix} 0.4 & 0.2 & 0.4 \\ 0.9 & 0 & 0.1 \\ 0.5 & 0.5 & 0 \end{bmatrix} \tag{8}$$

For the example presented in Figure 7, the probability of having the state x_2 at the time $t+1$ knowing the state is x_1 at the state t, is 0.2. Mathematically:

$$P(s_{t+1} = x_2 | s_t = x_1) = 0.2 \tag{9}$$

For a Markov process with an ensemble of n possible states noted $X = \{x_1, x_2, ..., x_n\}$, and a sequence of T states noted $S = \{s_1, s_2, ..., s_T\}$. The Markov property can be expressed by the equation (10).

$$P(s_{t+1} = x_i | s_1, s_2, ..., s_t) = P(s_{t+1} = x_i | s_t) \tag{10}$$

In long sequences, some states can be more present than others. Indeed, each state has a different probability distribution. The probability distribution of each state is given by the stationary

distribution π, also called the initial probability vector. This vector π can be estimated with very long sequences (the probability of each state is calculated by the number of occurrences of the state divided by the total number of elements in the sequence). Mathematically, the vector π is the left eigenvector of the transition matrix A (equation 11).

$$\pi A = \pi \tag{11}$$

A Hidden Markov model (HMM) is a Markov model where the sequence of states S is non-observable. The states of the Markov process are hidden. For a sequence of non-observable states S (called the hidden sequence), there is always a sequence of observations Y of the same size. In an HMM, the observation at time t only depends on the hidden state at time t.

$$P(y_t = o_i|s_1, s_2, ...s_t) = P(y_t = o_i|s_t) \tag{12}$$

The probabilities of the observations given a hidden state are called emission probabilities. The possible observations can be either discrete or continuous. If the observations are discrete, their emission probabilities form the emission matrix B. An example of a Hidden Markov Model with three possible hidden states (x_1, x_2, and x_3) and two possible observations (o_1 and o_2) is given in Figure 8. The emission matrix for this HMM is given in equation (13).

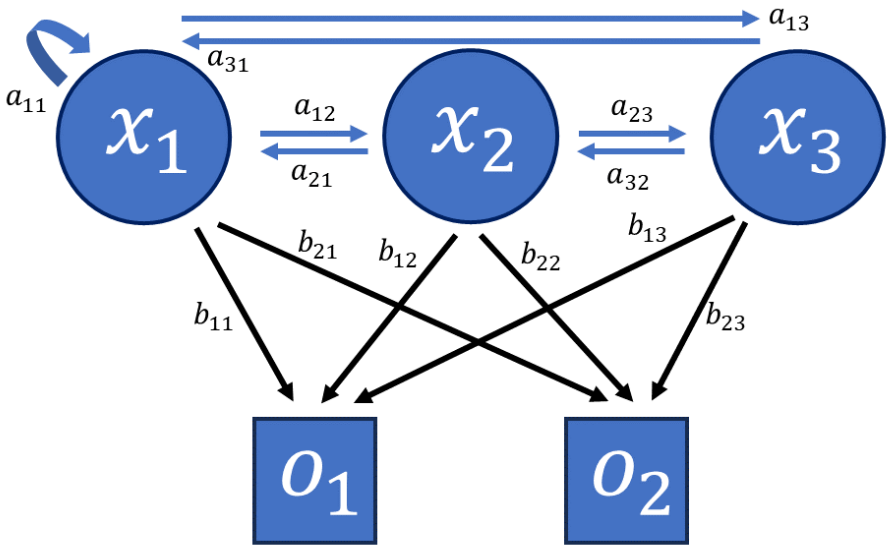

Figure 8: Representation of a three-state HMM with two possible observations.

$$B = \begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \end{bmatrix} \tag{13}$$

An example of a sequence of hidden states and observations in an HMM is given in Figure 9.

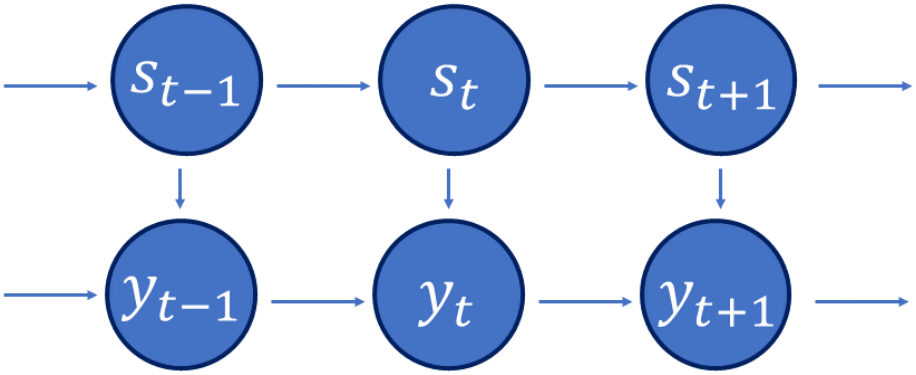

Figure 9: Sequence of observations and hidden states in an HMM.

When modeling a process with an HMM, the idea is to be able to solve the following problem: From a sequence of observable variables Y, find the hidden sequence S for which the probability of S knowing Y is maximal. Mathematically:

$$\underset{S=s_1,s_2,...,s_T}{\text{argmax}} \quad P(S = s_1, s_2, ...s_T | Y = y_1, y_2, ...y_T) \tag{14}$$

Thanks to Bayes's theorem, equation 14 can be rewritten in equation 15.

$$\underset{S=s_1,s_2,...,s_T}{\text{argmax}} \quad P(S|Y) = \underset{S=s_1,s_2,...,s_T}{\text{argmax}} \quad \frac{P(Y|S)P(S)}{P(Y)} = \underset{S=s_1,s_2,...,s_T}{\text{argmax}} \quad \frac{P(S,Y)}{P(Y)} \tag{15}$$

Where $P(S|Y)$ is the conditional probability of S knowing Y and $P(S,Y)$ is the joint probability of S and Y (the probability of both events S and Y occurring simultaneously). For a given sequence of observations Y, the sequence S that maximizes equation (15) is the same regardless of the denominator. The denominator can thus be neglected. Given the particularity of the HMM, equations (10) and (12), the expression can be rewritten again in equation (16).

$$\underset{S=s_1,s_2,...,s_T}{\text{argmax}} \quad P(S,Y) = \underset{S=s_1,s_2,...,s_T}{\text{argmax}} \quad P(s_1)P(y_1|s_1) \prod_{t=2}^{T} P(y_t|s_t)P(s_t|s_{t-1}) \tag{16}$$

In this equation (16), the term $P(s_1)$ is given by the initial probability vector π, the terms $P(s_t|s_{t-1})$ are given by the transition probabilities and the terms $P(y_t|s_t)$ are given by the emission probabilities. Suppose the parameters of the HMM are known (i.e., the transition probabilities, the emission probabilities, and the stationary probabilities). In that case, it is possible to calculate for a sequence of observations Y the probability of each possible sequence of states S. Then, the sequence S with the highest probability can be selected as the estimated true sequence. The probability of each possible S sequence can be calculated efficiently with the forward-backward algorithm [24]. This way of estimating the sequence S with Y is known as "brute force." The Viterbi algorithm [25] can be used to find the hidden sequence S with the highest probability more efficiently without calculating the probability of every sequence.

To determine the most probable sequence S of hidden states, the transition, emission, and stationary probabilities must be known. If a labeled dataset is available, containing both the observations Y and the corresponding hidden states X, these probabilities can be estimated straightforwardly. However, one of the most essential advantages of HMM is that it can be an unsupervised method. The parameters of the HMM can be learned with only observations thanks to an Expectation-Maximization (EM) algorithm. The expectation-maximization algorithm is an iterative optimization algorithm that can

be used to estimate unknown parameters in a statistical model. It comprises two steps that will be done iteratively until the convergence of the unknown parameters. The first step is the expectation step (E-step). Before the first E-step, the model's parameters must be set with initial values. These values can be set randomly or to an initial guess close to the actual value, thanks to previous knowledge. Setting the parameters with previous knowledge increases the probability of converging to a global maximum. During the E-step, the expected value of the log-likelihood function given the current parameters is calculated. During the M-step, the model's parameters are updated to maximize the log-likelihood calculated in the E-step. The likelihood function represents the probability of the observed data given a set of parameters in a statistical model. Maximizing the log-likelihood (the logarithm of the likelihood function) is equivalent to maximizing the likelihood function, but simplifies the calculation and avoids numerical imprecision. The M-step adjusts the parameters to increase the likelihood of the observed data under all possible hidden state sequences S. These two steps are done iteratively until the convergence of the model's parameters. The expected log-likelihood of the observed and hidden data is given by equation (17), where θ represents the model's parameters. In the case of the Hidden Markov Model, the Expectation-Maximization algorithm used to estimate the unknown parameters is called the Baum-Welch algorithm.

$$Q(\theta_{new}|\theta) = E[logP(Y, X|\theta_{new})|Y, \theta] \tag{17}$$

For energy disaggregation, the observed sequence Y of the HMM model is the aggregated consumption measured at the smart meter. Each hidden state represents one possible combination of all the appliance consumption states. Considering a building with M appliances, each with K power states, the HMM model used to represent this building will have K^M possible hidden states. The model will already have 27 possible hidden states for only three appliances, with three states each. The transition matrix will have $K^M \times K^M$ elements (729 elements for three appliances with three states each). As illustrated with this simple example, the complexity of the problem increases rapidly with the number of appliances, and the problem quickly becomes intractable.

Factorial Hidden Markov Models (FHMM) are introduced to reduce the computational complexity of HMM. In FHMM, there is not one sequence of hidden states S but M sequences of hidden states S^*. $S^* = \{S^1, S^2, ..., S^M\}$. In the context of energy disaggregation, each hidden sequence will represent the evolution of the consumption of one of the appliances. An example of an FHMM process is given in Figure 10.

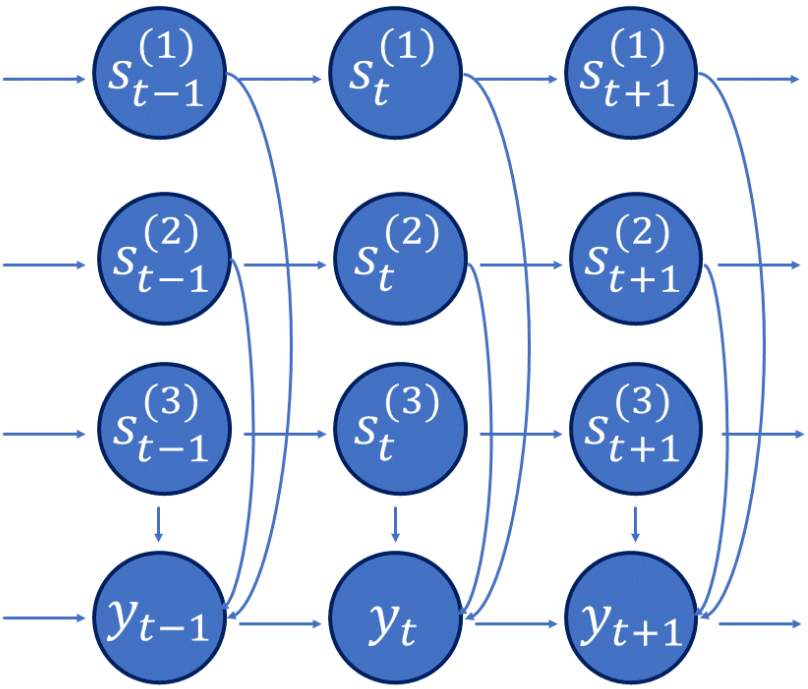

Figure 10: FHMM process with three hidden sequences.

With no additional constraint, an FHMM with M Markov chains, each with K possible hidden states, is equivalent to an HMM with one Markov chain with K^M hidden states. The complexity of the model is reduced by considering an FHMM where the consumption state of one appliance is independent of the consumption state of the other appliances. This means that the hidden state of one Markov chain is independent of the hidden state of the other Markov chains. Mathematically:

$$P(s_t^*|s_{t-1}^*) = \prod_{m=1}^{M} P(s_t^m|s_{t-1}^m) \tag{18}$$

Then, for an FHMM with M appliances, equation (16) can be rewritten in equation (19).

$$\underset{S^*}{\operatorname{argmax}}\, P(S^*, Y) = \underset{S^*}{\operatorname{argmax}} \left(P(y_1|s_1^*) \prod_{m=1}^{M} P(s_1^m) \right) \prod_{t=2}^{T} P(y_t|s_t^*) \prod_{m=1}^{M} P(s_t^m|s_{t-1}^m) \tag{19}$$

Where $S^* = \{S^1, S^2, ..., S^M\}$, $S^1 = \{s_1^1, s_2^1, ..., s_T^1\}$ and $s_t^* = \{s_t^1, s_t^2, ..., s_t^M\}$. For FHMM, the exact E-step is computationally intractable, as noted in [23]. To approximate the inference, various techniques have been proposed, including Gibbs sampling [26], structured variational inference, and completely factorized variational inference [23].

The parameters that define an FHMM are the following:

- The number of Markov chains.
- The transition probability matrix A.
- The emission probability matrix B.
- The stationary state probability vector or initial state probability vector π.

In the literature about the use of FHMM for energy disaggregation, the emission probabilities are often considered continuous and modeled by Gaussian distributions [26],[27], [28], [29]. Supposing M appliances and K power states per appliance, the emission probability of the mth appliance is given by $B^{(m)} \sim \mathcal{N}(\mu^{(m)}, \sum^{(m)})$. Where $\mu^{(m)}$ is of length K and contains the mean powers of the states of the mth appliance, and $\sum^{(m)}$ is the precision (the reciprocal of variance) associated with these consumption states. One of the principal advantages of NILM based on FHMM is that there is no need for a labeled dataset to train the model. Authors use sequences of consumption of appliances to learn the parameters of FHMM [28] or consider periods during only one appliance is operating [27], [30]. The parameters of the FHMM are estimated using the following procedure: First, an HMM model is created for each appliance, and the parameters of these HMMs are learned by EM or clustering [30] from the measure of consumption of the appliances. Then, the parameters of the FHMM are calculated by combining the parameters of the HMMs of each appliance. When the parameters are learned, the FHMM can then be used to find the consumption of each appliance from the aggregated consumption data. A lot of variations of the FHMM to solve the energy disaggregation problem are proposed in the literature; some of them are mentioned in the next paragraphs. Generally, FHMM for NILM can be categorized as unsupervised and eventless, even though it can theoretically be supervised.

Implementations from literature

This box summarizes key studies from the literature on the application of Hidden Markov Model methods for NILM.

In [26], the authors compared four Hidden Markov Models for energy disaggregation. The first one is a classical FHMM. The second one is a Conditional Factorial Hidden Markov Model (CFHMM). CFHMM extends FHMM by allowing the use of additional features such as time of day, time of the week, sensor measures, etc. In CFHMM, the transition probabilities are not constant. They are conditioned by the additional features. The third algorithm is a Factorial Semi-Hidden Markov Model (FSHMM), which allows the consideration of the state duration of the appliances in the model. The fourth algorithm is a Conditional Factorial Hidden Semi-Markov Model (CFHSMM) that combines the advantages of CFHMM and FHSMM. The authors state that CFHSMM outperforms the three other models.

In [29], the authors proposed a Time-Efficient FHSMM. This TE-FHSMM improved the computational efficiency of classical FHSMM.

In [31], in order to reduce the difficulties of inference with a large number of appliances, the authors propose an approximate inference procedure that exploits the additive structure of the disaggregation problem. Indeed, the observations (the aggregated data) are the sum of the hidden appliance states. Their inference algorithm is called AFAMAP for Additive Factorial Approximate MAP. In this algorithm, the authors constrain the posterior probability to have only one appliance (or HMM) change state at a time.

In [30], the authors proposed to improve the performance of the FHMM disaggregation algorithm by using both the active and reactive power as observations. The inference is made by an alternative version of the AFAMAP algorithm proposed in [31].

In [28], the authors proposed a factorial hidden Markov model based on Adaptive Density Peak Clustering (ADPC). ADPC is used to find the consumption states of each appliance thanks to measures of the appliance's consumption. With this information, an HMM is created for each appliance and then combined in an FHMM. In the context of NILM, the principal advantage of ADPC over other clustering methods like K-means is that the knowledge of the number of

clusters (i.e., the number of power states per appliance) is not needed.

In [32], the authors used Particle Filtering (PF) to infer the appliance states in an FHMM. Particle filtering is used to overcome the shortcomings of the Bayesian approach, such as Gibbs sampling, concerning nonlinear problems and non-Gaussian noise. This improved the performance of the algorithm with nonlinear appliances like a dimmer or a drill.

In [33], the authors presented a super-state HMM for energy disaggregation. The super-state HMM is a classical HMM, where each hidden state or super-state represents a possible combination of appliance states. In this configuration, the number of hidden states increases exponentially with the number of appliances. The number of states is equal to 2^M for M two-state (ON-OFF) appliances. This makes the problem intractable, and this is why authors usually turn to factorial Hidden Markov Models (FHMM). Here, the authors presented a new sparse Viterbi algorithm that took advantage of the sparsity of the matrix in the HMM. This new algorithm can efficiently process sparse matrices. The appliance models are learned with prior appliance consumption information and then combined into the super-state HMM. The main advantage of this "super-state HMM" implementation is that it preserves load-dependence information, where this information is lost in disaggregation with classical FHMM.

3.4 Machine Learning

In the following sections, Machine Learning (ML) algorithms applied to the NILM problem will be presented. There is often confusion between the terms Artificial Intelligence (AI), machine learning, and Deep Learning (DL). Thus, a definition of these terms is given as an introduction to this section.

- Artificial intelligence is a broad area of computer science that focuses on creating programs that can perform tasks that normally require human intelligence. An example of an AI program is a rule-based system, also called an expert system. This program used a pre-defined set of rules to make a decision or solve a problem.
- Machine learning is a subset of AI; it regroups the algorithms that can learn from data to perform a task instead of being explicitly programmed to perform this task.
- Deep Learning is a specialized subset of ML that relies on artificial neural networks (ANNs) with multiple layers to automatically extract and learn hierarchical representations of data. While traditional ML models require manual feature engineering, deep learning can learn relevant features directly from raw data. Neural networks with a small number of hidden layers are considered shallow networks and are classified as ML, whereas those with many layers (deep neural networks, DNNs) fall under deep learning.

In section 3.4, machine learning algorithms that are not considered deep learning algorithms are presented. In [34], the authors propose a comparative overview of different ML algorithms for load classification in an event-based NILM method.

3.4.1 K-means clustering

The K-means algorithm is an unsupervised machine learning clustering algorithm. A clustering algorithm is an algorithm that groups similar data into clusters. For K-means, the number of clusters is set a priori and is equal to k. The algorithm's functioning is as follows: First, k cluster centers (called centroids) are chosen randomly, and then the distance between those centroids and the points of the training dataset is calculated. Euclidean distance is often used by default, but other distance metrics

can also be used. Once all the distances are calculated, each point of the dataset is associated with the nearest centroid to form clusters. Once the clusters are formed, new centroids are calculated as the mean of the clusters' data points. The algorithm iterates through the two steps of calculating distances and updating centroids until convergence.

In NILM, clustering algorithms are often used to obtain the different steady-state power consumption of ON/OFF and Multi-State appliances, like in [30], [35]. Clustering algorithms can also be used to group similar appliance signatures together in a classification algorithm.

Implementations from literature

This box illustrates key instances from the literature on the application of clustering methods for NILM.

In [36], the authors presented a comparative analysis of eight unsupervised clustering algorithms, including K-means, in the context of NILM. The NILM method used for the comparison is event-based. After the detection of an event, transient features are extracted, like the standard deviation and the variation of P, Q, and I during the transient. These features are then used by the clustering algorithms to identify the appliance responsible for the event.

3.4.2 k-nearest neighbors

The k-nearest neighbors algorithm, or KNN, is a supervised machine learning clustering algorithm. As it is a supervised algorithm, a labeled dataset is needed for training. In this case, the labels in the dataset are the clusters assigned to each data point. The algorithm's functioning is the following: For each new point, the algorithm calculates the distance (the Euclidean distance, for instance) between the new point and the points of the dataset (called the neighbors). The neighbors of the new point are then sorted by their distance. The new point belongs to the cluster that is in the majority among the k nearest neighbors. The value of k has to be chosen carefully; a small k leads to overfitting, and a big k leads to underfitting. KNN can also be a regression algorithm. The regression output is then the mean value of the k nearest neighbors.

Implementations from literature

This box illustrates key instances from the literature on the application of KNN methods for NILM.

In [37], the authors used the nearest-neighbor algorithm for load classification on features based on the wavelet transform (see section 3.7.6).

3.4.3 Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm primarily used for classification, but it can also be applied to regression. Given a labeled dataset, SVM identifies the optimal hyperplane that best separates the classes. In a two-dimensional feature space, this hyperplane is a straight line. The optimal hyperplane is the one that maximizes the margin, which is the distance between the hyperplane and the closest data points from each class. In cases where perfect separation is not feasible, some points may be allowed to fall within or on the wrong side of the margin—this is known as a soft margin. When the data is not linearly separable, SVM can map it to a higher-dimensional space where a hyperplane can effectively separate the classes. In this transformed space,

the hyperplane corresponds to a nonlinear decision boundary in the original space. This transformation is achieved by introducing additional features that are nonlinear functions of the existing ones. The process of efficiently computing this transformation without explicitly increasing dimensionality is known as the kernel trick. The concept of SVM is illustrated in Figure 11 from [38].

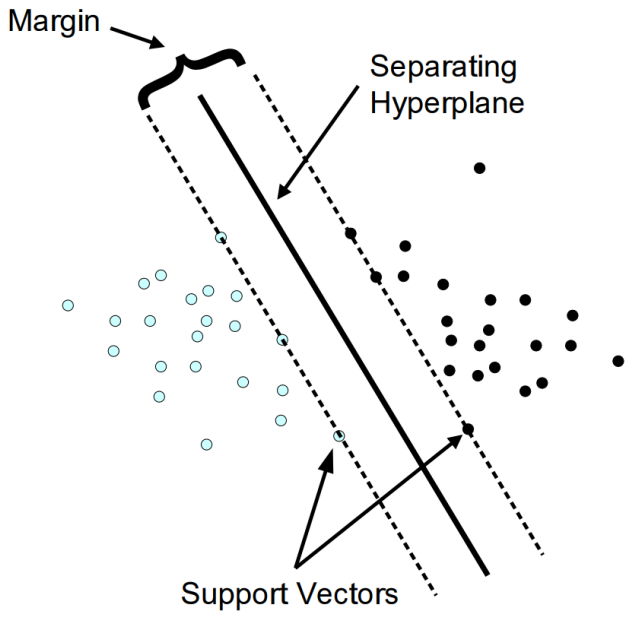

Figure 11: SVM concept for classification with two dimensional features from [38].

Implementations from literature

This box illustrates key instances from the literature on the application of SVM for NILM.

In [39], the authors used SVM to classify load based on features extracted from the V-I trajectory of the appliance, like area under the curve, loop direction, self-intersection, current span, etc.

In [40], the authors used SVM to classify load based on active power, reactive power, and fundamental and harmonics of voltage and current.

3.4.4 Decision Tree

Decision Tree (DT) is a supervised ML algorithm. It can be used for classification and regression. A decision tree is composed of nodes. There are two types of nodes: the decision node and the leaf node. The first decision node is called the root node. At each decision node, the data is split into two nodes by a condition. The condition is chosen during the training phase. The algorithm chooses the condition with the most significant information gain. The information gain is calculated with the formula 21. When all the elements of a node are of the same class, the node is called a leaf, and there is no condition. When the training is complete, each new point will travel through the tree by following the conditions. The new node is classified according to the class label of the leaf it falls into, typically the majority class within that leaf. For regression, the output is the mean value of the mean node. Figure 12 illustrates the decision tree concept. The mathematics developments and the Figure in this

section were inspired by Sujan Dutta's work[3].

$$Entropy = \sum_i -p_i log_2(p_i) \tag{20}$$

$$IG = Entropy(parent) - \sum_j Entropy(child_j) w_j \tag{21}$$

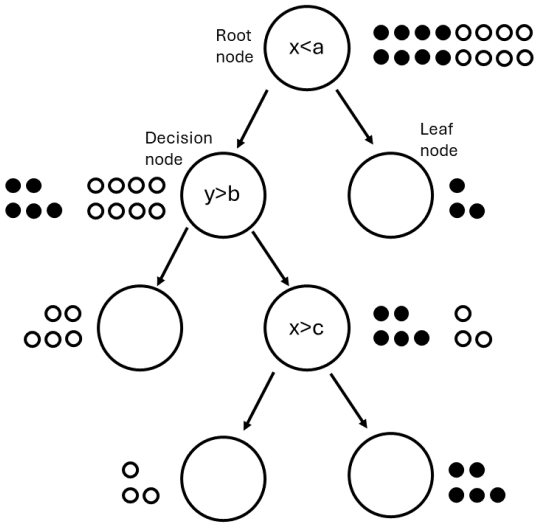

Figure 12: Decision tree concept for classification with two features (x and y)

Entropy is a measure of information in a state; a high entropy indicates that a state contains little information. In equation (20) p_i is the probability of class i. At the root node, in Figure 12, the probabilities of class 0 and 1 are both 0.5. Still, in Figure 12, at the root node, the entropy is 1 because the probability of belonging to class 0 or 1 is the same. During training, the condition that gives the highest information gain is chosen. The information gain in equation (21) is the difference between the entropy of the parent node and the weighted sum of the entropies of the children nodes.

Implementations from literature

This box presents notable cases from the literature on the use of DT methods for NILM.

In [41], the authors use a decision tree in an event-based disaggregation algorithm for load classification using the change in active power as a feature.

In [42], the authors use DT for load classification with the active, reactive, and apparent power change as features.

3.4.5 Random Forest

Random Forest (RF) is a supervised machine learning algorithm used for both classification and regression tasks. It enhances the decision tree algorithm by creating an ensemble of trees. The process

[3] https://github.com/Suji04?tab=repositories

begins by generating multiple datasets from the original dataset through random sampling with replacement. For each of these new datasets, a decision tree is built. During the construction of each tree, a random subset of features is selected to split the nodes, which adds another layer of randomness. Once all the decision trees are trained, each new data point is passed through every tree in the forest. For classification tasks, the final class of the new data point is determined by the majority vote of the trees' outputs. The predicted value for regression tasks is the average of all the tree outputs. This ensemble method significantly reduces the risk of overfitting compared to a single decision tree, providing more robust and accurate predictions.

Implementations from literature

This box presents notable cases from the literature on the use of the RF methods for NILM.

In [43], the authors employed Elliptical Fourier Descriptors (EFD) to analyze the contour of a VI trajectory derived from an event in the aggregated data. They then utilized a random forest algorithm to classify the event as a specific appliance, using the EFD as input features.

In [44], the authors used current, active power, reactive power, and power factor as features in a random forest based multi-target regression algorithm. Each target is the disaggregated power of one appliance. In this implementation, a separate random forest model was trained for each appliance.

3.4.6 Naive Bayesian Classifier

A naive Bayesian classifier is a supervised classification machine learning algorithm. The concept of a naive Bayesian classifier is the following: given a set of features $X = (x_1, x_2, ...x_n)$, the objective is to find the true label Y associated with these features. For every label Y, the following conditional probability can be calculated:

$$P(Y = y|X = (x_1, x_2, ...x_n)) \tag{22}$$

Then, the class or label Y can be estimated with the label having the highest probability. The problem is that the probabilities of equation (22) are hard to find. Using Bayes' theorem, the equation can be rewritten as follows:

$$P(Y|X) = \frac{P(X|Y) \times P(Y)}{P(X)} \tag{23}$$

The first term on the left side of the equation is called the posterior, the first factor on the right side of the equation is called the likelihood, the last factor on the right side of the equation is called the prior, and the term at the denominator is called the evidence. As the idea is to compare the probability of every possible label Y for the same set of features X, the term at the denominator is constant and can be ignored. $P(Y)$ and $P(X|Y)$ can be directly estimated from a labeled dataset with features and labels. $P(Y = y_1)$ is equal to the number of times $Y = y_1$ is present in the dataset divided by the total number of elements in the dataset. The problem is that to find a good estimate of $P(X = (x_1, x_2, ...x_n)|Y = y)$, a very large dataset is needed, and the more features x there are, the larger the dataset needs to be. If the dataset is not sufficiently large, $P(X|Y = y)$ will be equal to zero or close to zero, and the label will be estimated with poor accuracy. This is why the naive assumption is used. The naive assumption is that every feature $x_1, x_2, ..., x_n$ is independent of each other. Mathematically :

$$P(X = (x_1, x_2, ...x_n)|Y = y) = P(X_1 = x_1|y) \times P(X_2 = x_2|y) \times ... \times P(X_n = x_n|y) \tag{24}$$

In real-world scenarios, the features are rarely independent of each other. Nevertheless, a naive classifier can still be effective in many of these scenarios. Naive classifiers need a small dataset and are very simple to implement. Naive Bayesian classifiers are used in NILM event-based methods.

Implementations from literature

This box illustrates key instances from the literature on the application of the Naïve Bayes classifier for NILM.

In [45], the authors employed a Naïve Bayes classifier to classify appliances using a single feature: active power. The same approach could be extended to incorporate additional features such as apparent power, reactive power, and others, potentially improving classification performance.

3.4.7 Ensemble model

In machine learning, an ensemble model combines the predictions of multiple algorithms to enhance accuracy and robustness. The key idea is to aggregate several 'weak' models into a more powerful 'strong' model, where each weak model captures different aspects of the data. Ensemble methods can be broadly categorized into three types:

- Bagging: Multiple instances of the same algorithm are trained on different subsets of the data to reduce variance and mitigate overfitting. A classic example of this approach is Random Forest, as explained in Section 3.4.5.
- Boosting: Models are trained sequentially, with each model learning from the errors of its predecessors. This iterative refinement process improves prediction accuracy by giving more weight to misclassified instances.
- Stacking: Different models are trained independently, and their predictions are combined—often using a meta-model—to generate a final, more accurate prediction.

Implementations from literature

This box highlights key examples from the literature on the application of ensemble models to NILM.

In [46], the authors combine different regression ML algorithms such as decision tree, random forest, KNN, and others to create an ensemble model. The parameters of the different base models are tuned by Bayesian optimization during training.

3.5 Deep learning

Artificial Neural Networks (ANNs) are computational models inspired by the structure and function of biological neural networks in animal brains. The term ANN refers to any neural network with one or more layers and encompasses various architectures, including Multilayer Perceptrons (MLP) (Section 3.5.1), Recurrent Neural Networks (RNN) (Section 3.5.2), and Convolutional Neural Networks (CNN) (Section 3.5.3). Generally, an ANN with more than two hidden layers is considered a Deep Neural Network (DNN). The increased depth allows DNNs to capture complex patterns and hierarchical representations in data, though at a higher computational cost. Neural networks can be applied to both classification and regression tasks. Their parameters are learned during training, typically in a supervised manner. Training is usually performed using backpropagation, a gradient-based optimization method that adjusts the network's weights by propagating errors backward from the output layer to the input layer, enabling the model to learn from its mistakes and improve its predictions iteratively.

3.5.1 Multilayer Perceptron

Multilayer Perceptrons (MLPs) consist of multiple layers of interconnected neurons. The structure of a single neuron is illustrated in Figure 13. Given an input vector $X = (x_1, x_2, ...x_n)$, a neuron generates an output through three sequential operations. First, it computes a weighted sum of the inputs. Next, a bias term is added to this sum. Finally, the result is passed through a nonlinear activation function. The output of a neuron is given by equation (25) and is then transmitted to other neurons in the network.

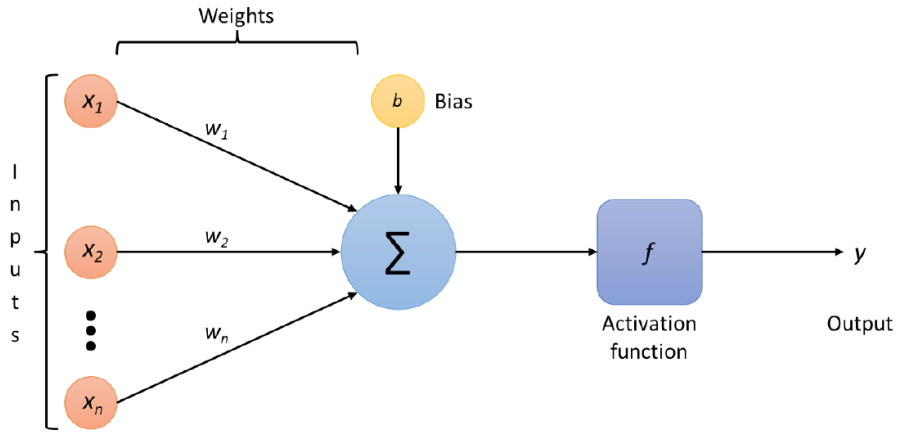

Figure 13: Structure of a neuron in an ANN from [47].

$$y = f\left(\sum_{i=1}^{n} x_i w_i + b\right) \tag{25}$$

MLPs consist of three types of layers: an input layer, one or more hidden layers, and an output layer. The number of neurons in the input layer corresponds to the number of input variables, while the number of neurons in the output layer matches the number of output variables. For instance, in a classification task with five input features and three classes, the MLP will have five input neurons and three output neurons. Figure 14 illustrates an MLP with one hidden layer, three input neurons, and two output neurons. During supervised training using backpropagation, the network iteratively adjusts its weights and biases to minimize prediction errors and improve accuracy.

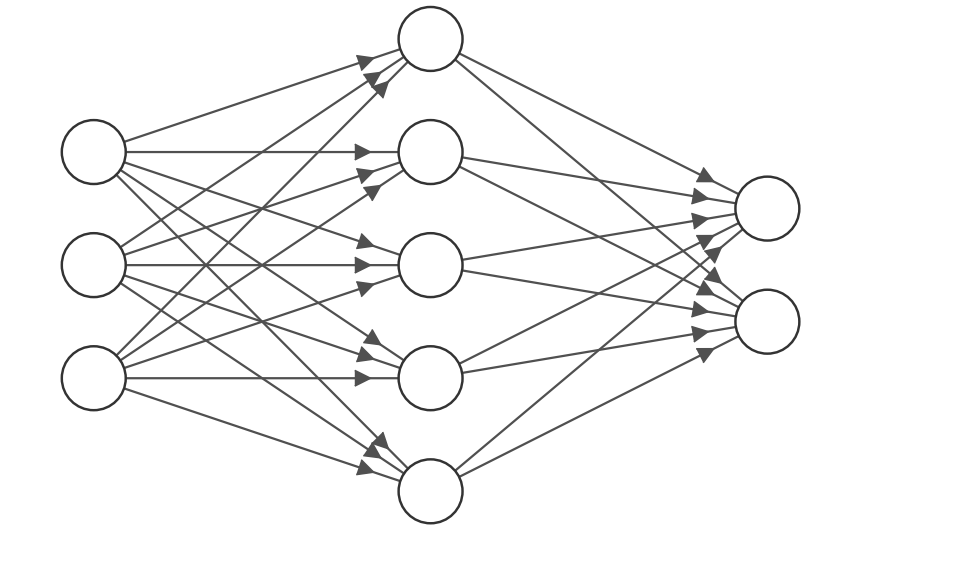

Figure 14: Structure of an MLP with three layers from [48]

In the literature, the term "ANN" is often used interchangeably with "MLP", which can cause confusion. In this document, "ANN" and "DNN" refer to all types of neural networks, including MLPs, CNNs, and RNNs. Typically, the term "MLP" is associated with networks with a few layers, also known as shallow networks. For networks with structures similar to Figure 15, which have more layers, the term "deep fully connected neural network" is more appropriate.

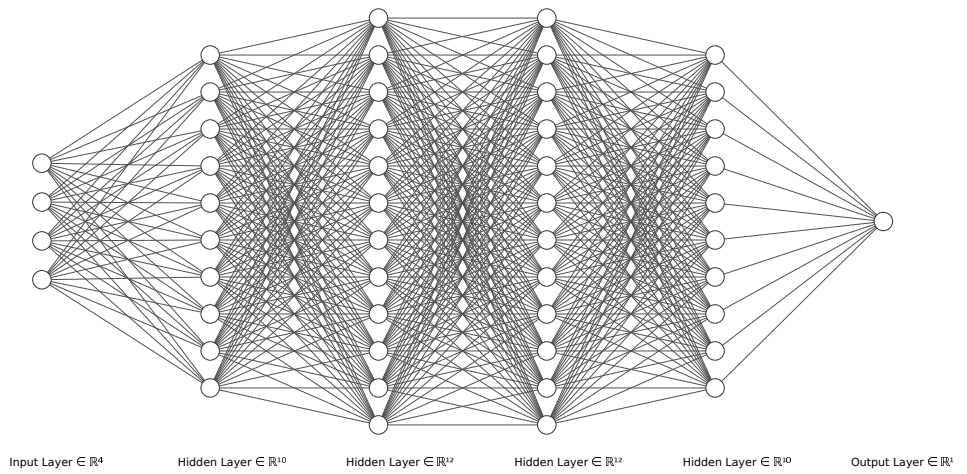

Figure 15: Deep fully connected neural network (created with [48]).

In the context of NILM, features such as steady-state active and reactive power, as well as the fundamental and harmonic components of current, can be used as inputs to an artificial neural network. Using these features, the ANN can then classify whether an appliance is ON or OFF during an event, or estimate its power consumption. The inputs to the ANN can also be a sequence of measured power values, while the output may indicate either the state of an appliance (ON/OFF) or its estimated power consumption over that sequence. MPLs are inefficient when dealing with time series as input.

Different types of neural networks, such as recurrent neural networks (Section 3.5.2) or convolutional neural networks (Section 3.5.3), are better suited to this kind of input. It is also important to note that either a separate single-target neural network can be trained for each appliance, or a multi-target network can be trained to handle multiple appliances simultaneously

In [49], the authors proposed a deep neural network with fully connected layers, consisting of three hidden layers, each containing five hundred nodes, to classify appliance profiles. These appliance profiles were generated using an edge detection algorithm. In this event-based, supervised NILM method, a separate mono-target neural network was trained for each appliance.

3.5.2 Recurrent neural network

Recurrent Neural Networks (RNN) are neural networks designed to process sequential data. Thanks to an internal hidden state, RNNs can have a memory of previous input, influencing the processing of future inputs. This capability is absent in standard artificial neural networks. However, traditional RNNs encounter challenges capturing long-term dependencies within data sequences, primarily due to the vanishing or exploding gradient problem, which complicates the learning process for events that occur at significantly different time steps. To overcome these limitations, Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) have been developed as advanced variants of RNN. These models introduce mechanisms to selectively retain or discard information over long periods, significantly enhancing the network's ability to learn from long-term dependencies. As a result, LSTMs and GRUs have largely surpassed traditional RNNs in popularity and performance, offering more robust solutions for complex sequence modeling tasks. A representation of a cell of an RNN is given in Figure 16, the elements of a time series are treated sequentially, and the memory of previous input is kept in the hidden state h.

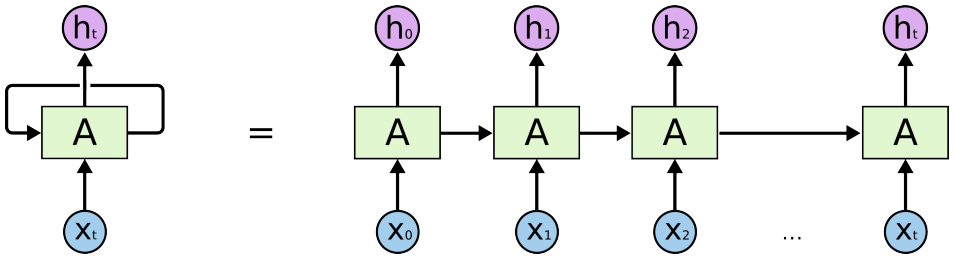

Figure 16: An unrolled Recurrent Neural Network from [50].

The representation of a cell of RNN is given in Figure 17. The hidden state at time t $h(t)$ of an RNN is calculated thanks to equation (26). Where U is a weight matrix for the input, W is a weight matrix for the hidden states at time $t-1$, and b is the bias.

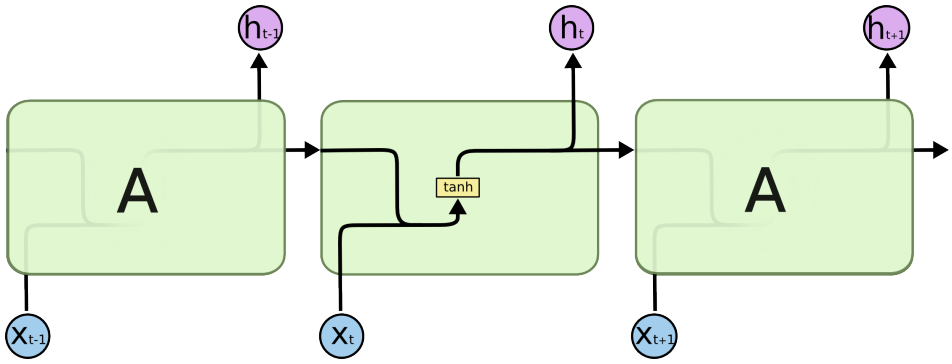

Figure 17: Representation of a RNN cell from [50].

$$h(t) = tanh\left(Ux(t) + Wh(t-1) + b\right) \tag{26}$$

The architecture of an LSTM cell is illustrated in Figure 18. LSTMs are capable of learning long-term dependencies by maintaining a cell state C_t, which serves as long-term memory. This memory is regulated by three gates: the input gate i_t, forget gate f_t, and output gate o_t. The output of the LSTM at time step t, denoted as h_t, is governed by the set of equations in (27) (from [50]). In these equations, $\tilde{C}_t$ represents the candidate cell state, which contributes to updating C_t based on the input gate's activation. The functions σ and tanh denote the sigmoid and hyperbolic tangent activation functions, respectively. The weight matrices W_f, W_i, W_o, and W_C correspond to the forget gate, input gate, output gate, and candidate cell state, while b_f, b_i, b_o, and b_C are their associated bias terms. Similar formulations can be derived for GRU and other LSTM variants.

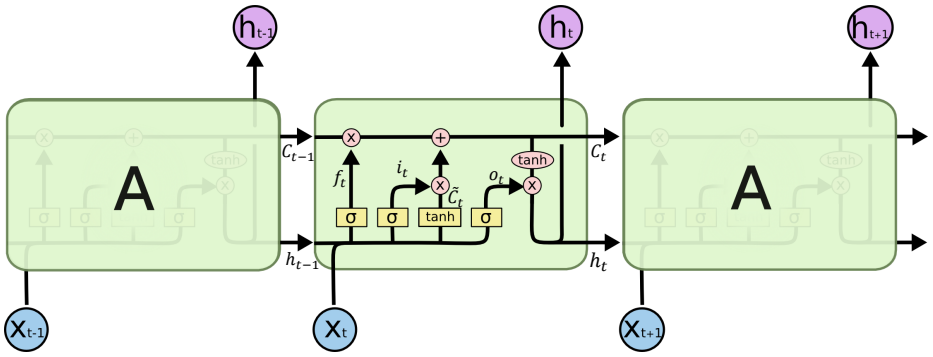

Figure 18: Representation of a LSTM cell from [50].

$$\begin{cases} f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \\ i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \\ \tilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C) \\ C_t = f_t C_{t-1} + i_t \tilde{C}_t \\ o_t = \sigma(W_o[h_{t-1,x_t}] + b_o) \\ h_t = o_t tanh(C_t) \end{cases} \tag{27}$$

Implementations from literature

This box summarizes key studies from the literature on the application of recurrent neural networks for NILM

In [35], the authors compared two NILM methods: a method based on combinatorial optimization with a genetic algorithm and a method based on an RNN. According to the authors, the RNN performed better than the optimization method.

In [51], the authors developed a multilabel classification LSTM autoencoder. In multilabel classification, the algorithm can assign one input to multiple classes. The labels are non-exclusive. Here, the input is a sequence of aggregated power, and the output is a vector of labels the same size as the number of appliances. If an appliance is ON, the corresponding element in the vector will be equal to 1; if the appliance is OFF, it will be equal to 0. The authors argue that this technique is truly non-intrusive because only the aggregated power and the state of the appliances (ON or OFF) are needed for the training of the algorithm. There is no need for a dataset with individual appliance consumption.

In [52], the authors developed a hybrid deep learning model with LSTM and CNN in parallel for load classification. The authors state that this hybrid architecture allows the use of the temporal and spatial characteristics of the consumption data.

In [28], the authors proposed an event-based load transient classification method. The study revealed that multivariate LSTM, with a sequence of active and reactive power as input, outperforms a similar model with only active power as input.

In [53], the authors presented a NILM classification method based on LSTM with a novel signature that emphasizes power variation to improve the classification of multistate appliances.

3.5.3 Convolutional neural network

Convolutional Neural Networks (CNNs) are a specialized type of deep neural network originally designed for computer vision tasks. Computer vision is a field of artificial intelligence that enables computers to interpret and analyze visual data, such as images and videos, by mimicking human vision. CNNs have achieved remarkable success in this domain and have also been effectively applied to other fields, such as time series analysis [54].To understand CNNs and their advantages over classical neural network architectures, the following paragraphs explain how a CNN processes an image.

An image can be represented as a matrix of numbers, with color images requiring three matrices, one for each primary color channel. A naive approach would be to flatten these matrices into a vector and input it into a traditional neural network architecture composed only of fully connected (dense) layers. However, this method performs poorly because it disregards the spatial relationships between pixels, which are crucial for capturing patterns and structures within an image. CNNs address this limitation by leveraging convolutional layers that preserve spatial hierarchies. A typical CNN consists of multiple convolutional layers, followed by fully connected layers, before producing the final output. A convolutional layer applies small matrices called kernels to the input image, scanning it spatially to detect patterns such as edges, textures, or more complex structures. Each kernel extracts specific features by performing element-wise multiplications and summing the results, producing a feature map that highlights relevant information while reducing the number of parameters compared to a fully connected layer. The architecture of a CNN is illustrated in Figure 19.

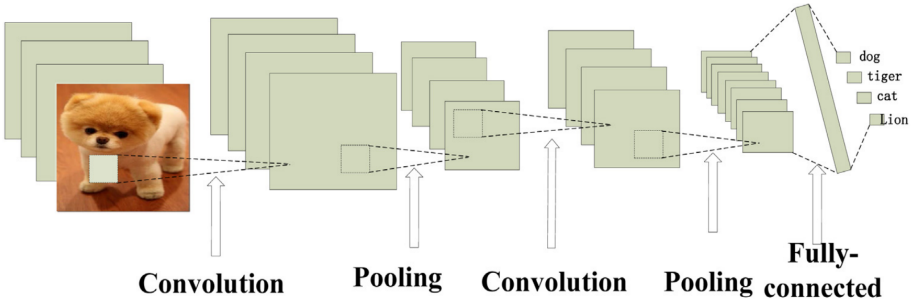

Figure 19: General architecture of a CNN from [52].

Figure 20 demonstrates the convolution operation. In this example, the input consists of three matrices, known as channels. The convolution operation involves sliding kernels over the input, computing element-wise multiplications, and summing the results. The output is the sum of the convolutions between each channel and its corresponding kernel, plus a bias term. The kernels and biases, collectively known as filters, are learned during training. Multiple filters can be applied in each layer, enabling the network to extract different features and increase the model's representation capacity.

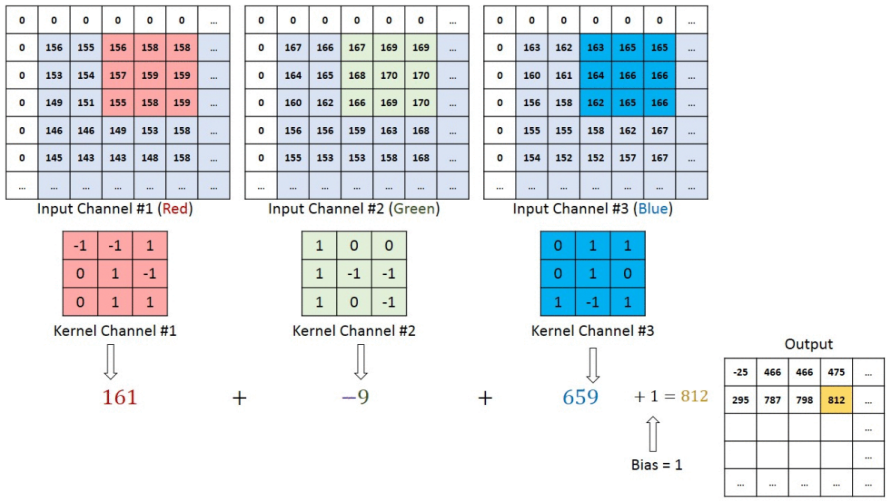

Figure 20: Convolution in a CNN from [55].

After a convolution layer, it is very usual to have a pooling layer. The goal of the pooling layer is to reduce the dimension of the convolutional layer output. The reduction of the output reduces the number of parameters in the network, improving the training of these parameters and helping the network to focus on important features. In Figure 21, max-pooling and average pooling are explained.

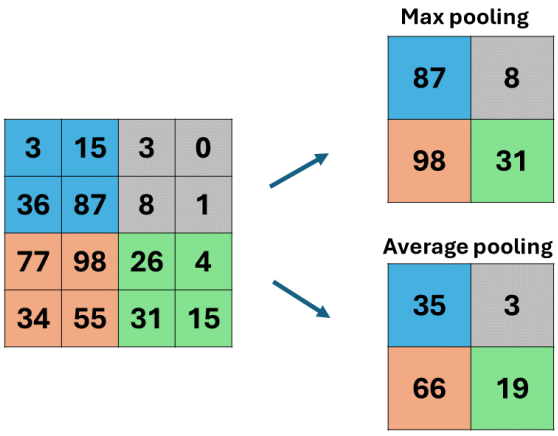

Figure 21: Max pooling and average pooling.

In the context of NILM, CNNs can be utilized for processing time series data, such as active power measurements over time, or for image recognition in VI-trajectory-based methods (see Section 3.6).

Implementations from literature

This box summarizes key studies from the literature on the use of convolutional neural networks in NILM applications.

In [56], the authors introduced a sequence-to-point (S2P) CNN architecture for energy disaggregation and compared it to a sequence-to-sequence (S2S) architecture. In the S2S approach, the input is a sequence of aggregated power readings, and the output is a corresponding sequence of predicted appliance power. These input sequences are generated using a sliding window, which causes each output point to be predicted multiple times. The final power estimate for an appliance is obtained by averaging these overlapping predictions, which results in smoothed edges and may reduce sharp transitions in power consumption. In contrast, the proposed S2P architecture predicts only a single appliance power value, the one corresponding to the midpoint of the input sequence. This approach leverages both past and future aggregated consumption data, allowing for a more accurate estimation at the central time step and leading to improved disaggregation performance.

In [57], the authors investigated two architectures, a single-target and a multi-target point-to-point convolutional neural network for NILM. The models are called point-to-point because they estimate the appliance powers at one time step with aggregated features (P, Q, S, PF, etc.) at one time step. In this study, the general performances of the multi-target model are lower than those of the single-target model.

In [58], the authors proposed an event-based, supervised disaggregation method based on active power transients. The method operates as follows: first, an event is detected when a change in active power exceeds a specified threshold. Then, a 6-second window of the active power transient is extracted at 100 Hz. This window is used as input for a sequence-to-point CNN classifier. A different CNN is employed for each targeted appliance to classify whether the appliance is running or not. Finally, the consumption of the detected appliance is calculated

using a heuristic algorithm, assuming the appliance's power consumption remains constant during its operation. This power is estimated as the difference in power before and after the switching event.

In [59], the authors proposed enhancements to the original sequence-to-point method using a "temporal" CNN. The method was improved with three key elements: First, causal convolution ensures that the output at time t does not depend on the input at time $t+1$ or later. Second, dilated convolution expands the kernel by adding "holes" between elements, allowing the model to capture information over longer time series, thus enlarging the receptive field. Third, the addition of residual connections helps mitigate the gradient vanishing or exploding problem.

In [60], the authors also proposed to improve the original sequence-to-point method with dilated convolution and residual connection as in [59] but this time with noncausal convolution.

In [61], the authors proposed a sequence-to-point CNN for HVAC load disaggregation. The main particularity of this proposed algorithm is the two-dimensional input composed of the sequence of normalized temperature and the sequence of aggregated active power.

In [62], the authors developed a lightweight sequence-to-sequence CNN for NILM. The model is computationally efficient, consisting of only two CNN layers and one dense layer. This lightweight design allows deployment on edge devices (small, low-power computing units that process data locally rather than relying on cloud computing). By running directly on edge devices, the model enhances privacy by keeping user data on-site rather than transmitting it to external servers.

In [63], the authors compared two sequence-to-point models for energy disaggregation—one using a convolutional neural network (CNN) and the other based on gated recurrent units (GRUs). Unlike traditional sequence-to-point architectures, where the predicted output corresponds to the midpoint of the input sequence, the proposed models predict the appliance power at the endpoint of the sequence. This design enables real-time processing, as the model can generate predictions immediately after receiving the full input sequence, without waiting for future data.

3.5.4 Autoencoder

An AutoEncoder (AE) is a type of neural network architecture where the input layer and the output layer have the same number of neurons, i.e., the same dimension. Autoencoders are trained to recreate a copy of the input at the output. A representation of this kind of network is shown in Figure 22. The layers can be fully connected or convolutional. The number of neurons per layer decreases to a bottleneck and then increases again so that the output matches the input dimension. An autoencoder consists of two parts: the encoder, which compresses the input data into a lower-dimensional space, and the decoder, which takes the compressed data and attempts to reconstruct the input. Because the network aims to reproduce a copy of the input at the output, it can be trained using unsupervised learning. The goal of an AE is to transform the input data into a compressed or latent representation that contains the most useful information. A denoising AutoEncoder (dAE) is a variant of an AE trained to receive a corrupted data version as input and output the original version.

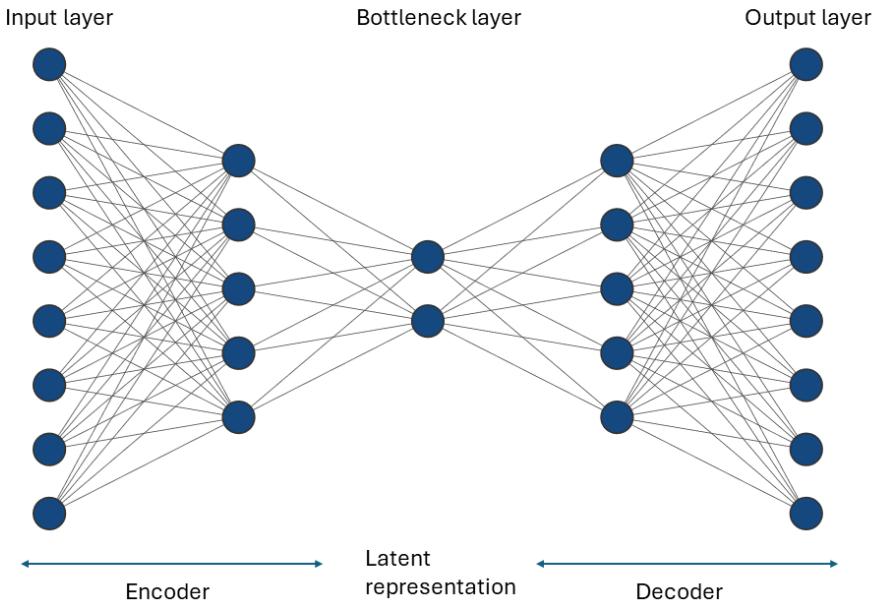

Figure 22: Representation of an autoencoder (created with [48]).

A Variational Autoencoder (VAE) is an extension of the classic Autoencoder (AE) that introduces probabilistic modeling into the latent space. In a standard AE, the input is encoded into a fixed vector at the bottleneck layer, which is then decoded back to reconstruct the original input. However, in a VAE, instead of encoding the input into a single deterministic vector, it is mapped into a probability distribution characterized by a mean vector $\boldsymbol{\mu}$ and a standard deviation vector $\boldsymbol{\sigma}$. This allows the model to learn a more structured and continuous latent space, which is beneficial for generating diverse and realistic reconstructions. Unlike traditional AEs, where the latent representation is a single point, VAEs introduce stochasticity by sampling a latent vector $\mathbf{z}$ from the learned distribution before passing it to the decoder. This sampling operation is defined by the reparameterization trick, given in equation (28):

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon, \quad \text{where } \epsilon \sim \mathcal{N}(0, 1) \tag{28}$$

A representation of the VAE architecture is illustrated in Figure 23. The authors in [64] state that the probabilistic nature of VAE and its regularized latent space improve the encoding of relevant information, which helps generate more accurate complex load profiles like the profile of a multi-state appliance, and improve the generalization capabilities of the model.

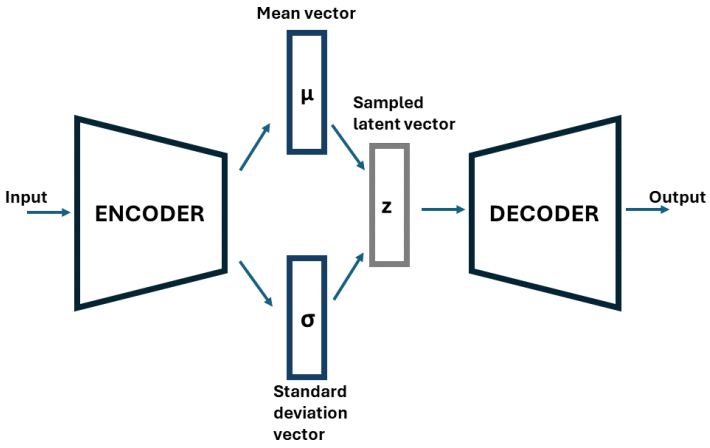

Figure 23: Representation of a variational autoencoder.

Implementations from literature

This box presents representative studies from the literature on the use of autoencoders for NILM.

In [65], the authors compared three deep learning architectures for the estimation of appliance power: an LSTM, a denoising autoencoder, and a deep fully connected neural network that regresses the start time, the end time, and the appliance average power. In the paper, the dAE and DNN seem to perform better than the LSTM.

In [66], the authors proposed a denoising autoencoder that inputs active and reactive power sequences and outputs the disaggregated power of the appliances. In this model, the encoder and decoder are composed of convolutional layers.

In [67], the authors introduced a variational autoencoder (VAE) with convolutional layers for energy disaggregation. The model takes a sequence of aggregated power as input and predicts a sequence of the target appliance's power as output. Their model demonstrated superior performance compared to other disaggregation methods, including the classic Denoising Autoencoder (DAE), Additive Factorial Hidden Markov Model (AFHMM), Sequence-to-Sequence (S2S), and Sequence-to-Point (S2P) architectures.

Building on this work, [64] proposed an enhanced VAE for NILM (see Section 6 and Figure 40 for details). Their improvements led to a significant performance boost, outperforming the VAE proposed in [67].

3.5.5 Generative adversarial network

A Generative Adversarial Network (GAN) is a type of machine learning framework that consists of two competing neural networks: the generator and the discriminator. The generator's goal is to create artificial data that resembles the training data, taking random noise as input. The discriminator's goal is to distinguish between real data from the training set and artificial data created by the generator. It acts as a classifier, taking both real and artificial data as input. The two networks are in competition:

the generator tries to increase the discriminator's error rate, while the discriminator tries to reduce its own error rate. The core concept of GANs is the indirect training (or fine-tuning) of the generator through the feedback from the discriminator. A representation of the GAN concept is shown in Figure 24.

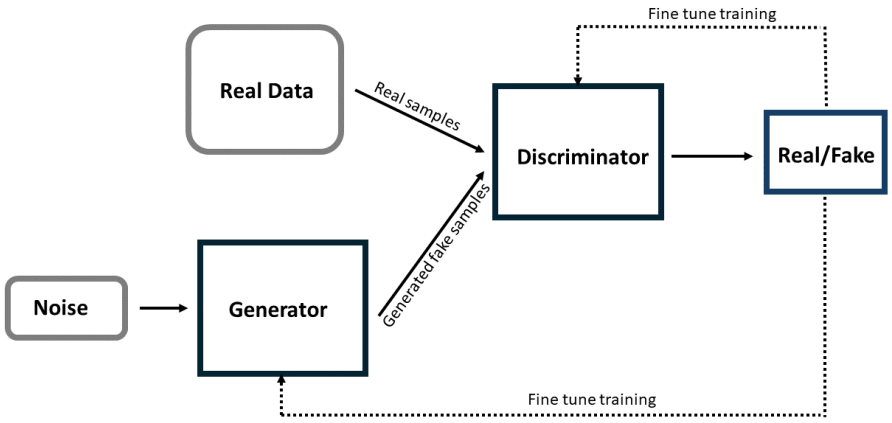

Figure 24: Representation of the GAN concept.

In [68], the authors proposed a simple GAN-based energy disaggregation algorithm called EnerGAN to evaluate the effectiveness of GANs in NILM. The GAN architecture consists of three components: a seeder, a generator, and a discriminator, as shown in Figure 25. The seeder and generator are composed of convolutional layers and together form an autoencoder. The seeder takes the aggregated power as input, and the generator outputs the estimated power of a single appliance. The discriminator takes a pair of sequences as input: the aggregated power and the disaggregated power of one appliance. The disaggregated power can be either the ground truth or the power generated by the generator. The discriminator's task is to determine whether the pair is real or generated, producing a binary output.

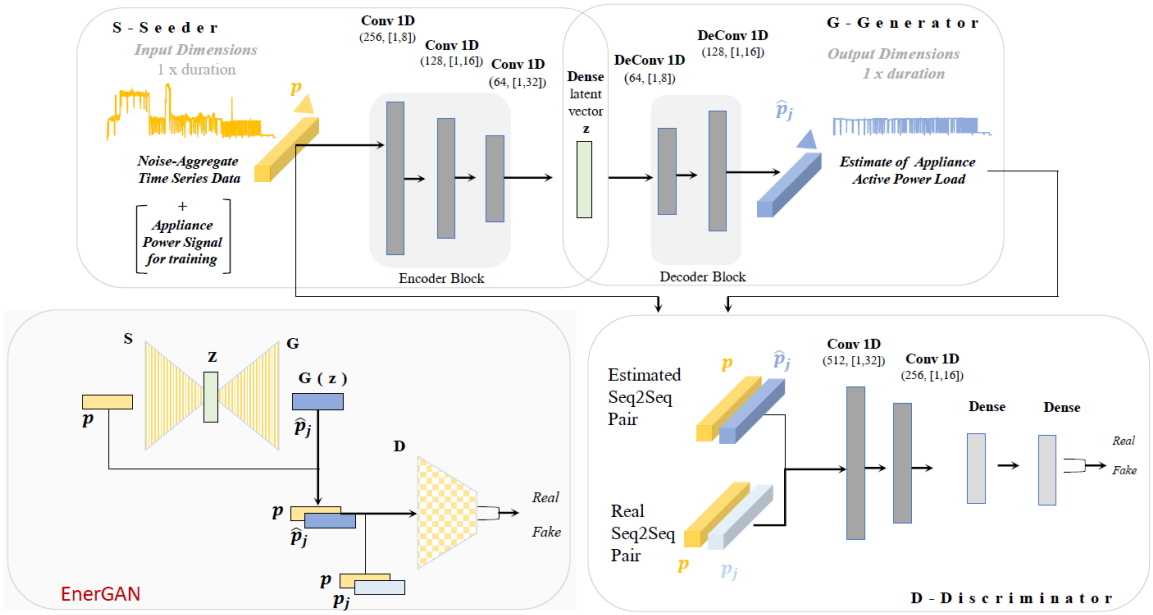

Figure 25: Architecture of EnerGAN from [68].

Implementations from literature

This box illustrates key instances from the literature on the application of generative adversarial networks for NILM.

In [69], the same authors as in [68] proposed an improved implementation of EnerGAN with a novel discriminator. The discriminator is a combination of convolutional layers and GRU layers. The GRU layers are supposed to add recurrent capabilities to the discriminator. Their experimental results indicate superior performance of this novel implementation over the state-of-the-art algorithms.

In [70], the authors discussed the uses of transfer learning to improve the generalization of GAN for energy disaggregation.

In [71], the authors used adversarial loss to improve the training and consequently the performance of their deep learning model.

3.5.6 Domain Adversarial Neural Networks

Domain Adversarial Neural Networks (DANN) is a semi-supervised deep learning method. It uses labeled as well as unlabeled datasets. Its main interest is the generalization of deep learning algorithms to new environments. Indeed, for instance, a NILM deep learning algorithm trained on a USA dataset will probably perform poorly on European data. DANN can be used to generalize the algorithm to European data using only unlabeled European data that is easily available.

In [72], the authors introduced a Domain Adversarial Neural Network (DANN) to enhance the generalization capability of NILM algorithms. Their approach leverages both labeled and unlabeled

data to improve performance across different datasets. Specifically, they trained their model using labeled data from the USA (REDD dataset) and unlabeled data from Europe (UKDALE dataset[4]). The results demonstrated that their approach outperformed a model trained solely on labeled data from the USA when tested on a European dataset, highlighting the effectiveness of domain adaptation in NILM. The proposed DANN is composed of three different neural networks: a Feature Extractor (FE) that processes the input aggregated power sequence and extracts meaningful patterns, a Classifier (C) that takes the extracted features from the FE and predicts the appliance state (e.g., ON, OFF, or other states), and a Discriminator (D) designed to distinguish whether a given feature representation originates from the source domain or the target domain. The DANN training process consists of three main steps:

1. Training the Feature Extractor and Classifier: Initially, the FE and C are trained using labeled data from the source domain. The input is a sequence of aggregated power, and the classifier predicts the appliance state.

2. Training the Discriminator: Next, the discriminator is trained while keeping the FE weights fixed. The discriminator is fed both source and target domain data and learns to classify whether a given feature vector comes from the source or target domain. This step does not require appliance labels; only domain labels (source or target) are used.

3. Adapting the Feature Extractor: Finally, the FE is updated while keeping the discriminator fixed. The source and target domain labels are inverted, forcing the FE to extract features that make it difficult for the discriminator to distinguish between domains. This encourages the FE to learn domain-invariant features that focus only on appliance classification rather than dataset-specific differences.

These three steps are repeated iteratively until convergence, as illustrated in Figure 26. By aligning feature distributions across domains, the DANN framework enhances generalization, enabling NILM models to effectively utilize both labeled and unlabeled data. This approach not only improves disaggregation accuracy but also makes better use of available real-world datasets, overcoming the challenge of limited labeled data.

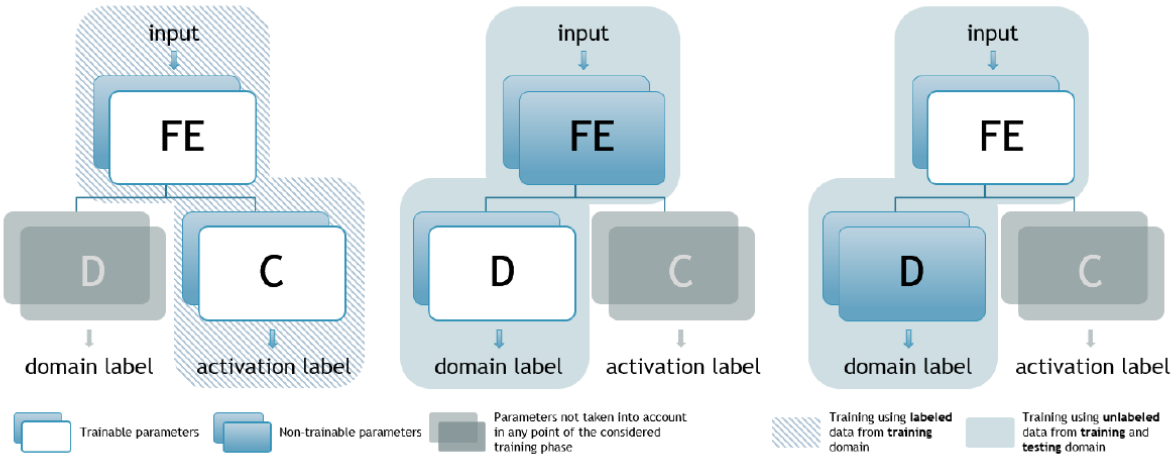

Figure 26: DANN training process from [72].

[4]REDD and UKDALE datasets are described in section 4.1

3.5.7 Transformers and attention mechanism

Transformers are a deep learning architecture introduced in 2017 in the seminal paper "Attention is All You Need" by Vaswani et al. [73]. Their key advantage over traditional recurrent architectures is their ability to process data in parallel rather than sequentially, leading to significantly faster training and inference. Transformers can efficiently handle larger datasets and capture long-term dependencies more effectively than classical RNNs, including LSTMs and GRUs. These advantages make them particularly promising for NILM, where identifying appliance patterns over long time spans is crucial. The superior performance of transformers stems from four core components: word embedding, self-attention, positional encoding, and residual connections. These concepts are detailed below in the context of NILM.

- Word embedding: Word embedding maps categorical data (such as words in natural language processing) into a high-dimensional numerical space. In NILM, a similar approach can be used to map power sequences into a feature space, enabling the model to learn meaningful representations of appliance consumption patterns (as in [74]).
- Self-Attention: Self-attention computes attention scores for each pair of elements in a sequence, allowing the model to capture complex, long-range dependencies. This is particularly useful in NILM, where appliance usage may exhibit temporal dependencies over minutes or even hours. By weighting the importance of different time steps, self-attention enables the model to focus on relevant consumption patterns while ignoring irrelevant fluctuations.
- Positional Encoding: Since transformers do not inherently capture the order of elements in a sequence, positional encoding is used to inject information about the relative position of each timestep in the input power sequence. This ensures that the model correctly interprets time-dependent patterns, which are critical for accurately disaggregating energy consumption.
- Residual Connections: Unlike traditional neural networks, where each layer only receives input from the previous layer, residual connections allow information to bypass multiple layers, facilitating better gradient flow. This prevents vanishing or exploding gradients, allowing the construction of deeper and more expressive models. In NILM, this helps preserve fine-grained appliance features across multiple layers, improving model accuracy.

By leveraging these components, transformers can offer state-of-the-art performance in NILM by effectively modeling long-term dependencies, handling large datasets, and capturing complex appliance interactions.

Implementations from literature

This box summarizes key studies from the literature on the application of transformers for NILM

In [71], the authors proposed a deep learning model based on CNN with an attention module to consider global context and improve performance.

In [75], the authors developed a sequence-to-sequence CNN with an attention layer after the convolutional layers and before the fully connected layers. The authors claim that introducing the attention layer improves the model's performance.

In [76], the authors used positional encoding (in particular, cosine positional encoding) on the aggregated active power input sequence to use the timing information in the neural network architecture.

In [74], the authors developed a sequence-to-point deep learning energy disaggregation model

based on a transformer architecture with word embedding, positional encoding, residual connection, and attention mechanism.

In [77], the authors proposed a Bi-directional Temporal Convolutional Network (BiTCN) improved with an attention mechanism. The TCN allows the network to capture long-term dependencies thanks to dilated convolution. The bi-directional aspect allows the processing of data forward and backward to efficiently use past and future contexts.

In [78], the authors proposed a Switch Transformer model for NILM (STNILM). The performance of this variant of a classic transformer is assessed in the paper.

3.5.8 Transfer Learning

Transfer learning is a powerful technique in machine learning, particularly in deep learning, that aims to reduce the computational cost of training or to achieve high performance with limited training data. The core idea is to leverage a model that has already been trained on a similar task and retrain only a subset of its parameters (such as the final layers) for the target task.

For instance, in [79], the authors utilized the pre-trained AlexNet convolutional neural network to classify VI trajectory (see section 3.6) colored images into appliance categories. AlexNet, an efficient CNN trained on millions of images across thousands of categories, had its last few fully connected layers replaced with new ones. The network was then retrained for VI trajectory classification, using a high learning rate for the new layers and a slower rate for the transferred layers. This approach allowed the model to efficiently adapt to the new task with high performance.

3.6 VI-trajectory

The voltage-current (VI) trajectory is a feature extracted from aggregated measurements of instantaneous voltage and current in event-based NILM methods. It is a plot or 2D image representing the voltage versus current during an appliance's steady-state operation, used for appliance classification. The basic steps of a VI trajectory method are as follows:

First, an event (such as the switching of an appliance) is detected in the aggregated signal. Then, voltage and current data are collected for multiple cycles before and after the transient state, covering the appliance's OFF state and steady-state ON operation. The VI trajectory method requires high-frequency measurements, typically above 1 kHz, to accurately capture the current and voltage waveforms. Afterward, the voltage and current data are pre-processed to isolate the specific waveforms of the appliance switched ON. The current and voltage are often pre-processed with equation (29) and (30) like in [39], [79] or [43]. In the equations, i_{before} and v_{before} are the current and voltage before the event. i_{after} and v_{after} are the current and voltage after the transient state when the appliance is in steady state. In these equations, the signals must have the same initial phase angle.

$$i = i_{after} - i_{before} \tag{29}$$

$$v = v_{after} \text{ or } v = \frac{v_{after} + v_{before}}{2} \tag{30}$$

Once pre-processed, the VI curve is plotted and often transformed into a matrix form. This matrix can be composed of binary elements or be more complex to include additional information about the appliance. The VI trajectory can then be used in various algorithms to classify the appliance responsible for the event. Classification of VI trajectories can be approached as an image recognition problem using

convolutional neural networks, or different features can be extracted from the VI trajectory and used for classification with other algorithms. Figure 27 shows examples of VI trajectories for six different appliances from the REDD dataset from [9].

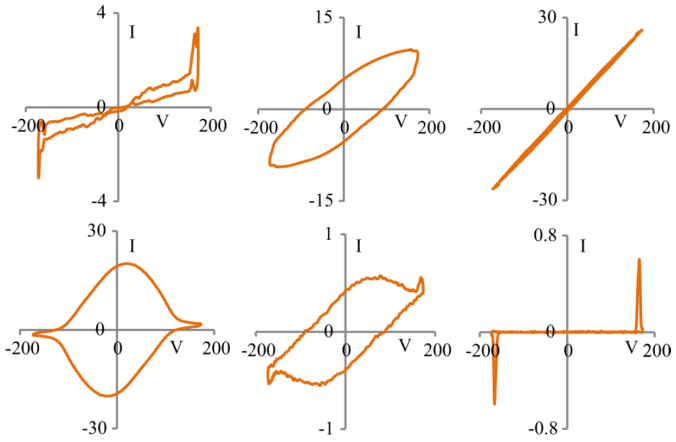

Figure 27: VI trajectories from six different appliances from the REDD dataset from [9].

Implementations from literature

This box presents representative VI-trajectory-based NILM methods from the literature.

In [9], the authors compared four classification algorithms using wave-shape features extracted from the VI trajectory of an event, such as looping direction, enclosed area, and the number of self-intersections, as input. These algorithms included an artificial neural network, an ANN coupled with an evolutionary algorithm, a support vector machine, and the AdaBoost algorithm.

In [39], the authors proposed a method where ten features are extracted from a VI trajectory plot, such as the area enclosed by the trajectory, direction of trajectory, current span, self-intersection, etc. An SVM classification algorithm is then used to identify the load thanks to the features extracted.

In [79], the authors proposed transforming the VI trajectories into color images using the HSV color space (The HSV color space represents colors using three components: Hue, Saturation, and Value). This transformation aims to add information about trajectory direction, repeatability, and active and reactive power ratio to the VI trajectory. After the HSV transformation, a CNN is used to classify the colored images into appliances. In this paper, the authors also used transfer learning; the CNN was pre-trained on ImageNet, a dataset with millions of images. Then, it was trained again with VI trajectory images.

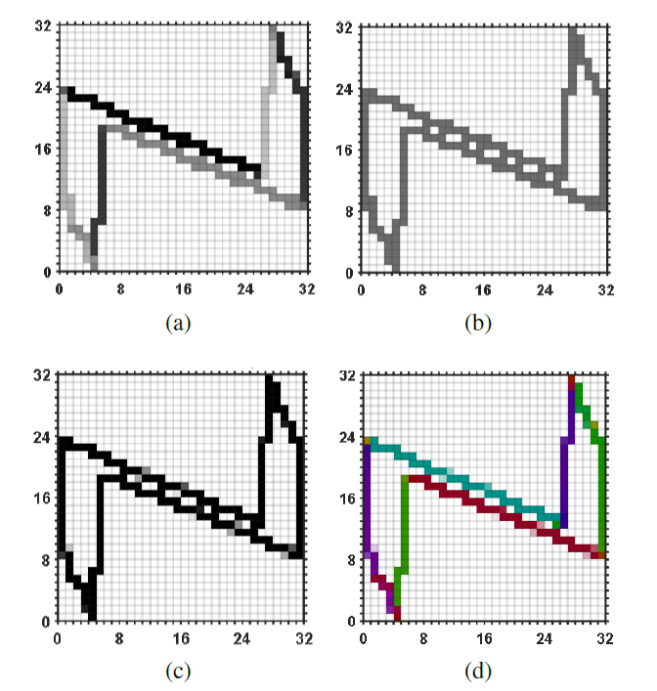

Figure 28: VI trajectory (a) Hue, (b) Saturation, (c) Value, (d) RGB representation. From [79].

In [43], the authors compared two algorithms to classify VI trajectories into appliances. A CNN is used on VI trajectory images, and a Random forest is used on elliptical Fourier descriptors that characterize the contour of the trajectory.

In [80], the authors proposed representing VI trajectories as weighted, pixelated images and using a convolutional neural network to classify them.

In [81], the authors proposed a color encoding for VI trajectories where information about the motion, momentum, and power is added to the image. The VI trajectory image is then transformed into a feature vector and input to the gcforest algorithm. gcforest algorithm is an ensemble learning method used for classification.

3.7 Others

In the following, we present some lesser-known methods and tools for NILM. While not as popular, they are included to illustrate the diversity of approaches available to tackle the NILM problem.

3.7.1 Dynamic Time warping

Dynamic Time Warping (DTW) is a technique used to align and compare two time series of potentially different lengths. DTW works by finding the optimal alignment between two time series and determining which points of the first time series correspond to which points of the second time series. This alignment minimizes the overall distance between the time series, allowing for a meaningful comparison. DTW also provides a measure of similarity between the two time series based on the cost of this optimal alignment. The mathematical developments of this section are inspired by [82]. Figure

29 gives the alignment of two time series with DTW; the aligned points are indicated by the arrows. With DTW, one point from one series can be aligned with multiple consecutive points of the other time series.

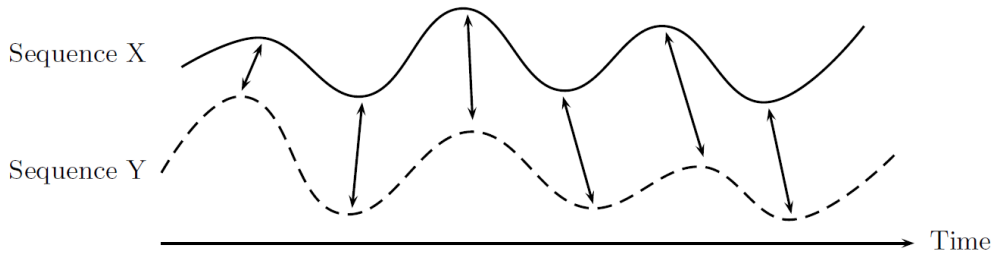

Figure 29: Alignment of two sequences with DTW from [82].

Figure 30 from [82] gives the cost matrix of two sequences, X and Y. The cost between two points is the absolute value of their difference. A high difference is indicated by a light color, and a low difference by a dark color.

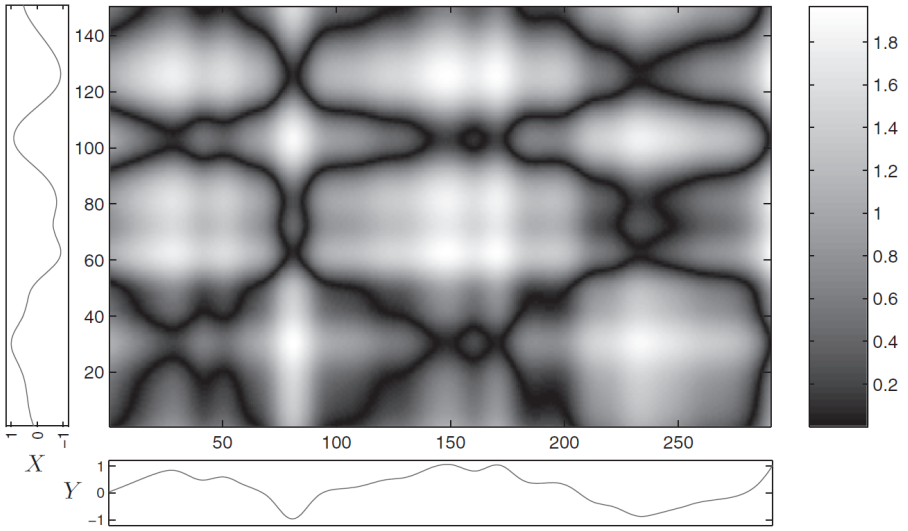

Figure 30: Cost matrix of sequences X and Y with the absolute difference from [82].

To find the optimal alignment between two series, it is necessary to create the accumulated cost matrix D. The accumulated cost matrix $D \in \Re^{(NxM)}$ of time series X with N elements and time series Y with M elements is calculated as follows: First elements of the matrix D are initialized with equation (31).

$$\begin{aligned} D_{i,0} &= \infty \\ D_{0,j} &= \infty \\ D_{0,0} &= 0 \end{aligned} \tag{31}$$

Then the matrix can be filled for i=1 to N and for j=1 to M with equation (32).

$$D_{i,j} = d(A_i, B_j) + min \begin{cases} D_{i-1,j-1} \ (match) \\ D_{i-1,j} \ (insertion) \\ D_{i,j-1} \ (deletion) \end{cases} \tag{32}$$

where $d(A_i, B_j)$ is a measure of distance between A_i and B_j. This distance can be equal to the absolute difference given in equation (33).

$$d(A_i, B_j) = |A_i - B_j| \tag{33}$$

The best alignment between X and Y, i.e., the alignment with the lowest overall cost, can be found thanks to the matrix D with the following algorithm. Starting from $D_{N,M}$, the adjacent element of the matrix ($D_{i-1,j-1}$, $D_{i-1,j}$, $D_{i,j-1}$) with lowest cost is selected each time until reaching $D_{1,1}$. This method always gives the best correspondence between X and Y and the lowest overall alignment cost. The alignment cost is given by the value of $D_{N,M}$. Figure 31 illustrates the best alignment between two time series X and Y.

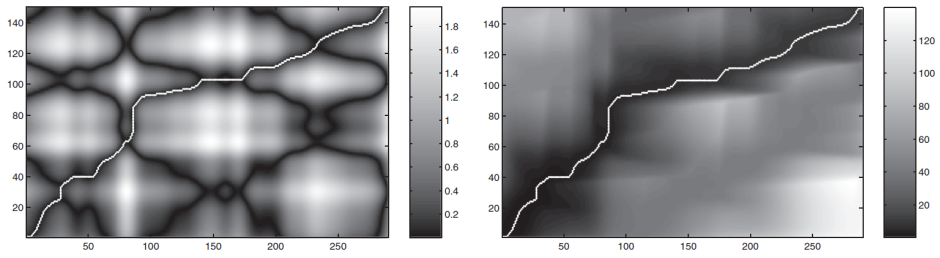

Figure 31: Optimal alignment path (i.e, white line) on the cost matrix (left) and the accumulated cost matrix D (right) from [82].

In the context of NILM, dynamic time warping provides a similarity metric between time series that may differ in length or temporal alignment. This distance measure can be effectively used in classification algorithms to compare, for example, the power signature of an observed event with a set of known appliance signatures. Unlike traditional distance metrics such as Euclidean distance, DTW aligns sequences by matching their overall shape rather than relying on exact time-aligned points. DTW can lead to improved classification accuracy, particularly when appliance cycles exhibit variability in duration or timing.

Implementations from literature

This box illustrates key instances of NILM methods using DTW from the literature.

In [41], the authors developed an unsupervised, event-based NILM algorithm using Dynamic Time Warping (DTW) based on active power at a low sampling rate (less than 1 Hz). After detecting and extracting an event, its signature (the aggregated active power) is compared with known signatures from a library using DTW. The event is then associated with the closest signature in the library, i.e., the one with the lowest distance cost.

In [83], the authors compared different distance metrics, including DTW, for clustering appliances with K-nearest neighbors classifiers.

In [28], the authors used the dynamic time warping distance to cluster transient signal appliances thanks to a DBSCAN algorithm (clustering algorithm).

In [84], the authors used DTW for detecting the introduction of a novel appliance in a building without prior information about the novel appliance.

In [85], the authors proposed an event-based NILM method where the transient power waveforms of loads are identified with the nearest neighbors clustering algorithm using dynamic time warping as distance metrics.

3.7.2 Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique. The goal of PCA is to reduce the number of variables or parameters characterizing a set of data by creating new variables. They are mutually orthogonal, hence statistically independent, and are designed to capture as much information (i.e., variance) from the original variables as possible. The components are ranked according to the amount of variance they explain. A difference of the same magnitude in the first principal component reflects a greater distinction between two data points than an equal difference in the second component. The principal components form a set of vectors. The ith vector is the vector that best fits the Data (captures the most variance) while being orthogonal to the i-1 vector. The concept of PCA is illustrated in Figure 32. The steps for calculating the principal component from a dataset are briefly explained below. The mathematical developments are from [86]. Considering a dataset with n points. Each point is described by p variables. The data can be represented as a set of n vectors x_1,...x_n each having p elements.

1. The mean of each variable is calculated with equation (34).

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{n} x_i \tag{34}$$

2. The covariance matrix is calculated with equation 35.

$$S = \frac{1}{N}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T \tag{35}$$

3. The eigenvalues and eigenvectors of the covariance matrix S are calculated.

4. The eigenvalues and eigenvectors are sorted in order of decreasing eigenvalues. The first l eigenvectors are the first l principal components and can be used to project the data points into a space with $l < p$ dimensions.

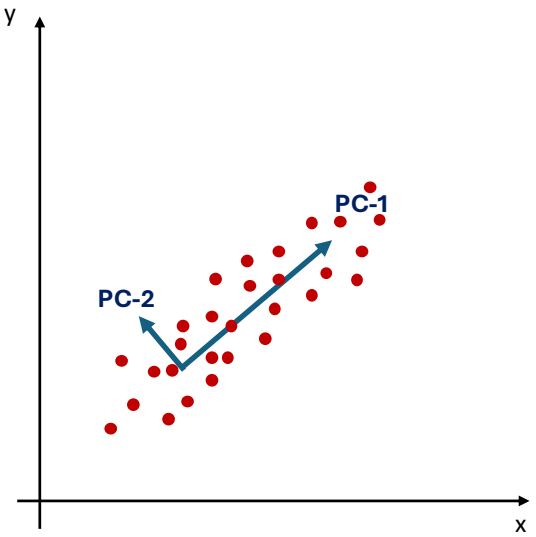

Figure 32: Representation of the principal component analysis concept.

Implementations from literature

This box illustrates key instances of NILM methods using PCA from the literature.

In [87], the authors propose an event-based appliance classification method that utilizes Principal Component Analysis (PCA) on active, reactive, and distortion power signals to extract distinguishing features.

In [88], the authors proposed to use PCA to reduce the dimension of the power features of classification algorithms in order to improve computational efficiency. In the paper, the authors demonstrated that with the first principal components calculated from features like active, reactive, and apparent power, current, and harmonics, the efficiency of the classification algorithms can be improved while the accuracy is preserved. The classification algorithms used in the experiments were combinatorial optimization and modified cross-entropy.

3.7.3 Sparse Dictionary Learning or sparse coding

Sparse Dictionary Learning (SDL), also known as sparse coding, is an unsupervised signal processing technique that seeks to represent signals as sparse linear combinations of basis functions (atoms) drawn from a dictionary. In this framework, a signal y is approximated as $y = Dy$, where D is the dictionary matrix and x is a sparse coefficient vector in which most entries are zero. The dictionary D can either be predefined or learned from training data, while the sparse representation x is typically found by solving an optimization problem with sparsity-inducing regularization, such as l_1 penalty. This sparsity ensures that only a few dictionary atoms contribute significantly to the reconstruction of the signal. More details about the mathematical background can be found in [89].

Implementations from literature

This box illustrates key instances of NILM methods using SDL from the literature.

In the context of energy disaggregation, [90] proposes a novel discriminative extension of sparse coding to address the challenge of separating whole-home energy usage into its constituent appliance-level signals using low-frequency smart meter data. Their approach first learns individual dictionaries for each appliance type using plug-level measurements. These appliance-specific dictionaries are then combined and used to disaggregate aggregate consumption signals from unseen homes. The authors introduce a discriminative training procedure called Discriminative Disaggregation Sparse Coding (DDSC), significantly improving performance over traditional sparse coding approaches.

3.7.4 Graph signal processing

Graph Signal Processing (GSP) is a field of study that analyzes signals by associating them with a graph composed of nodes and edges. Each node in the graph represents an element of the dataset, and the edge weight between two nodes represents the similarity between these two elements. In the context of NILM, GSP treats the power consumption data of a household as a signal over a graph, where each node corresponds to a power measurement at a specific time, and the edges represent the relationship (similarity) between these measurements. One main advantage of GSP is its lower computational complexity compared to classical state-based approaches like hidden Markov models. An example of the mathematical development for energy disaggregation with supervised GSP is briefly presented below from [91].

Considering a household with M appliances and the following notation. $p(i)$ the aggregated active power at time i, $p_m(i)$ the active power of appliance m at time i. The change in the active power signals $\Delta p(i) = p(i+1) - p(i)$ and $\Delta p_m(i) = p_m(i+1) - p_m(i)$. Moreover, supposing that for $I \leq n < N$, the appliance's power $p_m(i)$ is known. The task is to find $p_m(i)$ for $n < i < N$. The graph signal s^m is a vector defined by 36 with Thr_m a threshold.

$$s^m = \begin{cases} +1 \; for \; |\Delta p_m(i)| \geq Thr_m \; and \; i \leq n \\ -1 \; for \; |\Delta p_m(i)| < Thr_m \; and \; i \leq n \\ 0 \; for \; i > n \end{cases} \tag{36}$$

This graph signal is indexed by a graph $G_m = (V_m, A_m)$ with V_m the set of vertices (nodes) and A_m the adjacency matrix. Where the node of V_m are defined with $v_m(i) = \Delta p_m(i)$. The weight of the edge between two nodes v_i and v_j reflect their similarity and can be defined thanks to the Gaussian kernel weighting function (equation (37)) like in [91], [92], [93], [94], [95]. Where σ is a scaling factor.

$$A_{i,j} = exp\Big\{-\frac{(v_i - v_j)^2}{\sigma^2}\Big\} \tag{37}$$

Then with D_m a diagonal matrix defined by equation (38) for k=1,...,N.

$$D_m(k,k) \sum_{j=1}^{M} A_m(j,k) \tag{38}$$

And the Laplacian matrix $L_m = D_m - A_m$. The optimization problem can be defined with equation (39) for $i = 1, ..., N$.

$$\underset{p_m(i)}{\text{argmin}} = \|\Delta p(i) - \sum_{m=1}^{M} \Delta p_m(i)\|_2^2 + \lambda \sum_{m=1}^{M} \|s_m^T L_m s_m\|_2^2 \tag{39}$$

The first term in equation (39) minimizes the difference between the aggregated power and the sum of the appliance power, and the second term maximizes the smoothness of the graph signal. The parameter λ is the trade-off between the smoothness and the disaggregation error. The smoothness of the graph signal is maximized because the power of an appliance tends to change smoothly.

Implementations from literature

This box illustrates key instances of NILM methods using GSP from the literature.

In [91], [93], and [94], the authors proposed to use supervised GSP for solving the NILM problem.

Conversely, in [92] and [95], the authors proposed using unsupervised GSP for solving the NILM problem.

3.7.5 Recurrence Graph

A Recurrence Plot (RP) or Recurrence Graph (RG) is a tool used to analyze dynamical systems to visualize the times at which a state of the system recurs. In the context of NILM, this method is similar to VI trajectory (see section 3.6) because it transforms electrical measurements such as current or power into a visual 2D representation that can be used for appliance classification. As VI trajectory methods, RG methods are event-based methods. The basic step of a NILM recurrence graph method is explained in the following paragraph. These steps are inspired from [96], [97], [98]. After the event detection, the current and voltage waveform of the event are extracted with equations (29), 30. They are pre-processed with the Piecewise Aggregate Approximation (PAA)[5] in order to reduce their dimensions to w. Then the binary recurrence matrix $R_{w,w}$ is created with the following equation (40). Where ε is a threshold and $d_{i,j}$ is the Euclidean distance between two points of the pre-processed signal.

$$R_{i,j} = \begin{cases} 0 \; if \; d_{i,j} > \varepsilon \\ 1 \; if \; d_{i,j} < \varepsilon \end{cases} \tag{40}$$

This binary representation can lead to a loss of information. In order to avoid this loss of information, Weighted Recurrence Graphs (WRG) were introduced. In WRG, the recurrence matrix is not limited to binary elements. Once the WRG of an event is created, this 2D representation can be used to classify the event with, for instance, a convolutional neural network.

Implementations from literature

This box illustrates key instances of NILM methods using WRG from the literature.

In [97], the authors propose using weighted recurrence graphs and convolutional neural networks for appliance classification. Their results show that WRG-based representations outperform V-I trajectory images in classification performance.

In [96], the same authors as in [97] developed an Adaptive Weighted Recurrence Graph (AWRG) where the parameters of the WRG are learnable and adjusted by computing the gradient in a similar way the parameters of a neural network are adjusted. The AWRGs

[5] Piecewise Aggregate Approximation (PAA) is a dimensionality reduction technique for time series that divides the series into equal-sized segments and represents each segment by its average value. This results in a lower-dimensional, smoothed version of the original time series.

calculated with the current signal are then classified with a CNN.

In [98], the authors proposed to use the power signal instead of the current signal in weighted recurrence graphs. A CNN is then used to classify different models of EV thanks to the weighted power recurrence graph.

3.7.6 Fourier and Wavelet Transform

The Fourier transform is a mathematical tool that converts a time-domain signal into its frequency-domain representation, making it highly useful for analyzing the frequency content of a signal. In the context of NILM, new features for appliance recognition or appliance power estimation can be developed from the Fourier transform of measured signals like active power, current, or voltage. The Fourier transform of a signal $x(t)$ is expressed in equation (41). Most of the mathematical developments in this section are from [99].

$$FT\{x(t)\} = X(f) = \int_{-\infty}^{+\infty} x(t)e^{-i2\pi ft}\, dt \tag{41}$$

As a signal from a data acquisition system like the smart meter is not continuous but sampled with a specific time interval ΔT, the Discrete Fourier Transform (DFT) of a signal $x[n]$ with N measurements is defined with equation (42).

$$DFT\{x[n]\} = X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{\frac{-i2\pi kn}{N}} \tag{42}$$

Most of the time, as discussed in this document, the Fourier transform is applied to high-frequency measurements of aggregated current to extract current harmonics, which are then used as features for appliance recognition. However, Fourier transforms are not well-suited for signals with time-varying frequency characteristics, such as electrical signals during an appliance's transient state. To address this, the Short-Time Fourier Transform (STFT) is introduced. STFT divides the signal into multiple segments and applies the Fourier transform to each segment, allowing analysis in both the time and frequency domains. The STFT can be expressed by equation (43), where g(t) is a window function (e.g., Gaussian window, Hamming window, Kaiser-Bessel window) and τ is the time shift.

$$STFT\{x(t)\} = X(\tau, f) = \int_{-\infty}^{+\infty} x(t)g(t-\tau)e^{-i2\pi ft}\, dt \tag{43}$$

STFT uses a fixed-size window for all frequencies. This results in a trade-off between time and frequency resolution that is the same across all frequencies. Wavelet Transform (WT) is introduced to overcome this limitation. WT uses variable-sized windows to analyze different frequency components. High-frequency components are analyzed with short windows, providing good time resolution, while low-frequency components are analyzed with long windows, providing good frequency resolution. The WT provides a way to perform multiresolution analysis. The Continuous Wavelet Transform (CWT) of a signal $x(t)$ is expressed in equation (44). Where $\Psi^*(t)$ denotes the complex conjugate of the base wavelet $\Psi(t)$, τ is the time shift, and s is the scaling parameter that controls frequency and time resolution.

$$CWT\{x(t)\} = WT(s,\tau) = \frac{1}{\sqrt{s}}\int_{-\infty}^{+\infty} x(t)\Psi^*\left(\frac{t-\tau}{s}\right)\, dt \tag{44}$$

The Discrete Wavelet Transform (DWT) can be obtained using discrete values for the scaling parameter s and the shifting parameter τ, like in equation (45). (With $s_0 > 1, \tau_0 \neq 0, j \in Z, k \in Z$.)

$$\begin{aligned} s &= s_0^j \\ \tau &= k\tau_0 s_0^j \end{aligned} \tag{45}$$

By substituting equation (45) into equation (44), using $s_0 = 2$ and $\tau_0 = 1$[6], equation (46) is obtained.

$$DWT\{x(t)\} = WT(j,k) = \frac{1}{\sqrt{2^j}} \int_{-\infty}^{+\infty} x(t)\Psi^* \left(\frac{t - k2^j}{2^j} \right) dt \tag{46}$$

The DWT decomposes a signal into a set of wavelet coefficients, which represent the signal at different scales. The DWT can be implemented by calculating the approximate coefficients $a_{j,k}$, which represent the low-frequency component of the signal, and the detailed coefficients $d_{j,k}$, which represent the high-frequency component of the signal. In a multilevel decomposition, the approximate and detailed coefficients of level j can be calculated by convolving the approximate coefficients of the previous level (j-1) with a low pass filter and a high pass filter as illustrated in equation (47) with the low pass filter h and the high pass filter g. For the first level, the approximate and detailed coefficients are calculated from the discrete signal. These equations are from [99], where more detailed mathematical developments are available. The decomposition of a signal with DWT is illustrated in Figure 33.

$$\begin{aligned} a_{j,k} &= \sum_m h(m - 2k) a_{j-1,m} \\ d_{j,k} &= \sum_m g(m - 2k) a_{j-1,m} \end{aligned} \tag{47}$$

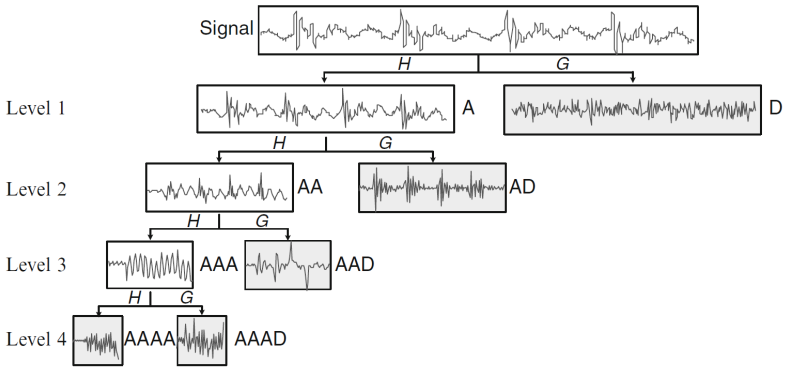

Figure 33: Illustration of a DWT with four-level decomposition from [99]. H: Low pass filter; G: High pass filter; A: Approximate information; D:Detailed information

Implementations from literature

This box illustrates key instances of NILM methods using the Fourier and wavelet transforms from the literature.

In [101], the authors compared the use of STFT and DWT to analyze the turn-on transient current of loads. The results indicated that the DWT performs better.

[6]The values $s_0 = 2$ and $\tau_0 = 1$ correspond to dyadic discretization. A common, simple, and efficient choice (see [100])

In [102], the authors (one is the same as in [101]) used DWT to detect and localize turn-on transient events. Then, an ANN, which takes active power and reactive power and turns on transient energy as input, is used to identify the load.

In [103], the authors proposed to use CWT instead of STFT to compute feature vectors from switching voltage transient. A support vector machine algorithm is then used to identify appliances with these feature vectors. The paper's results indicated that CWT improved the classification accuracy and reduced the computational requirement.

In [104] and [105], the authors proposed to use the Fast Fourier Transform (FFT) of the current waveform sampled at high frequencies (10.24 kHz and 50kHz) to create distinct load signatures. The FFT is an algorithm that computes the DFT efficiently.

In [37], the authors proposed a semi-supervised NILM method that uses both labeled and unlabeled data. First, new wavelets are created with wavelet design and procrustes analysis. Then, these wavelets are used to calculate the approximation and detailed coefficients of the current signal. The energy of wavelet coefficients is computed from these coefficients and used as a signature. A decision tree and nearest-neighbor classifier are combined to form a semi-supervised method. The training is done with the following process: a labeled dataset is split into two; the first part is used to train the DT, and the second part is used to train the NN. Then, if both algorithms agree on the label of an unlabelled data point, this data point is labeled and used to re-train the two algorithms. This process is done until there is no more unlabelled data.

In [106], the authors proposed an enhanced sequence-to-point method. The S2P architecture is enhanced by adding a bi-directional LSTM layer, an auto-encoder, and pruning after training. DWT pre-processes the main active power signal before sending it to the S2P algorithm. The authors stated that using DWT highlights each appliance's characteristics while reducing interference from device interactions and noise.

3.8 Review conclusion

In Section 3, a comprehensive review of most of the algorithms used for non-intrusive load monitoring was presented. These algorithms include optimization-based algorithms, hidden Markov models, classical machine learning techniques, deep learning architectures, methods utilizing voltage-current trajectories, and several other approaches. For each class of algorithms, their operational methodologies were explained in the context of energy disaggregation, and representative studies from the literature were highlighted to illustrate practical implementations.

These examples covered a wide range of NILM paradigms: supervised and unsupervised methods, event-based and eventless approaches, and both classification and regression tasks. In many implementations, multiple algorithms are combined. For instance, VI trajectories can be used with convolutional neural networks, dynamic time warping can be combined with k-nearest neighbors, and wavelet transforms can be paired with support vector machines, among others. It is also noteworthy that a specific algorithm may be applied within different taxonomic frameworks. For example, convolutional neural networks can be used in an event-based methodology when combined with VI trajectories, or independently in an eventless framework.

Overall, this literature review reveals that the NILM landscape is both methodologically diverse and extensively studied. No single approach can be universally identified as superior, given the trade-offs between accuracy, generalizability, computational complexity, and data requirements. The suitability

of a given method often depends on specific objectives (e.g., load classification vs. fine-grained energy estimation), data availability (e.g., sampling frequency, labeled vs. unlabeled data), and deployment constraints (e.g., smart meter capabilities). Even with well-defined objectives and constraints, a wide range of viable methods and algorithms remains available.

Nonetheless, if the primary objective is to maximize disaggregation performance in terms of standard error metrics (see Section 4.2), the current consensus in the literature suggests that supervised deep learning methods tend to outperform traditional techniques [15, 14, 107, 65]. However, these methods typically require substantial computational resources and access to large labeled datasets, which remain a significant barrier to their widespread deployment in real-world applications.

4 Data source and metrics

4.1 Public datasets

Datasets are essential for training and testing NILM methods. They can be labeled or unlabeled. A labeled dataset contains simultaneous measurements of aggregated consumption and individual appliance consumption. In contrast, an unlabeled dataset may include only aggregated data or only individual appliance consumption data. For testing NILM methods, supervised or unsupervised, having a labeled dataset is mandatory. Currently, there are several open-access datasets available online. Each public dataset has unique characteristics, such as the number of appliances, measurement frequency, measurement duration, country of origin, number of houses, year of measurement, and features (e.g., active power, reactive power, current, voltage, temperature, date, occupation). Typically, these datasets include measurements from one to several houses, with durations ranging from a few hours to several months and sampling rates from 15-minute intervals to dozens of kHz. The most common feature is active power, but voltage and current measurements are also frequently included. Below is a summary of the most popular and pertinent labeled datasets in the context of NILM in Belgium.

- REDD. The Reference Energy Disaggregation Data Set [108] is the most popular dataset in the literature [109]. It was released by MIT in 2011. This dataset has a sampling frequency of 15 kHz for the main voltage and current and 0.5 Hz for the power at the lower circuit level. From six houses over a month in Massachusetts, USA. At the time of this report, it is no longer accessible online.
- UKDALE. The United Kingdom's Domestic Appliance-Level Electricity recording [110] is also one of the most popular datasets in the literature. The first version of the dataset was released in 2015. This dataset has a maximal sampling frequency of 16 kHz for the whole house and 1/6 Hz for the individual appliances. The dataset contains measurements of voltage, current, active power, and apparent power at the whole-house level from five houses for 3 to 17 months in the UK. The dataset is freely available online[7].
- ECO. The Electricity Consumption and Occupancy dataset [111] was released in 2014. This dataset has a sampling frequency of 1 Hz for the whole house and for individual appliances. The data originate from four houses over eight months in Switzerland, with measurements of voltage, current, and phase shift at the whole-house level. The dataset is freely available online[8].
- REFIT. The REFIT dataset [112] was released in 2017. This dataset contains measurements with a sampling frequency of 1/8 Hz of the active power for the whole house and for individual appliances. The data were collected from twenty houses in the UK for two years. The dataset is freely available online[9].

More examples of open-access datasets are provided in the paper "A Critical Review of State-of-the-Art Non-Intrusive Load Monitoring Datasets" [109], a comprehensive review of the most widely used public datasets as of 2021. This review analyzes datasets from around the world, including several from Europe. Notable examples include the Tracebase dataset, which contains measurements from fifteen residential buildings in Germany; the IHEPCDS dataset, which contains measurements from one residential building in France; the ACS-Fx dataset, which includes measurements from one hundred home appliances in Switzerland; and the DRED dataset, which contains measurements from one residential building in the Netherlands.

[7] https://jack-kelly.com/data/
[8] https://vs.inf.ethz.ch/res/show.html?what=eco-data
[9] https://www.kaggle.com/datasets/kyleahmurphy/uk-electrical-load

To avoid the potentially time-consuming and costly measurement campaigns required to create new NILM datasets, some researchers advocate for the creation of synthetic datasets. In addition to eliminating the need for measurement campaigns, synthetic datasets offer the advantage of being free from measurement errors and data gaps, which are common in real datasets. Below are two notable examples of synthetic datasets:

- The Synthetic energy dataset (SynD) [113]. This synthetic dataset was released in 2020. It has a sampling frequency of 5 Hz for aggregated and appliance consumption. It is composed of data from 2 imaginary houses for 180 days. The dataset was generated with real representative consumption patterns of 21 appliances. The consumption patterns are the power consumption of appliances for a single operation measured in two Austrian households. The SynD dataset is available online[10].
- SmartSim [114] is a tool to generate synthetic datasets for NILM. It was released in 2016. A user can generate a synthetic dataset of aggregated and appliance consumption in three steps. First, the user selects appliance consumption patterns based on actual measurements from a library of devices. Second, the user selects a potential background noise. Third, the user selects the usage pattern for each appliance previously selected. The authors released a dataset generated with the tool. This dataset has a sampling frequency of 1 Hz for aggregated and appliance consumption. It is composed of data for one imaginary house for seven days. The SmartSim dataset is available online[11].

From this review [109], and an analysis of the available datasets, it can be noted that, today, the open-access datasets are insufficient to develop a fully operational NILM method for Belgium (or any other country). These open-access datasets are not sufficient for multiple reasons. First, the datasets are too small. Indeed, most of the time, they contain only comparable measurements from a limited number of households[12]. This is not sufficient to generalize or capture the diversity of an appliance's operation. One kind of appliance (e.g., a fridge) can have multiple manufacturers, multiple models, and multiple operating modes. In order to have a signature that generalizes all the possible instances of one kind of appliance, the dataset must include enough instances of this kind of appliance. Second, there is a lack of datasets with high-frequency sampling. In Belgium, the smart meter currently in deployment has a sampling frequency of 1 Hz. The number of datasets with a similar frequency is limited. Third, there is a lack of European and Belgian datasets. The same kind of appliance can be very different in different regions of the world regarding size or efficiency. The habits and lifestyles are also different, which can impact the time of use and the kind of appliance used. The use of air-conditioners is very rare in residential buildings in northern Europe but very common in the USA or China [115]. The grid frequency is also different in different regions of the world (e.g., 50 Hz in Europe and 60 Hz in North America), which can impact the signature of the appliances. Fourth, modern appliances like chargers for electric vehicles, heat pumps, electric boilers, and others that are fundamental in the energy transition at the residential level are often missing in these datasets.

In view of these observations, the authors of this document believe that to deploy NILM, new datasets should be created, in particular, European and Belgian datasets. The creation of new datasets can be easier than in the past. In Belgium, the aggregated data can be collected by connecting a dongle[13] on the P1 port of the smart meter (see section 5, and the appliance data can be collected with a simple and cheap smart plug. Access to such datasets would be extremely valuable to researchers seeking to develop and evaluate improved NILM algorithms. However, given the current level of algorithmic maturity in the NILM field, it is the authors' view that the responsibility for dataset collection

[10] https://springernature.figshare.com/collections/SynD_A_Synthetic_Energy_Dataset_for_Non-Intrusive_Load_Monitoring_in_Households/4716179

[11] https://github.com/sustainablecomputinglab/smartsim/

[12] Comparable measurements refer to data with similar characteristics, such as features, sampling frequency, appliance types, geographic region, etc.

[13] https://maconsosouslaloupe.be/ems

and software deployment should increasingly shift to private companies. In particular, stakeholders already active in the energy sector, such as energy providers, transmission system operators (TSOs), distribution system operators (DSOs), and aggregators, possess the necessary customer base, financial resources, and domain expertise to implement and scale NILM technologies effectively. Nonetheless, one challenge remains: these actors may currently lack strong financial incentives to adopt NILM, as the direct economic benefits for them appear limited under existing market structures.

4.2 Metrics

To effectively compare different implementations of NILM algorithms, it is essential to use common metrics. Since regression and classification algorithms have distinct goals and outputs, they require different sets of metrics for accurate evaluation.

4.2.1 Regression metrics

Regression metrics will measure if the predicted power of an appliance is close to its actual power and if the predicted energy consumption during the operation of an appliance is close to its actual energy consumption. Some of the regression metrics are defined below. With p_t, the ground-truth appliance power at time t, and $\hat{p}_t$, the estimated appliance power at time t, for a sequence of T predictions.

$$Mean\ Square\ Error\ (MSE) = \frac{1}{T}\sum_{t=1}^{T}(p_t - \hat{p}_t)^2 \tag{48}$$

$$Mean\ Absolute\ Error\ (MAE) = \frac{1}{T}\sum_{t=1}^{T}|p_t - \hat{p}_t| \tag{49}$$

$$Signal\ Aggregated\ Error\ (SAE) = \frac{|\sum_{t=1}^{T} p_t - \sum_{t=1}^{T}\hat{p}_t|}{\sum_{t=1}^{T} p_t} \tag{50}$$

In the literature, there are a lot of other metrics for energy disaggregation. Some of them are detailed in [14]. The reason for this abundance of metrics is that no metric can alone measure the quality of a disaggregation. An example is given in the Figure 34; it is obvious that the algorithm performs poorly and does not capture the cyclic consumption patterns of the fridge. Nevertheless, the SAE is very low, around 0.03, indicating that the estimated energy consumption is close to the actual energy consumption (With a 3% difference). The MAE, around 40 watts, indicates, however, the poor disaggregation performance. This example does not mean that MAE is a better metric than SAE; examples with low MAE for poor algorithms could also be found. Most authors, therefore, use multiple metrics to assess the quality of their disaggregation.

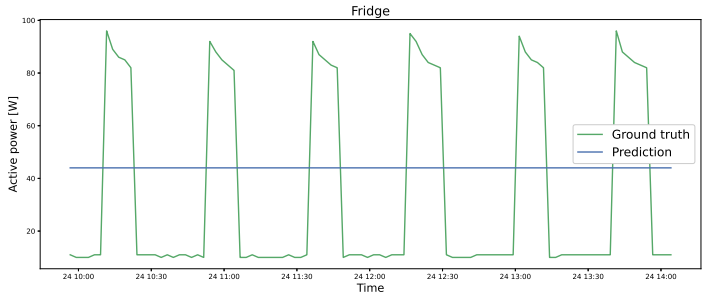

Figure 34: Example of incorrect disaggregation of a fridge with low SAE.

4.2.2 Classification metrics

The metrics used for the NILM classification algorithm are the same as for any binary classification problem. First, True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) are defined thanks to the confusion matrix in table 1.

Table 1: Confusion matrix.

Confusion matrix	Ground-truth Positive (ON)	Ground-truth Negative (OFF)
Predicted Positive (ON)	True Positive (TP)	False Positive (FP)
Predicted Negative (OFF)	False Negative (FN)	True Negative (TN)

The classification metrics are then defined below.

$$Precision = \frac{\#TP}{\#TP + \#FP} \tag{51}$$

$$Recall = \frac{\#TP}{\#TP + \#FN} \tag{52}$$

$$Accuracy = \frac{\#TP + \#TN}{\#TP + \#TN + \#FP + \#FN} \tag{53}$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{54}$$

The F1-score is commonly used to compare NILM classification algorithms. It is the harmonic mean of precision and recall, effectively combining these two metrics into a single representative value.

5 Belgium case

In Belgium, traditional electromechanical electricity meters are progressively being replaced by digital (or smart) meters, with distinct deployment objectives defined by each region. The deployment is managed by the regional Distribution System Operators (DSOs). In Flanders, *Fluvius* has already installed over 4.4 million digital meters for electricity and gas and aims to achieve 100% coverage by 2029[14] [15]. In the Brussels-Capital Region, *Sibelga* initiated a large-scale rollout in October 2023 and plans to equip nearly all households with smart meters by 2030[16]. In Wallonia, the Walloon Parliament has mandated that by 2030, 80% of customers with an annual standardized consumption of at least 6 MWh, or those generating electricity, must be equipped with digital meters[17]. These regional rollouts align with the requirements of the European Union Directive 2012/27/EU [116].

The digital meters currently being deployed are standardized across the major Belgian Distribution System Operators (DSOs), including *Fluvius*, *Sibelga*, *Ores*, and *Resa* (Ores and Resa being Walloon DSOs). The single-phase meters in use are the *Sagemcom-Siconia S211*, *XS212*, or the *Landis+Gyr E360 1P* models. For three-phase installations, the meters are the *Sagemcom-Siconia T211*, *XT211*, or the *Landis+Gyr E360 3P*[18]. The remaining Walloon DSOs, *AIEG*, *AIESH*, and *REW*, deploy smart meters manufactured by *Iskra*[19].

All of these smart meters are capable of transmitting consumption data to the respective DSOs at 15-minute intervals. Additionally, each device is equipped with a P1 port, a user-accessible interface that provides real-time data at a sampling rate of 1 Hz. The data available via the P1 port includes active power, root mean square (RMS) voltage, and RMS current measurements. The physical interface and communication protocol of the P1 port comply with the DSMR 5.0.2 P1 specification[20].

This standardized metering infrastructure enables the deployment of unified hardware and software solutions across all regions of Belgium. To maximize the performance of energy disaggregation algorithms, it is essential to fully exploit the available data from the P1 port. Therefore, the most effective algorithms should operate at a sampling frequency close to 1 Hz and leverage features derived from active power, voltage, and current signals.

Figures 35, 36, and 37 respectively show the digital meters *Sagemcom-Siconia S211*, *Landis+Gyr E360 1P*, and *Iskra*.

[14] Fluvius, https://pers.fluvius.be/rollout-of-digital-meters-among-prosumers-is-on-track, accessed 29/05/2025

[15] Flemish region, https://www.vlaanderen.be/bouwen-wonen-en-energie/elektriciteit-en-aardgas/digitale-meter, accessed 29/05/2025

[16] Sibelga, https://www.sibelga.be/en/connections-meters/smart-meters/when-will-i-get-a-smart-meter, accessed 28/05/2025

[17] Decree of 5 May 2022 of the Walloon parliament, https://www.ejustice.just.fgov.be/cgi/article_body.pl?language=fr&caller=summary&pub_date=2022-10-05&numac=2022033591

[18] Sibelga, https://www.sibelga.be/en/connections-meters/smart-meters/the-different-types-of-smart-meters consulted 28/05/2025

[19] AREWAL, https://www.arewal.be/presentation-du-compteur/ consulted 28/05/2025

[20] Ma conso sous la loupe, https://maconsosouslaloupe.be/developpeurs consulted 28/05/2025

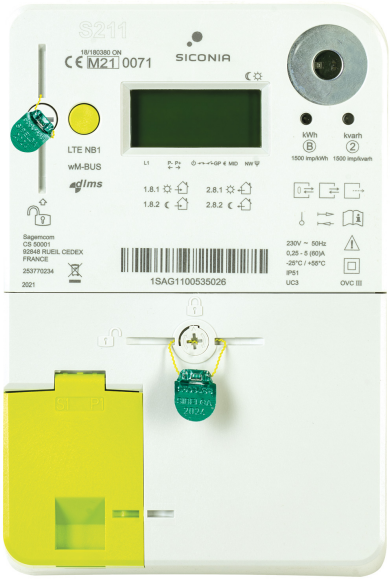

Figure 35: Sagecom-Siconia S211 one phase digital meter [117].

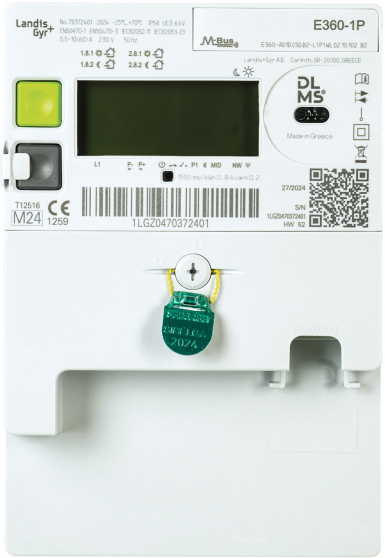

Figure 36: Landys+Gyr E360 1P one phase digital meter [118].

Figure 37: Iskra digital meter [119].

6 Algorithms implementation

In this section, the concrete implementation and performance of two algorithms suitable for NILM in Belgium are investigated. Deep learning methods were selected for implementation and further investigation due to their high performance, as explained in section 6.1. It is indeed the opinion of the authors that high-performing algorithms are necessary for a successful deployment of NILM. And that the complex labelled data collection necessary for those algorithms is justified by their performance. As explained in section 5, voltage, current, and aggregated active power are features available at a frequency of 1 Hz with the Belgian smart meter. We decided to investigate the potential benefit of such measurements on DL methods. In particular, our work will investigate and demonstrate that state-of-the-art Deep Learning (DL) disaggregation methods can be improved by considering reactive power. The following sections concerning our personal NILM implementation are structured as follows: Section 6.1 presents the relevance of our contribution. Sections 6.2 and 6.3 present the algorithms and the experiments implemented to test our proposal. Finally, sections 6.4 and 6.5 present the results and conclusions of our work. A summary of the work presented here has been published in the conference proceedings of IEEE ISGT 2024 [120].

6.1 Motivations

Deep learning methods have been very popular these last few years as they tend to achieve higher disaggregation performance [15], [14] [107], [65]. DL techniques have the potential to extract the relevant information in the aggregated signals and adapt well to unseen houses. Their main disadvantage is the need for a labeled dataset. In particular, DL methods composed of Convolutional Neural Networks (CNN) have been particularly successful. In [56], the authors proposed a Sequence-to-Point (S2P) CNN architecture; the input is a sequence of aggregated power, and the output is the estimated power of one appliance at the midpoint of the sequence. In [65], the authors analyzed a Sequence-to-Sequence (S2S) denoising Autoencoder (dAE). In [67] and [64], two S2S Variational Autoencoder (VAE) models based on convolutional layers were proposed. In [74], the authors proposed an S2P deep neural network based on residual connections and the attention mechanism. In these DL models and in the vast majority of publications, only aggregated sequences of the active power are used. To the best of the author's knowledge, active and reactive power sequences are only used in [66] to improve the performance of a dAE. Energy disaggregation is an ill-posed problem, and all the easily available information must be used to obtain methods with high performance. Today, modern smart meters like the Belgian smart meter do not only measure active power but also voltage, current, and apparent power. This information could be used to improve the performance of the existing algorithms.

This work proposes the use of active and reactive power sequences for the estimation of appliance power consumption with deep-learning regression methods. It is supposed that the additional use of reactive power will improve not only the detection of the appliance in use but also the estimation of its power consumption. In particular, the impact of a two-dimensional (2D) input with active and reactive power is evaluated in two state-of-the-art methods: (i) an updated version of the S2P algorithm introduced in [56], and (ii) a high-performing VAE from literature (VAE-NILM) [64].

6.2 Proposed method

Sequence-to-point (S2P) learning for Non-Intrusive Load Monitoring (NILM) was first introduced in [56]. The core idea of S2P is to use a sequence of aggregated power measurements as input to a neural network, which then predicts the power consumption of a target appliance at the midpoint of that sequence. Input sequences are generated using a sliding window approach, enabling disaggregation at multiple time steps. By focusing on the midpoint, the S2P architecture leverages contextual information from both the past and future aggregated consumption to improve prediction accuracy. In contrast, sequence-to-sequence (S2S) approaches predict an entire sequence of appliance-level power

outputs from the input sequence. In S2S, a single point in the output sequence is often predicted multiple times, once for each overlapping window, and the final value is typically obtained by averaging these predictions. However, this leads to degraded performance because the elements near the edges of the input windows are poorly predicted. Experimental results in [56] demonstrate that S2P outperforms S2S on two datasets and for two disaggregation metrics, MAE and SAE (see Section 4.2.

Figure 38 illustrates the concept of a sliding window in the context of two-dimensional sequence-to-point prediction (2D S2P). On the left, two-dimensional input sequences are generated using a sliding window (highlighted in red) applied to the aggregated active and reactive power measurements recorded by the household's smart meter. On the right, the goal is to predict the appliance-level power consumption at the midpoint of each input window. In this example, the sliding window covers one hour.

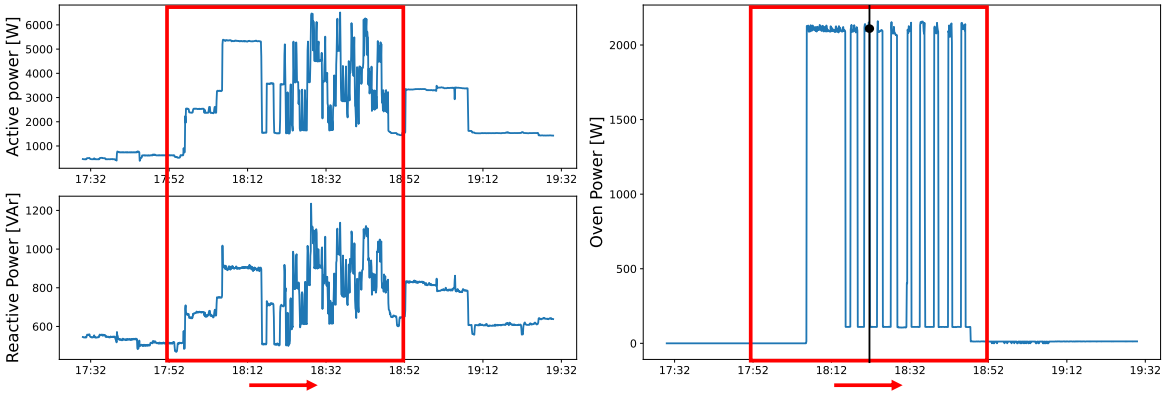

Figure 38: Sliding window concept for a 2D S2P.

In this work, we propose an improved version of the S2P architecture introduced in [56]. Compared to the original design, we introduce two key enhancements to the algorithm. First, active and reactive aggregated power sequences are used at the input instead of only the active power sequences. This changes the input from 1D to 2D. The addition of reactive power should improve the detection of appliances in use and the estimation of their power consumption, thus improving the overall disaggregation results. Active (P) and reactive (Q) power are indeed strongly linked together and to the appliance impedance (Z) through the apparent power (S) in 55:

$$S = \sqrt{P^2 + Q^2} = V_{rms} I_{rms} = \frac{V_{rms}^2}{Z} \tag{55}$$

The use of reactive power can improve the signature of each appliance by making them more unique and containing more information. Second, the architecture of the original CNN is modified with the following elements: the addition of Batch Normalization (BN) layers, the addition of Max Pooling (MP) layers, and the addition of a Drop-out (Dr) layer. This updated architecture is presented in Figure 39. It comprises five convolutional layers followed by two fully connected (dense) layers. In the first three convolutional layers, each is immediately followed by a Max Pooling layer and a Batch Normalization layer. In the remaining two convolutional layers, only Batch Normalization is applied after each layer. The network receives as input a two-dimensional time series representing one hour of aggregated active and reactive power, sampled every six seconds (resulting in 599 time steps). The model predicts the active power consumption of a specific appliance at the midpoint of the input window. One model has to be trained for every appliance.

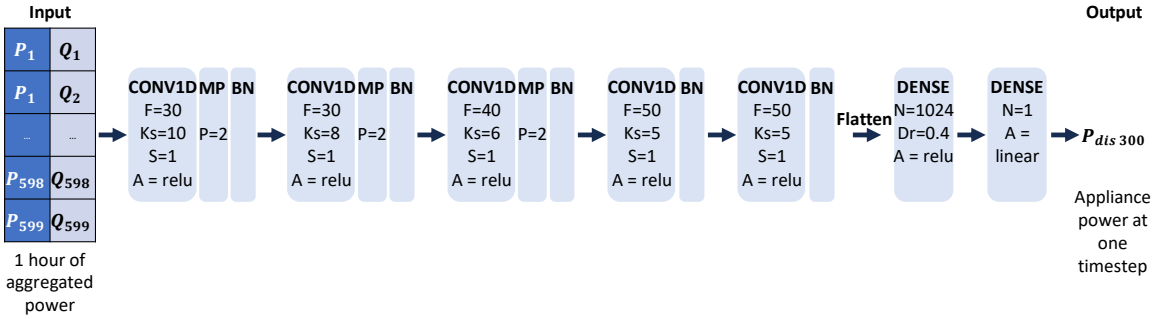

Figure 39: Architecture of the proposed updated S2P algorithm. Filter (F), Kernel size (Ks), Stride (S), Activation (A).

Batch normalization was first introduced in 2015 in [121]. BN standardizes the input (z) of each layer in a neural network. This process has many advantages; it helps prevent the gradient vanishing or exploding problem, accelerates training, improves performance, and reduces the impact of initialization. The equation (56) gives the normalized output $\hat{z}$ of a BN layer. During training, μ and σ are the mean and standard deviation of the current batch. During the inference, μ and σ are fixed and equal to a moving average of the mean and standard deviation seen during the training. γ and β are parameters learned during the training. BN was used to improve the performance of DL NILM models in [64], [122], [61], and [60].

$$\hat{z} = \gamma \times \frac{(z - \mu)}{\sigma} + \beta. \tag{56}$$

Dropout is a popular regularization technique in deep neural networks introduced in 2012 [123]. It helps to avoid over-fitting by randomly dropping a fixed percentage of the nodes during training.

Max pooling is a technique used to reduce the dimension of the input feature map. A reduced number of parameters reduces the risk of overfitting and helps generalize the model to unseen data. It also lowers the computational cost of the model. With MP, the updated S2P (presented in Figure 39) has three million parameters instead of thirty million in the original architecture from [56].

The model architecture and hyperparameters are inspired by the literature and have been adapted either heuristically or by grid-search. An analysis of the optimal number of layers with a similar architecture is done in [61]. Like in [66] and [61], although the input has a two-dimensional shape (window size, 2), the model is composed of 1D convolutional layers, meaning that the kernels are only moving along the temporal axis. The input data are pre-processed before training and inference. The 2D input and the target are standardized by subtracting the mean and standard deviation of the appliance active power with the same values as in [56].

The S2S VAE is exactly the same as in the original paper [64], except for the input layer, which has been modified to accept 2D input sequences. Figure 40 illustrates the VAE-NILM architecture. The variational autoencoder (VAE) consists of two main components: an encoder and a decoder (see Section 3.5.4). The encoder is composed of seven IBN-Net sub-networks [124], followed by two fully connected layers. Its objective is to map the relevant information from the aggregated power signal into a regularized latent space. The decoder mirrors the encoder architecture and aims to reconstruct the power signal of the target appliance from this latent representation. To improve reconstruction accuracy, skip connections are introduced between corresponding IBN-Net blocks in the encoder and decoder. These connections enhance performance by allowing the decoder to reuse informative details from the aggregated signal during the reconstruction of the appliance-specific load. The implementation is explained in detail in the original paper. The original *python* code is available in the VAE-NILM

repository[21].

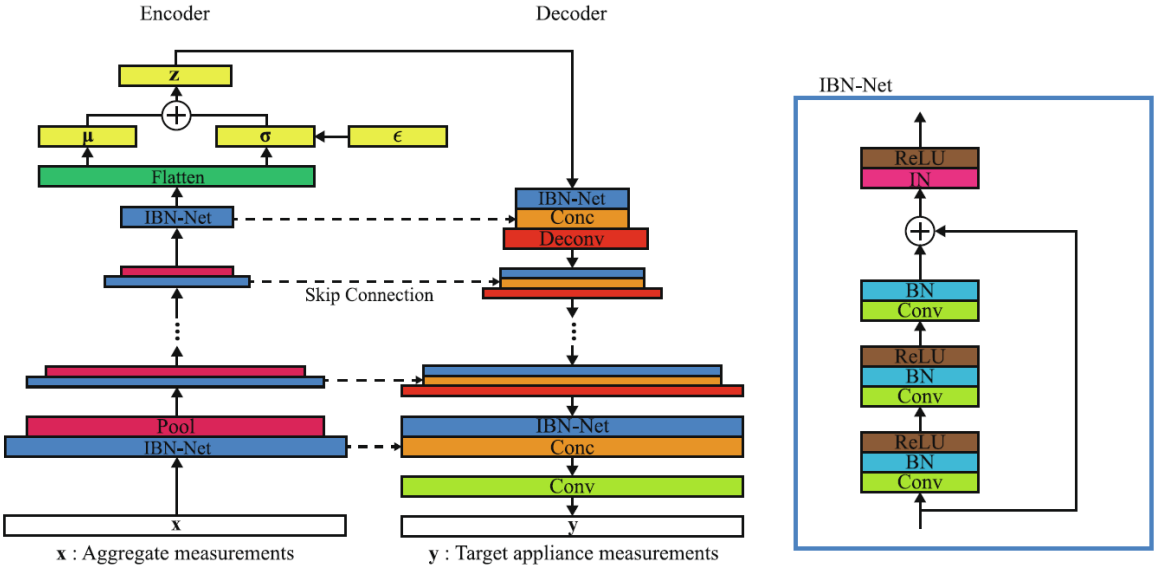

Figure 40: Architecture of the VAE-NILM algorithm from [64].

6.3 Experiments

The UK-DALE dataset [110] was chosen to conduct the experiments and benchmark our model against existing approaches. This dataset was selected for two main reasons: (i) it provides measurements of both aggregated active power and apparent power, enabling the construction of a two-dimensional time series input that includes both active and reactive power; and (ii) it is one of the most widely used datasets in the NILM research community [109], facilitating comparison with results from the literature. The UK-DALE dataset contains a recording of the whole-house power and appliance power of five houses in the UK with a sampling frequency of 1/6 Hz (see section 4.1). Only houses 1, 2, and 5 were used in our experiments as they are the only ones having a recording of the active (P) and apparent power (S). The reactive power (Q) is calculated with equation (57).

$$Q = \sqrt{S^2 - P^2} \tag{57}$$

For the updated S2P, the length of the input sequences is one hour or 599 points. This length could be adapted for each appliance, like in [65]. Nevertheless, for simplicity's sake, the input sequence length is the same for each appliance. To create the training set, sequences are selected with a stride of 20 points (2 min). h1 and h5 are used for training, and h2 for testing. It is essential to evaluate the model on unseen houses, as it will be the case for real-life applications. The experiments are done in *python* with *Tensorflow.keras*. The number of epochs is set to ten. *Adam* optimizer and Ridge regularization are used. The optimizer and regularization hyperparameters are listed in Table 2. They were selected using Grid Search over a limited set of candidate values.

[21] https://github.com/ETSSmartRes/VAE-NILM

Table 2: Optimizer and regularization parameters of the updated S2P model.

Adam	
learning rate	10^{-3}
β_1	0.9
β_2	0.999
ϵ	10^{-7}
Ridge regression	
l_2	10^{-7}

The two models (VAE-NILM and updated S2P) are trained on houses 1 and 5 and tested on house 2. They are trained and tested ten times for each appliance, and the average results are presented in section 6.4. The models are evaluated with two popular metrics: the Mean Absolute Error (MAE) and the normalized Signal Aggregated Error (SAE). The MAE and SAE are calculated with equations (49) and (50) in section 4.2.

6.4 Results

Figure 41 presents examples of disaggregation with our 2D updated S2P model. The model used for the Figure is one of the best of the ten training models. In the figure, the ground-truth active power of the appliance is shown in green, while the predictions generated by the proposed model are depicted in blue.

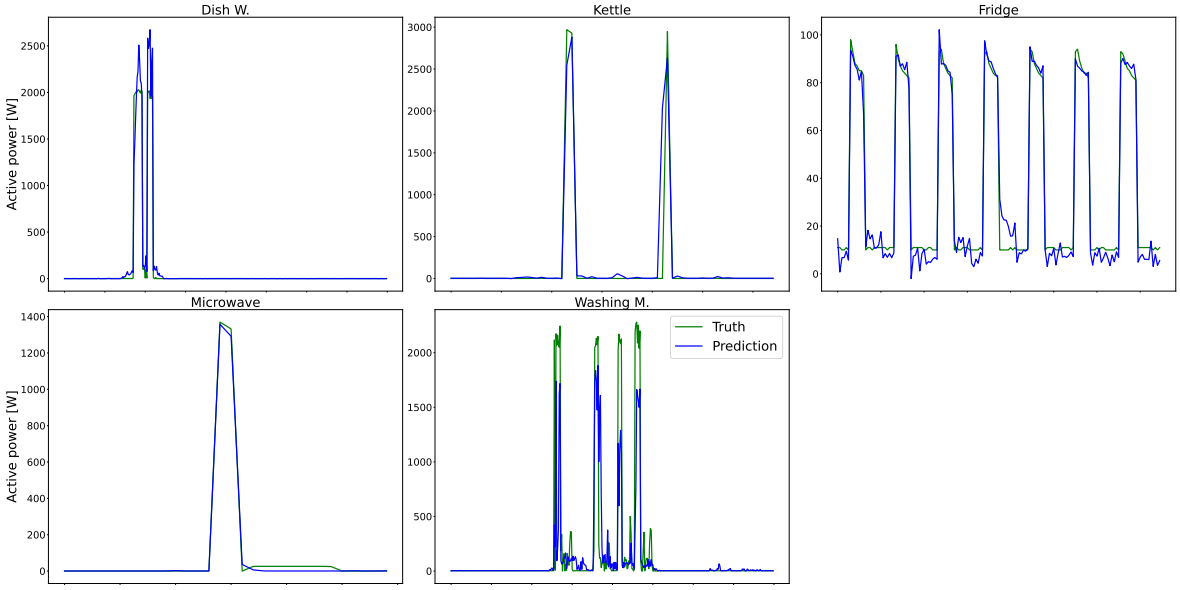

Figure 41: Energy disaggregation with the updated 2D S2P algorithm on the unseen house 2 of UKDALE dataset.

Figure 42 presents examples of disaggregation with the 2D VAE-NILM model. The model used for the Figure is one of the best of the ten training models. In the examples of Figure 41 and 42, both algorithms perform pretty well.

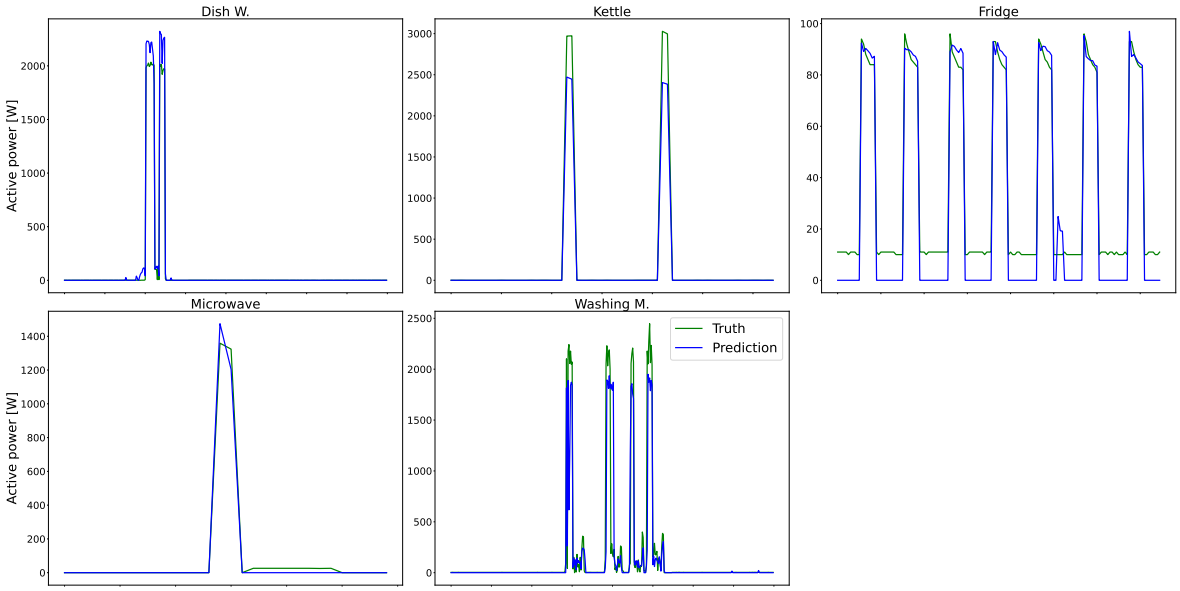

Figure 42: Energy disaggregation with the 2D VAE-NILM algorithm on the unseen house 2 of UKDALE dataset.

Table 3 presents a comparison of the results of our experiments. The results of the VAE-NILM algorithm obtained with only the active power are coherent with the results of the original paper [64] as expected. For the VAE-NILM algorithm, introducing the reactive power sequence improved the MAE and SAE for most appliances. The kettle is the only appliance where the introduction of reactive power deteriorates both MAE and SAE. On average, the MAE is improved by 11% and the SAE by 36%. For our updated S2P algorithm, the MAE and SAE are improved on every appliance except for the MAE of the WM. On average, the MAE is improved by 30% and the SAE by 57%. It is worth noting that this comparison is fair regarding model complexity. The four models each have approximately 3.8 million parameters. Introducing the reactive power increases the number of parameters by a few hundred, as it affects only the first convolutional layer of the models.

Table 3: Comparison of models from experiments on house 2 of the UK-DALE dataset

Metric	Model	Kettle	Microwave	Fridge	Dish W.	Washing M.	Average
MAE [W]	1D VAE-NILM	**6.2 ± 0.4**	5.2 ± 0.2	14.9 ± 0.19	**10.7 ± 2.1**	6.3 ± 0.4	8.7
	2D VAE-NILM	6.6 ± 0.3	**2.0 ± 0.1**	**11.2 ± 0.3**	12.6 ± 1.8	**6.2 ± 0.6**	**7.7**
	1D updated S2P	9.0 ± 2.5	11.8 ± 2.9	12.2 ± 0.6	32.1 ± 3.8	**10.6 ± 2.5**	15.1
	2D updated S2P	**8.5 ± 1.6**	**4.2 ± 1.1**	**11.4 ± 0.7**	**17.2 ± 4.9**	11.5 ± 2.0	**10.6**
SAE	1D VAE-NILM	**0.17 ± 0.01**	0.42 ± 0.04	0.15 ± 0.02	0.13 ± 0.06	**0.37 ± 0.03**	0.25
	2D VAE-NILM	0.20 ± 0.01	**0.06 ± 0.02**	**0.11 ± 0.01**	**0.07 ± 0.07**	**0.37 ± 0.04**	**0.16**
	1D updated S2P	0.15 ± 0.07	1.10 ± 0.47	0.13 ± 0.03	0.61 ± 0.13	0.23 ± 0.12	0.44
	2D updated S2P	**0.09 ± 0.03**	**0.47 ± 0.19**	**0.10 ± 0.02**	**0.14 ± 0.08**	**0.16 ± 0.08**	**0.19**

Table 4 compares the improved algorithms against the performance of state-of-the-art algorithms from the literature. While comparing the performance, it is important to note that our results are the average of ten simulations. However, we do not know if the results from the literature are also an average of multiple simulations. It is also interesting to note that the other papers use houses 1,3,4, and 5 for training, whereas we only use houses 1 and 5. Against state-of-the-art models, the 2D VAE-NILM and the 2D updated S2P are the best and the second best for the MAE metric. They are the second and

the fourth best on the SAE metric. The SAE of the 2D updated S2P is significantly reduced because of its performance for the microwave. It can also be noted that our updated 2D S2P performs better than the original Seq2point algorithm, with an improvement of 35% for the MAE and 42% for the SAE.

Table 4: Comparison of models from literature on house 2 of the UK-DALE dataset

Metric	Model	Kettle	Microwave	Fridge	Dish W.	Washing M.	Average
MAE [W]	AFHMM [56]	47.4	21.2	42.3	199.8	103.2	77.7
	Seq2point [56]	7.4	8.7	20.9	27.7	12.7	16.2
	CVAE [67]	7.0	7.5	18.1	19.6	10.8	13.0
	ResNet+att [74]	-	5.7	12.3	18.2	**6.2**	10.6
	2D VAE-NILM	**6.6**	**2.0**	**11.2**	**12.6**	**6.2**	**7.7**
	2D s2p	8.5	4.2	11.4	17.2	11.5	10.5
SAE	AFHMM [56]	1.06	1.04	0.98	4.5	8.28	1.89
	Seq2point [56]	0.07	0.49	0.12	0.64	0.28	0.33
	CVAE [67]	**0.06**	0.18	0.13	0.32	0.21	0.17
	ResNet+att [74]	-	0.17	**0.08**	0.21	**0.07**	**0.12**
	2D VAE-NILM	0.20	**0.06**	0.11	**0.07**	0.37	0.16
	2D updated S2P	0.09	0.47	0.10	0.14	0.16	0.19

Finally, Figure 43 presents a comparison of microwave disaggregation performance using either 1D input (blue) or 2D input (red) of our updated S2P over several days (the best models were selected for this comparison). In this example, incorporating reactive power in the 2D input improves both the detection and estimation of microwave consumption. Specifically, the 2D input yields fewer false positives and false negatives, and the estimated power more closely aligns with the ground truth.

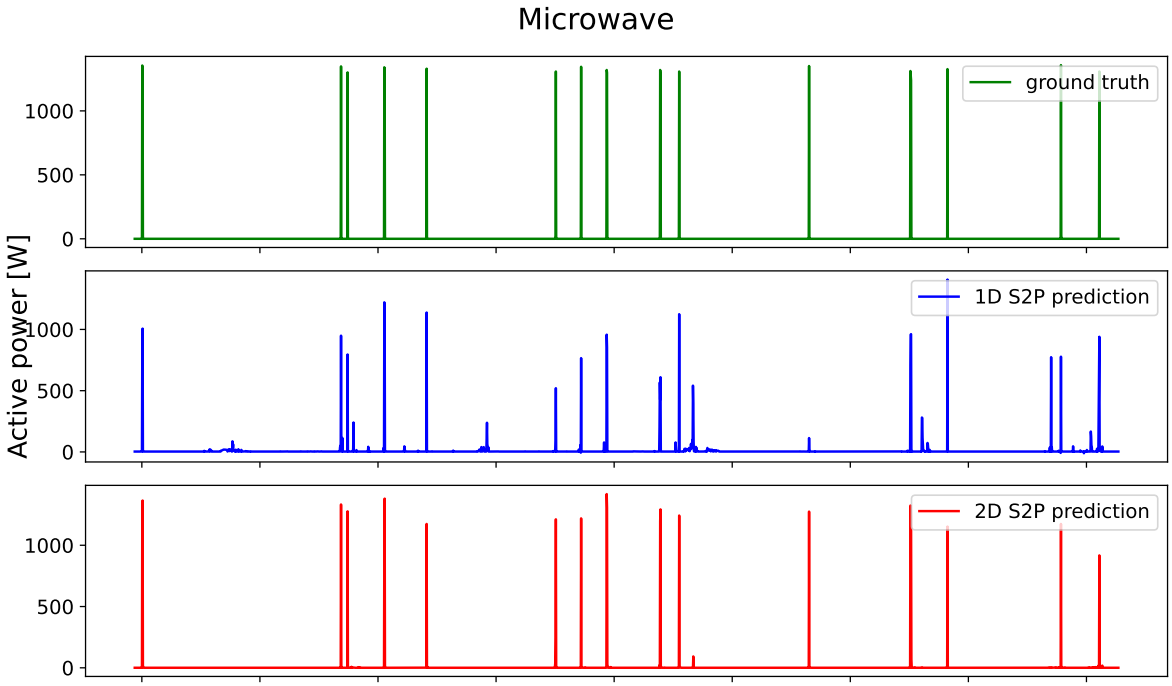

Figure 43: Comparison of microwave disaggregation with 1D and 2D updated S2P.

6.5 Conclusion of the implementation

In our experiments, we investigated the impact of using reactive power sequences for energy disaggregation with DL models. Our results indicated that the use of reactive power has a positive impact on the majority of the appliances for two state-of-the-art architectures: one VAE and one S2P CNN. As the impact is not positive for every appliance, it could be decided to use reactive power only for relevant appliances. Therefore, we believe that future DL models should incorporate reactive power in addition to active power as input, as it is an easy way to improve performance. The main reason why DL algorithms such as the ones presented in the previous sections are not deployed in real-life applications is the lack of labelled datasets. For instance, the UK-DALE dataset used in our experiments is already ten years old and contains a limited number of houses and appliances. Such datasets are very useful to compare the proposed algorithm to the literature, but they are not sufficient for the development of roll-out NILM algorithms. Therefore, we believe that new datasets should be developed. These new datasets should contain more instances of houses and modern appliances and correspond to modern smart meter measurements. The creation of new datasets would be extremely valuable for researchers to continue developing NILM algorithms. As NILM methods have already reached a high level of maturity and demonstrate strong performance, it may now be time for private companies to take the lead in developing such datasets to enable the deployment of commercial NILM solutions. However, the deployment of this technology must also be financially profitable for these companies.

7 Conclusion

This report provides an overview of non-intrusive load monitoring, with a particular focus on its application for residential users in Belgium. It introduces and motivates the concept of NILM, details the core principles and methodologies, and offers a comprehensive review of the literature. Additionally, the report investigates energy disaggregation algorithms based on deep learning through practical implementation. This report equips readers with the knowledge to understand the functioning and limitations of NILM and to select an appropriate disaggregation algorithm for specific situations.

In the context of deploying NILM for residential users in Belgium, we estimate that supervised deep learning methods are among the best candidates due to their superior performance. Disaggregated energy feedback is valuable only if it achieves a minimal performance threshold. For deployment in Belgium, we propose leveraging the information available from smart meters rather than focusing solely on the aggregated active power signal, as is common in many state-of-the-art deep learning models. This report demonstrates that using both active and reactive power signals can significantly enhance the performance of these models. Further investigation should be carried on to evaluate the potential impact of other available signals, such as current, voltage, or apparent power, on performance.

In this report, we highlighted that one of the main barriers to the deployment of NILM is the lack of available datasets, particularly recent, labeled datasets from Europe. Currently available datasets do not include enough houses and appliances to create robust disaggregation models suitable for deployment. Additionally, many of these datasets are outdated and sourced from different countries, leading to models that perform poorly in real-life applications. These older datasets also lack representation of modern appliances, such as electric vehicles and heat pumps. Therefore, it is the author's opinion that new datasets should be developed, with a focus on residential measurements in Belgium. In view of the high performance of state-of-the-art NILM algorithms, it may now be time for stakeholders and private companies to take the lead in developing datasets geared toward commercial deployment. The creation of new datasets may no longer fall primarily within the responsibility of academic researchers. Nevertheless, open-access datasets will remain highly valuable for the scientific community.

The creation of NILM datasets was a challenging task in the past. However, we believe that creating these datasets can be more straightforward and cost-effective today. In Belgium, aggregated data can be collected by connecting a dongle to the P1 port of a smart meter, while appliance-specific data can be gathered using inexpensive smart plugs. With the proposed deep learning algorithms, it is possible to develop a neural network for each appliance. This significantly simplifies the data collection process, as only the measurements of aggregated data and one appliance at a time are needed. These deep learning models also enable more efficient data collection by targeting a limited number of key appliances. Furthermore, using transfer learning, neural networks can initially be trained on a source dataset (which may include measurements from other countries or older datasets) and then fine-tuned on a smaller, more specific dataset. This approach can enhance the performance of the algorithms while reducing the amount of new data that needs to be collected.

It is evident from this report that there has been extensive research into the development of NILM algorithms over the past decades. While such theoretical research remains valuable, the authors suggest that future research should focus more on practical implementation. This includes conducting measurement campaigns for dataset creation, training algorithms, and developing user-friendly platforms to provide energy feedback to residential users. Practical implementations are essential to evaluate whether NILM can effectively lead to consumption reduction through disaggregated energy feedback. Additionally, they are crucial to assess the potential of NILM in supporting the development of smart grids, demand-side management, and overall grid flexibility.

References

[1] European Commission and Directorate-General for Energy. *EU energy in figures – Statistical pocketbook 2023*. Publications Office of the European Union, 2023.

[2] Rishika Agarwal, Madhur Garg, Dharani Tejaswini, Vishal Garg, Priyanka Srivastava, Jyotirmay Mathur, and Rajat Gupta. A review of residential energy feedback studies. *Energy and Buildings*, page 113071, 2023.

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

[4] Jack Kelly and William Knottenbelt. Does disaggregated electricity feedback reduce domestic electricity consumption? a systematic review of the literature. *arXiv preprint arXiv:1605.00962*, 2016.

[5] IM Chatzigeorgiou and GT Andreou. A systematic review on feedback research for residential energy behavior change through mobile and web interfaces. *Renewable and Sustainable Energy Reviews*, 135:110187, 2021.

[6] Kevin Trinh, Alan S Fung, and Vera Straka. Effects of real-time energy feedback and normative comparisons: results from a multi-year field study in a multi-unit residential building. *Energy and Buildings*, 250:111288, 2021.

[7] European Union Agency for the Cooperation of Energy Regulators (ACER). *Report on Electricity Transmission and Distribution Tariff Methodologies in Europe*. ACER, 2023.

[8] RESA S.A. INTERCOMMUNALE. Le compteur intelligent mode d'emploi modèle monophasé s211. https://www.resa.be/wp-content/uploads/2020/05/Compteur-Intelligent-Mode-demploi.pdf, 2020. Online; accessed 01 July 2024.

[9] Taha Hassan, Fahad Javed, and Naveed Arshad. An empirical investigation of vi trajectory based load signatures for non-intrusive load monitoring. *IEEE Transactions on Smart Grid*, 5(2):870–878, 2013.

[10] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[11] Panagiotis Papadimitroulas et al. Artificial intelligence: Deep learning in oncological radiomics and challenges of interpretability and data harmonization. *Physica Medica*, 83:108–121, 2021.

[12] D. K. Thara, B. G. PremaSudha, and Fan Xiong. Auto-detection of epileptic seizure events using deep neural network with different feature scaling techniques. *Pattern Recognition Letters*, 128:544–550, 2019.

[13] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[14] Pascal A Schirmer and Iosif Mporas. Non-intrusive load monitoring: A review. *IEEE Transactions on Smart Grid*, 14(1):769–784, 2022.

[15] Suryalok Dash and NC Sahoo. Electric energy disaggregation via non-intrusive load monitoring: A state-of-the-art systematic review. *Electric Power Systems Research*, 213:108673, 2022.

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

[17] Ram Machlev, Juri Belikov, Yuval Beck, and Yoash Levron. Mo-nilm: A multi-objective evolutionary algorithm for nilm classification. *Energy and Buildings*, 199:134–144, 2019.

[18] Wen Fan, Qing Liu, Ali Ahmadpour, and Saeed Gholami Farkoush. Multi-objective non-intrusive load disaggregation based on appliances characteristics in smart homes. *Energy Reports*, 7:4445–4459, 2021.

[19] Soumyajit Ghosh and Debashis Chatterjee. Artificial bee colony optimization based non-intrusive appliances load monitoring technique in a smart home. *IEEE Transactions on Consumer Electronics*, 67(1):77–86, 2021.

[20] Oladayo S Ajani, Abhishek Kumar, Rammohan Mallipeddi, Swagatam Das, and Ponnuthurai Nagaratnam Suganthan. Benchmarking optimization-based energy disaggregation algorithms. *Energies*, 15(5):1600, 2022.

[21] Chuyi Li, Kedi Zheng, Hongye Guo, and Qixin Chen. A mixed-integer programming approach for industrial non-intrusive load monitoring. *Applied Energy*, 330:120295, 2023.

[22] Jeewon Park, Oladayo S Ajani, and Rammohan Mallipeddi. Optimization-based energy disaggregation: A constrained multi-objective approach. *Mathematics*, 11(3):563, 2023.

[23] Zoubin Ghahramani and Michael I Jordan. Factorial hidden markov models. *Machine Learning*, 29(2-3):245, 1997.

[24] Lawrence Rabiner and Biinghwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[25] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

[26] Hyungsul Kim, Manish Marwah, Martin Arlitt, Geoff Lyon, and Jiawei Han. Unsupervised disaggregation of low frequency power measurements. In *Proceedings of the 2011 SIAM international conference on data mining*, pages 747–758. SIAM, 2011.

[27] Oliver Parson, Siddhartha Ghosh, Mark Weal, and Alex Rogers. An unsupervised training method for non-intrusive appliance load monitoring. *Artificial Intelligence*, 217:1–19, 2014.

[28] Zhao Wu, Chao Wang, Wenxiong Peng, Weihua Liu, and Huaiqing Zhang. Non-intrusive load monitoring using factorial hidden markov model based on adaptive density peak clustering. *Energy and Buildings*, 244:111025, 2021.

[29] Zhao Wu, Chao Wang, Huaiqing Zhang, Wenxiong Peng, and Weihua Liu. A time-efficient factorial hidden semi-markov model for non-intrusive load monitoring. *Electric Power Systems Research*, 199:107372, 2021.

[30] Roberto Bonfigli, Emanuele Principi, Marco Fagiani, Marco Severini, Stefano Squartini, and Francesco Piazza. Non-intrusive load monitoring by using active and reactive power in additive factorial hidden markov models. *Applied Energy*, 208:1590–1607, 2017.

[31] J Zico Kolter and Tommi Jaakkola. Approximate inference in additive factorial hmms with application to energy disaggregation. In *Artificial intelligence and statistics*, pages 1472–1482. PMLR, 2012.

[32] Dominik Egarter, Venkata Pathuri Bhuvana, and Wilfried Elmenreich. Paldi: Online load disaggregation via particle filtering. *IEEE Transactions on Instrumentation and Measurement*, 64(2):467–477, 2014.

[33] Stephen Makonin, Fred Popowich, Ivan V. Bajić, Bob Gill, and Lyn Bartram. Exploiting hmm sparsity to perform online real-time nonintrusive load monitoring. *IEEE Transactions on Smart Grid*, 7(6):2575–2585, 2016.

[34] Attique Ur Rehman, Tek Tjing Lie, Brice Vallès, and Shafiqur Rahman Tito. Comparative evaluation of machine learning models and input feature space for non-intrusive load monitoring. *Journal of Modern Power Systems and Clean Energy*, 9(5):1161–1171, 2021.

[35] Nguyen Viet Linh and Pablo Arboleya. Deep learning application to non-intrusive load monitoring. In *2019 IEEE Milan PowerTech*, pages 1–5. IEEE, 2019.

[36] Mozaffar Etezadifar, Houshang Karimi, and Jean Mahseredjian. Non-intrusive load monitoring: Comparative analysis of transient state clustering methods. *Electric Power Systems Research*, 223:109644, 2023.

[37] Jessie M Gillis and Walid G Morsi. Non-intrusive load monitoring using semi-supervised machine learning and wavelet design. *IEEE Transactions on Smart Grid*, 8(6):2648–2655, 2016.

[38] David Meyer and FT Wien. Support vector machines. *R News*, 1(3):23–26, 2001.

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

[40] Yaqian Li, Yuquan Yang, Kai Sima, Boyang Li, Tong Sun, and Xue Li. Non-intrusive load monitoring based on harmonic characteristics. *Procedia Computer Science*, 183:776–782, 2021.

[41] Jing Liao, Georgia Elafoudi, Lina Stankovic, and Vladimir Stankovic. Non-intrusive appliance load monitoring using low-resolution smart meter data. In *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 535–540. IEEE, 2014.

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

[43] Leen De Baets, Tom Dhaene, Dirk Deschrijver, Chris Develder, and Mario Berges. Vi-based appliance classification using aggregated power consumption data. In *2018 IEEE international conference on smart computing (SMARTCOMP)*, pages 179–186. IEEE, 2018.

[44] Bundit Buddhahai and Stephen Makonin. A nonintrusive load monitoring based on multi-target regression approach. *IEEE Access*, 9:163033–163042, 2021.

[45] Chuan Choong Yang, Chit Siang Soh, and Vooi Voon Yap. A non-intrusive appliance load monitoring for efficient energy consumption based on naive bayes classifier. *Sustainable Computing: Informatics and Systems*, 14:34–42, 2017.

[46] Mohammad Kaosain Akbar, Manar Amayri, Nizar Bouguila, Benoit Delinchant, and Frederic Wurtz. Evaluation of regression models and bayes-ensemble regressor technique for non-intrusive load monitoring. *Sustainable Energy, Grids and Networks*, 38:101294, 2024.

[47] Máximo Eduardo Sánchez-Gutiérrez and Pedro Pablo González-Pérez. Multi-class classification of medical data based on neural network pruning and information-entropy measures. *Entropy*, 24(2):196, 2022.

[48] Alexander LeNail. Nn-svg: Publication-ready neural network architecture schematics. *J. Open Source Softw.*, 4(33):747, 2019. Online tool https://alexlenail.me/NN-SVG/index.html; accessed 08 April 2024.

[49] Michael Devlin and Barry P Hayes. Non-intrusive load monitoring using electricity smart meter data: A deep learning approach. In *2019 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2019.

[50] Christopher Olah. Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. Online; accessed 04 April 2024.

[51] Sagar Verma, Shikha Singh, and Angshul Majumdar. Multi-label lstm autoencoder for non-intrusive appliance load monitoring. *Electric Power Systems Research*, 199:107414, 2021.

[52] Xinxin Zhou, Jingru Feng, and Yang Li. Non-intrusive load decomposition based on cnn–lstm hybrid deep learning model. *Energy Reports*, 7:5762–5771, 2021.

[53] Jihyun Kim, Thi-Thu-Huong Le, Howon Kim, et al. Nonintrusive load monitoring based on advanced deep learning and novel signature. *Computational intelligence and neuroscience*, 2017, 2017.

[54] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

[55] Sumit Saha. A comprehensive guide to convolutional neural networks—the eli5 way. *Towards data science*, 15:15, 2018.

[56] Chaoyun Zhang, Mingjun Zhong, Zongzuo Wang, Nigel Goddard, and Charles Sutton. Sequence-to-point learning with neural networks for non-intrusive load monitoring. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[57] Mohammed Ayub and El-Sayed M El-Alfy. Multi-target energy disaggregation using convolutional neural networks. *International Journal of Advanced Computer Science and Applications*, 11(10), 2020.

[58] Christos Athanasiadis, Dimitrios Doukas, Theofilos Papadopoulos, and Antonios Chrysopoulos. A scalable real-time non-intrusive load monitoring system for the estimation of household appliance power consumption. *Energies*, 14(3):767, 2021.

[59] Weijun Yang, Chengxin Pang, Jinhai Huang, and Xinhua Zeng. Sequence-to-point learning based on temporal convolutional networks for nonintrusive load monitoring. *IEEE Transactions on Instrumentation and Measurement*, 70:1–10, 2021.

[60] Ziyue Jia, Linfeng Yang, Zhenrong Zhang, Hui Liu, and Fannie Kong. Sequence to point learning based on bidirectional dilated residual network for non-intrusive load monitoring. *International Journal of Electrical Power & Energy Systems*, 129:106837, 2021.

[61] Kai Ye, Hyeonjin Kim, Yi Hu, Ning Lu, Di Wu, and PJ Rehm. A modified sequence-to-point hvac load disaggregation algorithm. In *2023 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2023.

[62] Wenpeng Luan, Ruiqi Zhang, Bo Liu, Bochao Zhao, and Yixin Yu. Leveraging sequence-to-sequence learning for online non-intrusive load monitoring in edge device. *International Journal of Electrical Power & Energy Systems*, 148:108910, 2023.

[63] Mohammed Ayub and El-Sayed M El-Alfy. Contextual sequence-to-point deep learning for household energy disaggregation. *IEEE Access*, 2023.

[64] Antoine Langevin, Marc-André Carbonneau, Mohamed Cheriet, and Ghyslain Gagnon. Energy disaggregation using variational autoencoders. *Energy and Buildings*, 254:111623, 2022.

[65] Jack Kelly and William Knottenbelt. Neural nilm: Deep neural networks applied to energy disaggregation. In *Proceedings of the 2nd ACM international conference on embedded systems for energy-efficient built environments*, pages 55–64, 2015.

[66] Michele Valenti, Roberto Bonfigli, Emanuele Principi, and Stefano Squartini. Exploiting the reactive power in deep neural models for non-intrusive load monitoring. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[67] Tharmakulasingam Sirojan, B Toan Phung, and Eliathamby Ambikairajah. Deep neural network based energy disaggregation. In *2018 IEEE International conference on smart energy grid engineering (SEGE)*, pages 73–77. IEEE, 2018.

[68] Maria Kaselimi, Athanasios Voulodimos, Eftychios Protopapadakis, Nikolaos Doulamis, and Anastasios Doulamis. Energan: A generative adversarial network for energy disaggregation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1578–1582. IEEE, 2020.

[69] Maria Kaselimi, Nikolaos Doulamis, Athanasios Voulodimos, Anastasios Doulamis, and Eftychios Protopapadakis. Energan++: A generative adversarial gated recurrent network for robust energy disaggregation. *IEEE Open Journal of Signal Processing*, 2:1–16, 2020.

[70] Awadelrahman MA Ahmed, Yan Zhang, and Frank Eliassen. Generative adversarial networks and transfer learning for non-intrusive load monitoring in smart grids. In *2020 IEEE international conference on communications, control, and computing technologies for smart grids (SmartGridComm)*, pages 1–7. IEEE, 2020.

[71] Kunjin Chen, Yu Zhang, Qin Wang, Jun Hu, Hang Fan, and Jinliang He. Scale-and context-aware convolutional non-intrusive load monitoring. *IEEE Transactions on Power Systems*, 35(3):2362–2373, 2019.

[72] Dea Pujić, Nikola Tomašević, and Marko Batić. A semi-supervised approach for improving generalization in non-intrusive load monitoring. *Sensors*, 23(3):1444, 2023.

[73] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[74] Zhuojie Nie, Yongbiao Yang, and Qingshan Xu. An ensemble-policy non-intrusive load monitoring technique based entirely on deep feature-guided attention mechanism. *Energy and Buildings*, 273:112356, 2022.

[75] Himanshu Grover, Lokesh Panwar, Ashu Verma, Bijaya K Panigrahi, and TS Bhatti. A multi-head convolutional neural network based non-intrusive load monitoring algorithm under dynamic grid voltage conditions. *Sustainable Energy, Grids and Networks*, 32:100938, 2022.

[76] Wenhao Zeng, Zhezhe Han, Yue Xie, Ruiyu Liang, and Yongqiang Bao. Non-intrusive load monitoring through coupling sequence matrix reconstruction and cross stage partial network. *Measurement*, 220:113358, 2023.

[77] LN Sastry Varanasi and Sri Phani Krishna Karri. Enhancing non-intrusive load monitoring with channel attention guided bi-directional temporal convolutional network for sequence-to-point learning. *Electric Power Systems Research*, 228:110088, 2024.

[78] LN Sastry Varanasi and Sri Phani Krishna Karri. Stnilm: Switch transformer based non-intrusive load monitoring for short and long duration appliances. *Sustainable Energy, Grids and Networks*, 37:101246, 2024.

[79] Yanchi Liu, Xue Wang, and Wei You. Non-intrusive load monitoring by voltage–current trajectory enabled transfer learning. *IEEE Transactions on Smart Grid*, 10(5):5609–5619, 2018.

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

[81] Shouxiang Wang, Haiwen Chen, Luyang Guo, and Di Xu. Non-intrusive load identification based on the improved voltage-current trajectory with discrete color encoding background and deep-forest classifier. *Energy and Buildings*, 244:111043, 2021.

[82] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

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

[84] Xiaochao Guo, Chao Wang, Tao Wu, Ruiheng Li, Houyi Zhu, and Huaiqing Zhang. Detecting the novel appliance in non-intrusive load monitoring. *Applied Energy*, 343:121193, 2023.

[85] Bo Liu, Wenpeng Luan, and Yixin Yu. Dynamic time warping based non-intrusive load transient identification. *Applied energy*, 195:634–645, 2017.

[86] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[87] José Alcalá, Jesús Ureña, Álvaro Hernández, and David Gualda. Event-based energy disaggregation algorithm for activity monitoring from a single-point sensor. *IEEE Transactions on Instrumentation and Measurement*, 66(10):2615–2626, 2017.

[88] Ram Machlev, Dmitri Tolkachov, Yoash Levron, and Yuval Beck. Dimension reduction for nilm classification based on principle component analysis. *Electric Power Systems Research*, 187:106459, 2020.

[89] Bogdan Dumitrescu and Paul Irofti. *Dictionary learning algorithms and applications*. Springer, 2018.

[90] JZNA Kolter, Siddharth Batra, and Andrew Ng. Energy disaggregation via discriminative sparse coding. *Advances in neural information processing systems*, 23, 2010.

[91] Vladimir Stankovic, Jing Liao, and Lina Stankovic. A graph-based signal processing approach for low-rate energy disaggregation. In *2014 IEEE symposium on computational intelligence for engineering solutions (CIES)*, pages 81–87. IEEE, 2014.

[92] Bochao Zhao, Lina Stankovic, and Vladimir Stankovic. Blind non-intrusive appliance load monitoring using graph-based signal processing. In *2015 IEEE global conference on signal and information processing (GlobalSIP)*, pages 68–72. IEEE, 2015.

[93] Kanghang He, Lina Stankovic, Jing Liao, and Vladimir Stankovic. Non-intrusive load disaggregation using graph signal processing. *IEEE Transactions on Smart Grid*, 9(3):1739–1747, 2016.

[94] Christy Green and Srinivas Garimella. Analysis of supervised graph signal processing-based load disaggregation for residential demand-side management. *Electric Power Systems Research*, 208:107878, 2022.

[95] Christy Green and Srinivas Garimella. Very low-resolution residential load disaggregation with unsupervised graph signal processing. *Electric Power Systems Research*, 215:108998, 2023.

[96] Anthony Faustine, Lucas Pereira, and Christoph Klemenjak. Adaptive weighted recurrence graphs for appliance recognition in non-intrusive load monitoring. *IEEE Transactions on Smart Grid*, 12(1):398–406, 2020.

[97] Anthony Faustine and Lucas Pereira. Improved appliance classification in non-intrusive load monitoring using weighted recurrence graph and convolutional neural networks. *Energies*, 13(13):3374, 2020.

[98] Han Wang, Jin Ma, and Jianguo Zhu. Identifying household ev models via weighted power recurrence graphs. *Electric Power Systems Research*, 217:109121, 2023.

[99] Robert X Gao and Ruqiang Yan. *Wavelets: Theory and applications for manufacturing*. Springer Science & Business Media, 2010.

[100] Paul S. Addison. *The Illustrated Wavelet Transform Handbook: Introductory Theory and Applications in Science, Engineering, Medicine and Finance*. Institute of Physics Publishing, Bristol, UK, 2002.

[101] Yi-Ching Su, Kuo-Lung Lian, and Hsueh-Hsien Chang. Feature selection of non-intrusive load monitoring system using stft and wavelet transform. In *2011 IEEE 8th international conference on e-business engineering*, pages 293–298. IEEE, 2011.

[102] Hsueh-Hsien Chang, Kun-Long Chen, Yuan-Pin Tsai, and Wei-Jen Lee. A new measurement method for power signatures of nonintrusive demand monitoring and load identification. *IEEE Transactions on Industry Applications*, 48(2):764–771, 2011.

[103] Cesar Duarte, Paul Delmar, Keith W Goossen, Kenneth Barner, and Eduardo Gomez-Luna. Non-intrusive load monitoring based on switching voltage transients and wavelet transforms. In *2012 future of instrumentation international workshop (FIIW) proceedings*, pages 1–4. IEEE, 2012.

[104] Aggelos S Bouhouras, Apostolos N Milioudis, and Dimitris P Labridis. Development of distinct load signatures for higher efficiency of nilm algorithms. *Electric Power Systems Research*, 117:163–171, 2014.

[105] Aggelos S Bouhouras, Paschalis A Gkaidatzis, Evangelos Panagiotou, Nikolaos Poulakis, and Georgios C Christoforidis. A nilm algorithm with enhanced disaggregation scheme under harmonic current vectors. *Energy and Buildings*, 183:392–407, 2019.

[106] Chang Xiong, Zhaohui Cai, Shubo Liu, Jie Luo, and Guoqing Tu. An improved sequence-to-point learning for non-intrusive load monitoring based on discrete wavelet transform. *IEEE Transactions on Instrumentation and Measurement*, 2023.

[107] Hasan Rafiq, Prajowal Manandhar, Edwin Rodriguez-Ubinas, Omer Ahmed Qureshi, and Themis Palpanas. A review of current methods and challenges of advanced deep learning-based non-intrusive load monitoring (nilm) in residential context. *Energy and Buildings*, page 113890, 2024.

[108] J Zico Kolter and Matthew J Johnson. Redd: A public data set for energy disaggregation research. In *Workshop on data mining applications in sustainability (SIGKDD), San Diego, CA*, volume 25, pages 59–62. Citeseer, 2011.

[109] Hafiz Khurram Iqbal, Farhan Hassan Malik, Aoun Muhammad, Muhammad Ali Qureshi, Muhammad Nawaz Abbasi, and Abdul Rehman Chishti. A critical review of state-of-the-art non-intrusive load monitoring datasets. *Electric Power Systems Research*, 192:106921, 2021.

[110] Jack Kelly and William Knottenbelt. The uk-dale dataset, domestic appliance-level electricity demand and whole-house demand from five uk homes. *Scientific data*, 2(1):1–14, 2015.

[111] Christian Beckel, Wilhelm Kleiminger, Romano Cicchetti, Thorsten Staake, and Silvia Santini. The eco data set and the performance of non-intrusive load monitoring algorithms. In *Proceedings of the 1st ACM conference on embedded systems for energy-efficient buildings*, pages 80–89, 2014.

[112] David Murray, Lina Stankovic, and Vladimir Stankovic. An electrical load measurements dataset of united kingdom households from a two-year longitudinal study. *Scientific data*, 4(1):1–12, 2017.

[113] Christoph Klemenjak, Christoph Kovatsch, Manuel Herold, and Wilfried Elmenreich. A synthetic energy dataset for non-intrusive load monitoring in households. *Scientific data*, 7(1):108, 2020.

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

[115] FJIEA Birol. The future of cooling: opportunities for energy-efficient air conditioning. *International Energy Agency*, 526, 2018.

[116] European parliament and council of the European Union. Directive (ue) 2019/944. *Official Journal of the European Union*, 2019.

[117] Sibelga. Mode d'emploi du compteur intelligent sagecom-siconia s211, t211. https://www.sibelga.be/asset/file/82962806-fdba-11ef-89a1-005056970ffd. Online; accessed 28 May 2025.

[118] Sibelga. Mode d'emploi du compteur intelligent landis+gyr e360-1p, e360-3p. https://www.sibelga.be/asset/file/8295dc52-fdba-11ef-992d-005056970ffd. Online; accessed 28 May 2025.

[119] AREWAL. Découvrez votre compteur communicant et profitez de tous ses avantages. https://www.aiesh.be/static/docs/Brochure%20Smart_v5.pdf. Online; accessed 28 May 2025.

[120] Jean-Luc Timmermans and Pierre Henneaux. Active and reactive power sequences for energy disaggregation with deep learning models. In *2024 IEEE PES Innovative Smart Grid Technologies Europe (ISGT EUROPE)*, pages 1–5. IEEE, 2024.

[121] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[122] Huan Chen, Yue-Hsien Wang, and Chun-Hung Fan. A convolutional autoencoder-based approach with batch normalization for energy disaggregation. *The Journal of Supercomputing*, 77(3):2961–2978, 2021.

[123] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[124] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *Proceedings of the european conference on computer vision (ECCV)*, pages 464–479, 2018.