Delft University of Technology
Master's Thesis in MSC Embedded Systems

Semi-supervised Energy Disaggregation Framework using General Appliance Models

Bontor Humala

Semi-supervised Energy Disaggregation Framework using General Appliance Models

Master's Thesis in MSC Embedded Systems

Embedded Software Section
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Bontor Humala
BontorHumala@student.tudelft.nl

15th January 2018

Author
Bontor Humala (BontorHumala@student.tudelft.nl)

Title
Semi-supervised Energy Disaggregation Framework using General Appliance Models

MSc presentation
22nd January 2018

Graduation Committee

Prof.dr. Koen Langendoen	Delft University of Technology
Dr. RangaRao Venkatesha Prasad	Delft University of Technology
Dr. Alessandro Bozzon	Delft University of Technology

Abstract

Providing detailed appliance-level energy consumption information helps consumers to understand their usage behavior and encourages them to optimize their energy usage. Non-intrusive load monitoring (NILM) or energy disaggregation aims to estimate appliance-level energy consumption data from the aggregate consumption data of households. NILM algorithms can be broadly classified into supervised and unsupervised (or semi-supervised) techniques. The former requires a large amount of prior data for each appliance and the latter relies on manual tuning of models of appliances based on some metadata information. While there is a significant interest from academia and industry, NILM techniques are still not adopted widely across households. This is mainly because the techniques developed for one household cannot be generalized and applied in other households (*applicability*), require tremendous manual-tuning to apply across households (*scalability*), and cannot run in *real-time*. To overcome the above issues, we propose a novel semi-supervised energy disaggregation framework – UniversalNILM. The key idea of UniversalNILM is to model appliances in a few (3-10) training houses, which has detailed appliance-level data and transfer this learning to test houses (*blind disaggregation*), which has only aggregate house consumption data to derive fine-grained appliance energy consumption. To this end, we develop an automated appliance modeling technique that creates general appliance models across various appliance brands and models. The general appliance models are analytical models which describe power consumption of each appliance. These general appliance models are then fine-tuned automatically on test houses to accurately disaggregate the energy consumption in real-time. To test the robustness of UniversalNILM, we empirically evaluated it across three publicly available real-world datasets. We show that the general appliance models learnt on a few households is able to accurately disaggregate on unseen test houses in the same dataset, as well as unseen houses from different datasets. This is the first work in NILM which is able to perform disaggregation across datasets. Another improvement is that UniversalNILM outperforms the reported accuracy from both state-of-the-art supervised and unsupervised NILM techniques.

Preface

In 2012, I co-founded an Internet of Things (IoT) company and become interested in IoT data analytics, especially energy data (electricity, gas, water). At one point during master study, I started to be aware of energy disaggregation algorithms. It is a perfect example of data analytics on IoT data. Today, smart meters are being installed in different countries and are projected to penetrate 50% of global market by 2022. However, customers are yet to benefit from smart meter data and energy disaggregation is not widely adopted yet. This opportunity motivated me to do this thesis which aims to create an applicable energy disaggregation algorithm. I believe that IoT will be an important infrastructure and data analytics algorithms like energy disaggregation will help users to understand IoT data effectively.

I am grateful to Akshay, who have provided me insightful guidance and discussion about energy disaggregation and machine learning. During his busy job as a researcher in Microsoft, he makes himself available for consultations via Skype regularly. I am also thankful to VP for his important guidance during thesis, both on technical and non-technical matters. Both VP and Akshay are truly supportive, especially during and after my father's passing. I cannot imagine how can I finish my thesis without their care and kindness.

A special gratitude to LPDP Indonesia who have provided the fund for me to pursue my master at TU Delft.

Finally, my deepest gratitude to my mother and late father for their never ending love and support.

Bontor Humala

Delft, The Netherlands
15th January 2018

Contents

Chapter 1

Introduction

We first introduce the the background of the topic energy disaggregation in Section 1.1. In section 1.2 we discuss the limitations of energy disaggregation at present. This will lead us to explain the research goal in Section 1.3 considering the limitations. We also enlist the contributions of this thesis. Finally, the organisation of this thesis is provided in Section 1.4

1.1 Importance of energy disaggregation

Electricity consumption in residential and commercial buildings plays a huge role in worldwide energy demand and carbon emission. Worldwide total energy consumption in residential and commercial buildings is estimated to be 30-40% of generation [21]. Furthermore, electricity generation for residential and commercial buildings accounts for one-fifth of fuel-based CO_2 emissions in the world [1]. Thus efficient electricity consumption in buildings is very important for global sustainability.

Smart metering comprises of networked electricity meters that are deployed to measure, collect and analyze total household energy consumption at consumer premises. The US and EU aims to replace at least 80% of the traditional meters with smart meters by 2020 [3]. These smart meters act as Internet of Things (IoT) devices, which enable bidirectional communication between the consumers and utility providers for a wide variety of services. Some of the services include providing immediate feedback on power usage, power quality, and pricing details.

Intelligent data analytics on smart meter data can provide feedback to consumers on daily average, monthly average and historical energy consumption information. Recent studies have shown that 5-15% of electricity consumption can be reduced with better real-time information of appliance level consumption statistics as opposed to total household [8,10,22]. Moreover, fine-grained appliance information can also be used to identify faulty or malfunctioning appliances that consume more energy than they should. Consequently, occupants know where the energy is being wasted. Home automation systems coupled with smart meters can now provide feedback on energy usage by monitoring household energy consumption.

The most common way of obtaining appliance level information is by deploying sensors for each appliance. Such a deployment is intrusive, cumbersome to maintain, and has a high cost. Alternatively, *non-intrusive load monitoring* (NILM) or *energy disaggregation algorithms* aim to break down a household's aggregate energy consumption into individual appliances energy consumption [9]. NILM [1] techniques are gaining popularity due to large-scale smart-meter deployments to obtain a households aggregate energy consumption and inference algorithms proposed for energy disaggregation [29].

1.2 Challenges in NILM

While NILM techniques seem attractive, they do not work *as is.* The current techniques require either huge training data in each individual house so as to model the appliances or significant metadata information about occupants, number of appliances, or type and model of the appliance in a household. The former approach is known as supervised NILM, where prior data for each appliance is required. This enables the supervised methods to be accurate, but challenging because sensors need to be deployed to obtain the prior data. The latter is known as unsupervised or semi-supervised NILM. Due to lack of appliance-level data, these approaches are less accurate than the supervised methods.

There still exist several challenges preventing NILM techniques to be widely adopted across households: (i) Applicability: While both supervised and unsupervised approaches do infer individual appliance consumption in a household, the techniques developed for one household cannot be generalized and applied in other households. (ii) Scalability: NILM techniques proposed till date focus and evaluate only on one dataset or a few houses in a dataset. Scaling to other datasets or households in other locations require tremendous manual-tuning and is impractical. (iii) Real-time: Majority of the NILM algorithms cannot run in real-time due to the computational complexity and the ones that do, utilize cloud services. This raises several issues related to cost and privacy. (iv) Metadata information: Detailed information of the household, number of occupants, number of appliances, their brand and model, and other information is required by NILM algorithms. This requires deployment of additional sensors or survey information, consequently making the system cumbersome to use.

[1]In this thesis, we use the term 'energy disaggregation' and 'NILM' interchangeably.

1.3 Research goal and contributions

All challenges mentioned in the previous section lead us to the research goal. The main goal of this thesis is to develop a universal energy disaggregation framework that can, (i) be applied to any household in any neighborhood/-city without additional sensor deployment; and (ii) run on an embedded system (such as Raspberry PI) in real-time, and locally at the household to derive appliance-level energy consumption information.

The main contributions of this thesis are:
(i) We develop an automated appliance modeling technique that creates general appliance models across various brands and models of appliances.
(ii) We propose a novel fine-tuning technique to tune the general appliance models based on the aggregate energy data from the test houses.
(iii) We present a trainingless real-time disaggregation system based on Combinatorial Optimization to derive appliance-level information using only smart meter data.
(iv) We perform extensive experimentation across datasets and show that the proposed solution outperforms the state-of-the-art supervised and unsupervised NILM techniques.

1.4 Organisation

This thesis is organised as follow. Chapter 2 explains appliance load modeling and various NILM algorithms. We also discuss opportunities for a novel research in that chapter. In Chapter 3 we explain a proposed solution to answer the research goal. The proposed solution consists of two main components. The appliance modeling component is explained in Chapter 4. The disaggregator component is explained in Chapter 5. These components are implemented and the evaluation results are discussed in Chapter 6. Finally, we draw conclusions on the research goal and propose possible future works in Chapter 7.

Chapter 2

NILM and Appliance Modeling

This chapter starts with an explanation about appliance modeling in Section 2.1. Numerous NILM algorithms, both supervised and unsupervised, have been proposed in the literature to derive fine-grained appliance-level information. We cover these algorithms in Section 2.2. Then, we discuss opportunities for a novel research on NILM in Section 2.3.

2.1 Appliance load modeling

Appliance load modeling is a key step for energy disaggregation, for both supervised and unsupervised NILM. Appliance models can vary from a simple on-off model with fixed power states to a multi-state model with varying power levels. We now review the state-of-the-art works about appliance modeling.

Barker et. al [5] showed that appliances power data can be manually categorized into several models based on their power signatures. The intuition behind their idea is that appliances can be categorized by their load types which are resistive, inductive, non-linear, or mixture of them (composite). They proposed different mathematical equations for each load type such as fixed power levels (resistive), growing or decaying power (inductive), stable min/max or random power (non-linear). However, this work requires significant manual efforts to classify appliances into different load types. Figure 2.1 illustrates examples of appliances signatures that are categorized into different load types.

An extension of these manually-derived models was proposed by Iyengar et. al in [11]. They developed an automatic non-intrusive model derivation (NIMD) method which can reproduce appliance power data with high similarity. However, the model only work for an appliance with a specific brand and model. For example, there will be three different mathematical models for washing machine brand A model 1, brand A model 2, and brand B model 1.

In general, all the appliance models developed hitherto focused on modeling a specific type of appliance in a dataset. These models are specific to a household and require manual/domain expert knowledge to classify them. UniversalNILM employs a systematic approach to model both simple and composite appliance, that can work across households and datasets. In this thesis, there are two contexts for the term 'model'. The first context is the mathematical model for an appliance load. The word 'model' in this first context occurs alone in a sentence. The second context is the appliance model from a specific brand/manufacturer. The word 'model' in this second context always occurs together with the word 'brand' in a sentence.

2.2 NILM algorithms

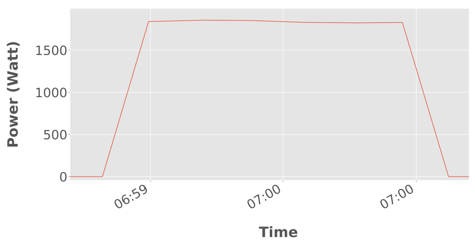
Numerous NILM algorithms, both supervised and unsupervised, have been proposed in the literature to derive fine-grained appliance-level information. We first provide details on the existing algorithms and then discuss opportunities for a novel research on NILM.

2.2.1 Supervised disaggregation

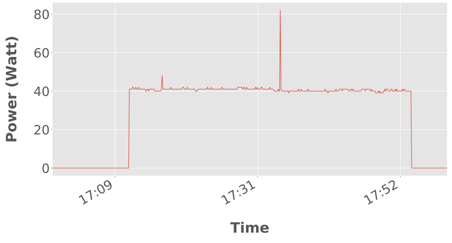
PowerPlay [6] aims to track the electrical power consumption of selected appliances in a household. PowerPlay uses offline modeling and feature extraction in order to obtain identifiable features for each appliance. Finally, several feature detectors (one for each load) operate on a sliding window against the smart meter data. However, PowerPlay can only track a few (1-3) predefined appliances in a household as it fails to handle the complexity when the number of appliances increases. A recent work *viz.,* SparseNILM [18] achieves high disaggregation accuracy (>90%) with 5 appliances. It can also disaggregate a large number of appliances (>18 loads), although the accuracy in this case is not reported. It uses prior data to obtain the appliances states and build a super-state Hidden Markov Model (HMM). Finally, the HMM matrices are used together with a Viterbi algorithm to disaggregate the smart meter data. Due to the complexity in collecting fine-grained appliance data at each household, supervised techniques have limited applicability at large scale.

2.2.2 Unsupervised disaggregation

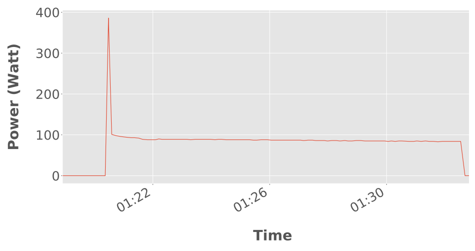
Unsupervised disaggregation is sometimes called blind disaggregation. This is because they do not have appliance-level data in test houses (blind). Unsupervised disaggregation algorithms rely on aggregated smart meter data.



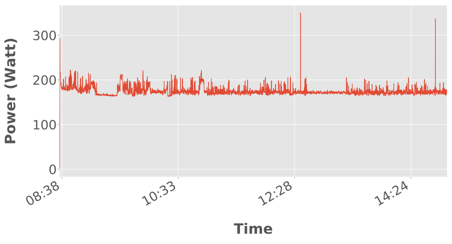
(a) Clothes iron, resistive



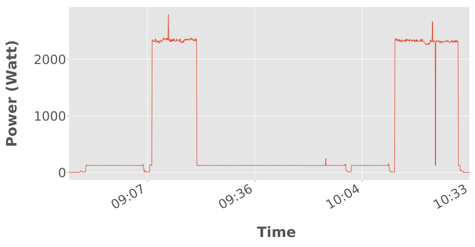
(b) Lamp, resistive



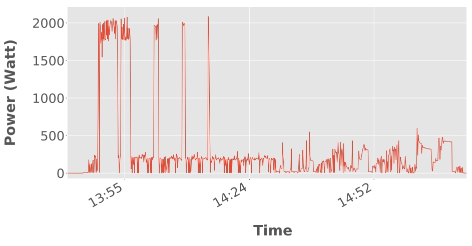
(c) Fridge, inductive



(d) Computer, non-linear



(e) Dish washer, composite



(f) Washing machine, composite

Figure 2.1: Examples of categorized appliances signatures based on the type of their load(s)

Parson et. al [23] developed an unsupervised energy disaggregation method using general appliance models. It models an appliance as a variant of Hidden Markov Model (HMM) called difference HMM. During deployment, these general appliance models are then tuned for periods when only a single appliance is turned on. The drawback is that it requires a domain expert to manually develop the difference HMM models. Furthermore it is only evaluated on the REDD dataset and the applicability of these models on different datasets is not studied.

CFHSMM is another unsupervised energy disaggregation algorithm that uses general appliance models [14]. It develops an HMM model and its parameters from appliance data in a house. Then it tunes the parameters in the test houses using Expectation-Maximization algorithm. Finally, it estimates the appliances states using simulated annealing algorithm. However, CFHSMM is only evaluated using two-state appliances like television, refrigerator, and game console. It is not tested on complex and multi-state appliances like washing machine and dishwasher, which often consume most of the energy in a house.

Neural NILM [12] is one of the first works to suggest the usage of neural network in NILM. The problem with Neural NILM, as in other deep neural network algorithms, is that the training phase requires a massive amount of data and a lot of parameters to be trained. Furthermore, the paper also mention that the disaggregation algorithm itself is too computationally expensive to be run on an embedded processor inside a smart meter or an in-home display.

SBNMF [19] is based on non-negative matrix factorization algorithm where the smart meter data is factorized into basis and activation matrices. The basis matrix is based on appliance power data and the activation matrix holds binary values that indicate which appliance is active. SBNMF does not require any model and can work with a standard smart meter. However the disaggregation accuracy is low ($< 20\%$), especially for appliances with a complex usage pattern like television and washing machine.

BOLT [17] performs unsupervised disaggregation on current waveforms using neural network and binary matrix factorization method. A neural network is employed to find the basis and activation matrices. The power consumption for each appliance is estimated naively, which assumes that each appliance only has two states (ON and OFF state) with predefined power value in each state. Another drawback is that it uses high frequency current data which is impractical since a standard smart meter usually only has low frequency power data. Furthermore, all of these unsupervised algorithms focus only on one dataset or a few houses in a dataset. Their scalability to other datasets or households in other locations has not been proven.

Table 2.1: Comparison with SoA

Goals	S-SoA	U-SoA	UniversalNILM
Accurate	Y	Y	Y
Trainingless	N	Y	Y
Simple	Y	Y	Y
Real time	Y	N	Y
Various appliance	Y	Y	Y
Scalability	N	N	Y

2.3 Research opportunity

Based on 2.2, we can see that supervised algorithms are not applicable. In terms of applicability, unsupervised algorithms are generally preferable since they do not need appliance-level data in the test houses. While there is significant interest in unsupervised NILM techniques, there still exist issues, (i) not scalable to houses in other datasets or locations, (ii) some of them require smart meter with high sampling rate, (iii) the disaggregation accuracy is very low, and (iii) too computationally expensive to run in real-time and locally in a household.

We propose *UniversalNILM* – a semi-supervised energy disaggregation framework. UniversalNILM overcomes all the above drawbacks and can work with any off-the-shelf smart meter data, derives appliance-level data with high accuracy and can run in real-time. Zeifman et. al [28] introduces six criteria for a good energy disaggregation method. We compare the proposed UniversalNILM with the state of the art (SoA) works viz., supervised (*S-SoA*) based on [18] and unsupervised (*U-SoA*) based on [15]. Table 2.1 shows the comparison against the six criteria.

Chapter 3

Framework for UniversalNILM

At the end of Chapter 2, we address the research goal by proposing UniversalNILM. In this chapter, Section 3.1 explains the idea of the proposed solution, UniversalNILM. Section3.2 describes the design of UniversalNILM.

3.1 Proposed model

The key idea of UniversalNILM is to model appliances in a few (3-10) training houses, which have detailed appliance-level data and transfer this learning on to the rest of the test houses, which have only aggregated smart meter data to derive fine-grained appliance energy consumption information.

Consider a neighborhood or a city where a few households (say 1-5) are instrumented with smart plugs (to collect appliance-level energy informa-

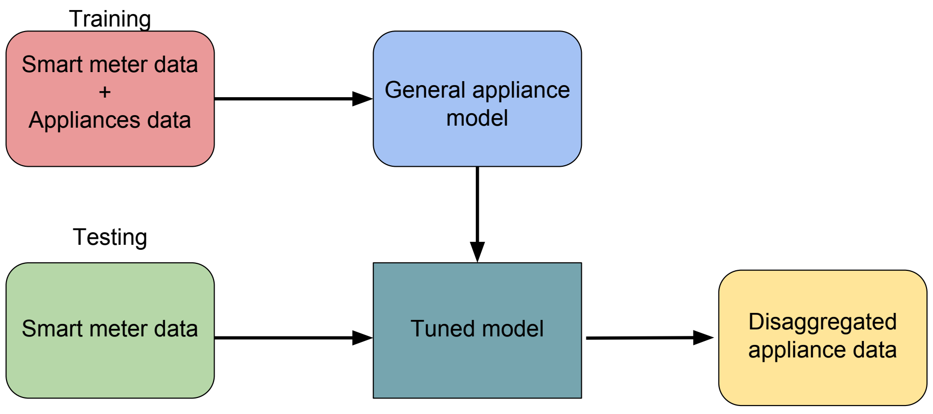


Figure 3.1: Overview of UniversalNILM.

tion) [2]. UniversalNILM uses the fine-grained data from these few households to learn model parameters for each appliance. These models after learning are then applied to any household in the neighborhood or city, which has a smart meter (collecting aggregate household data). The hypothesis here is that models of appliances learnt on a few training households can be fine-tuned on unseen test households systematically with minimal metadata information. These models of appliances are analytical models which describe power consumption of each appliance mathematically.

Consider a neighborhood/city with ten thousand households each equipped with a smart meter to measure total energy consumption of the house. It is infeasible in terms of cost and maintenance to deploy a smart plug for each appliance in all these households to provide appliance-level feedback. To this end, lets us assume that there are a few households (say 10), in which each appliance is equipped with a smart plug [2] to collect fine-grained appliance-level data along with smart meter data. These houses can be instrumented with smart plugs by an energy utility company or as pilot deployments. UniversalNILM uses appliance-level data from these 10 households to derive general appliance models, that learns the characteristics of each appliance in these households. UniversalNILM then applies these general appliance models across each test house in the city to derive fine-grained appliance information. While applying the general appliance models[1] on a test house, UniversalNILM performs fine-tuning in order to adapt to varying appliance brand/model[2]. In this thesis, we present a systematic and automated way of developing general models of appliances from the houses considered for training. Further, we propose how to fine-tune these models based on a test house aggregate data.

To evaluate the efficacy, we applied UniversalNILM on three publicly available datasets *viz.,* REDD dataset in the US [16], UK-DALE dataset in the UK [13], and DRED dataset in the Netherlands [27]. The UniversalNILM framework is made publicly available[3] for the community to support additional analysis.

3.2 System Design

UniversalNILM is a semi-supervised energy disaggregation framework where appliance models are learned on a few training households and then transferred to a large set of unseen test houses. UniversalNILM consists of two main components, which are the model builder and the disaggregator. The model builder aims to build general appliance models based on the training data from a few households as shown in Figure 3.2. As mentioned

[1]These are the mathematical models of appliances
[2]Model of an appliance from a specific vendor/manufacturer
[3]http://www.st.ewi.tudelft.nl/ akshay/dred/

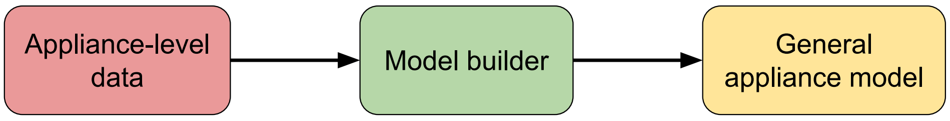


Figure 3.2: Model builder for general appliance models on a training house.

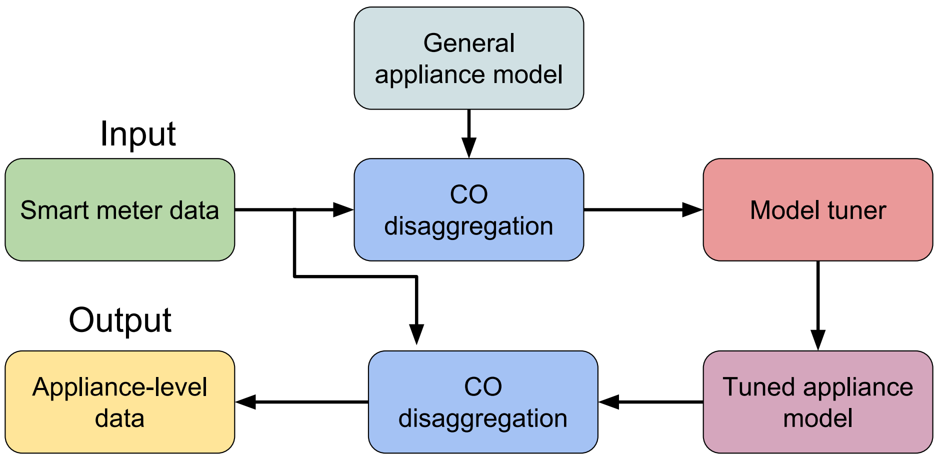


Figure 3.3: Disaggregator component on a test house.

before in Section 3.1, the general appliance models are analytical models which describe power consumption of each appliance mathematically. This component is executed offline, for example by an energy utility or a data analytics company, that instruments a few households to generate general appliance models.

The disaggregator then uses these general appliance models to perform online disaggregation in each household without any training. Disaggregator consists of two main modules: the general model based disaggregation (CO disaggregation) and the model tuner as shown in Figure 3.3. The CO disaggregation component uses general appliance models to disaggregate smart meter data in test houses. This provides initial appliance-level information for the test house. However, since the appliance brand and model may vary from training to test houses, the general appliance models need to be tuned. The model tuner component tunes the general appliance models based on the test house aggregated smart meter data for accurate appliance-level energy disaggregation.

3.3 Advantages of the framework

The UniversalNILM framework separates training and testing into two entirely different components which makes it modular. This modularity is important since it means that we can easily try out different features, algorithms, and evaluation scenarios. As mentioned in Section 2.3, the current NILM techniques are not scalable to houses in other datasets or locations.

Trying out different evaluation scenarios definitely helps us to know the scalability of our algorithm. For example, training and testing using different datasets tells us whether if the algorithm can work in any houses or locations. Since the training and testing components are modular, we can easily play around with the training sets, the testing sets, or the number of appliances which we will discuss later in Chapter 6.

Trying different algorithms and features is also important to develop an accurate NILM algorithm. For example, the amount of appliance-level data at this moment is limited which is the reason why deep neural network techniques are not used more often in NILM. However, because there are a lot appliances and each of them has a rich power consumption signature, developing a feature extractor manually may never be good enough. Therefore, a deep neural network technique could be a better fit to NILM, given that there is enough appliance-level data.

During development, we have tried several deep neural network architectures (convolutional neural network/CNN and LSTM) to replace the analytical model. In this case, the modularity of the model builder allows us to take appliance signatures and feed them to the deep neural network. Furthermore, the disaggregator is also modular which allows us to replace the CO-disaggregation with the deep neural network. Additionally, we also tried FHMM to replace CO-disaggregation. In the end, we keep the analytical model as the final proposed solution. Nevertheless, these facts show that the modularity of the framework allows us to try different algorithm and features in order to achieve a good accuracy and applicability.

3.4 Summary of the framework for UniversalNILM

The basic idea of UniversalNILM is to model appliances in a few training houses (3-10) using appliance-level data and transfer this learning to the rest of the test houses which have only aggregated smart meter data. To this end, the framework for UniversalNILM consists of two main components which are the model builder and the disaggregator. The models for appliances are analytical models which describe the power consumption of each appliance mathematically. This component is executed offline by an energy utility or a data analytics company. The disaggregator tunes the general appliance models using only aggregated smart meter data in the test houses. Finally, it uses the tuned appliance models to perform online disaggregation in each household without appliance-level data.

Chapter 4

General modeling of appliances

The goal of the model builder is to accurately derive appliance signatures that can be used to build general appliance models. We give a brief explanation of the technique used in Section 4.1 Then we enumerate on the steps involved in Section 4.2-4.7.

4.1 Modeling - An overview

For each appliance, the model builder first determines the active period and state of the appliance. This comprises several steps, from the active period extraction to the cycle removal. Model builder then applies various curve and distribution fitting algorithms to model the appliance in active state mathematically. Finally, the model parameters for all appliances in the training data are clustered to determine the best model parameters for each appliance across various appliance brands and models. Figure 4.1 shows the detailed steps of the appliance modeling.

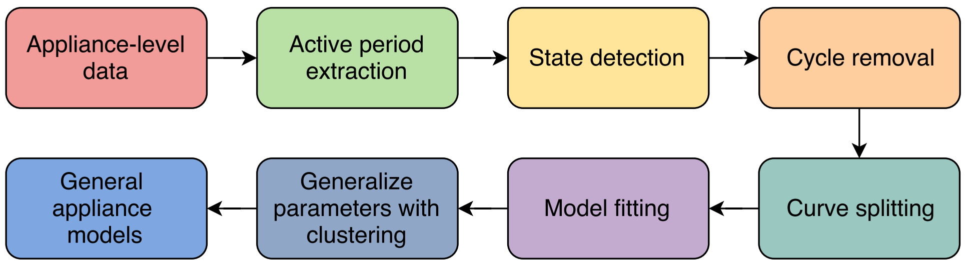


Figure 4.1: Steps involved in general appliance modeling.

4.2 Active period extraction

Appliance-level data consists of *active* and *inactive* periods. This step splits the appliance-level data when it draws electrical power over a certain *power threshold* and the results are called *active period.* Note that it is possible for an appliance to draw no electrical power for some time even though it is active. For example, when a washing machine stops spinning and prepares to rinse. Therefore, if there are short inactive periods between multiple active periods, then they are merged together into one active period. We refer to the maximum short inactive period as *inactive duration threshold.* In this work, we use 10 minutes as the *inactive duration threshold* for all appliances. We take 10 minutes based on our manual observation that complex devices such as dish washer and washing machine take up to 10 minutes inactive period.

4.3 State detection

An appliance may consist of multiple operational states such as on, off, and idle. For example, a washing machine can be in a washing, rinsing, or spinning state. Each state has a different electrical power signature. The state detection aims to split an active period of appliance into several smaller chunks. Ideally these smaller chunks indicate different states of the appliance.

To detect state changes we use – approximate entropy ($ApEn$) – a technique used to quantify the amount of regularity and the unpredictability of fluctuations over time-series data [25]. We run a sliding window over an active period and calculate $ApEn$ for each position of the sliding window. An entropy peak indicates a state change and finally, the active period is split at every peak of entropy. These smaller chunks are called *state chunks* since they represent different states in an appliance active period.

4.4 Cycle removal

In an appliance operational cycle (i.e., appliance ON to appliance OFF state), an appliance load may be active many times. For example, a washing machine may spin and stop several times during a wash state. Therefore the same electrical power signature will be repeated multiple times. This step splits a state chunk of an appliance into smaller chunks. These smaller chunks are called *cycle chunks* since they represent different cycles of a state in an appliance active period. A cycle chunk represents the electrical power consumption of a state of an appliance.

At this stage, a cycle chunk relates to a unique state of an appliance. Consequently, we should be able to perform model fitting at this point. However, a complex appliance might include more states that have not been identified by the state detection (Section 4.3) and cycle removal steps. Therefore, the cycle chunk of such appliance is also complex. To overcome this problem, the cycle removal identifies if the cycle chunk is simple or complex, by analyzing the time duration of each cycle chunk and its power consumption. If the time duration is too long and the variance in power consumption is too high, then the cycle chunk is considered complex (i.e. complex cycle chunk).

4.5 Curve splitting

If the cycle removal (Section 4.4) indicates that the cycle chunk is complex, the curve splitting is employed to split the complex cycle chunk further. The curve splitting uses a moving window to calculate standard deviations along the complex cycle chunk. If the standard deviation at a point is higher than the standard deviation of the whole complex cycle chunk, it suggests that the particular point is different than the most of the chunk and therefore is considered as an edge or end of the chunk. The complex cycle chunk is then cut at the edges, resulting in the actual single cycle chunk. At this stage, all appliance power data (simple or complex) can be split into single cycle chunk corresponding to a state of an appliance. The next step is to model each single cycle chunk mathematically.

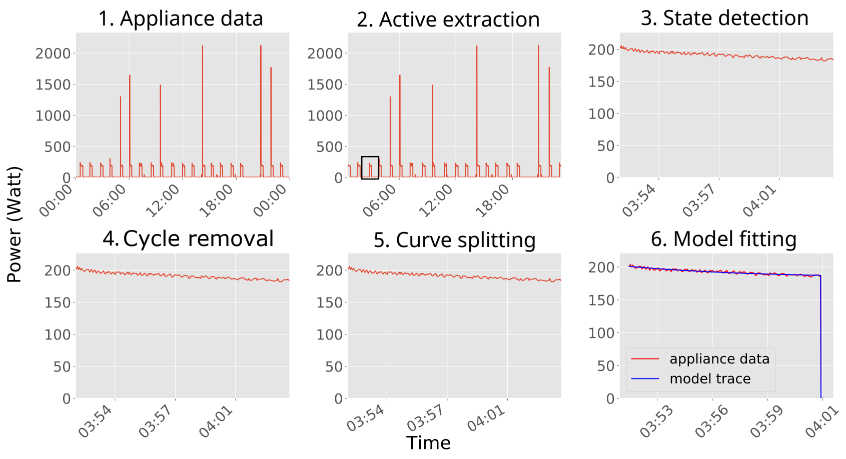
4.6 Model fitting

The model fitting tries to fit various curves and distributions to the cycle chunks that are previously obtained to get several mathematical models of appliances. The output of model fitting is a set of models and parameters for each appliance type as shown in Table 4.1. For each cycle chunk, all mathematical models are evaluated and the model that is most similar to the original trace is then selected (least root mean square error). Finally, for each appliance, the corresponding models and its parameters are stored. Note that, each appliance might include multiple models, for example, a fridge can include on-off decay (corresponding to compressor ON state) and on-off model (corresponding to fridge light ON state).

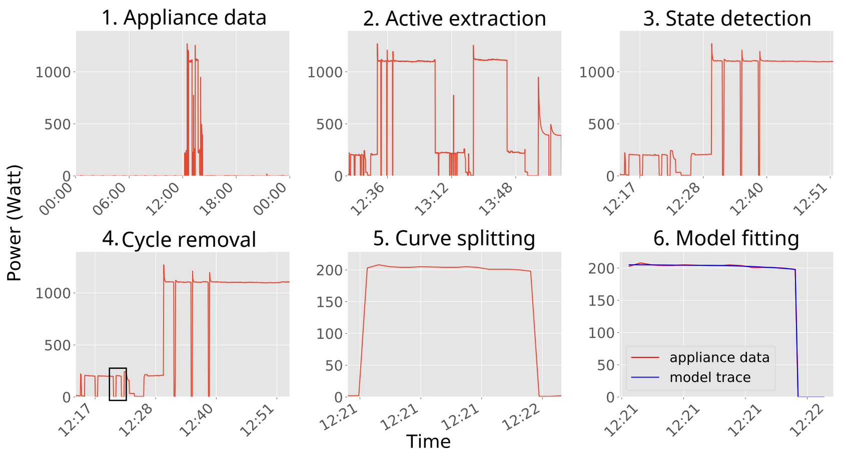
Figure 4.2 illustrates all the steps from the active period extraction until the model fitting. The plots are based on real appliances data from REDD dataset house 1. Note that since the dishwasher is a complex appliance, two different single cycle chunks of dishwasher are shown on figure 4.2b and 4.2c.

4.7 Generalize parameters with clustering

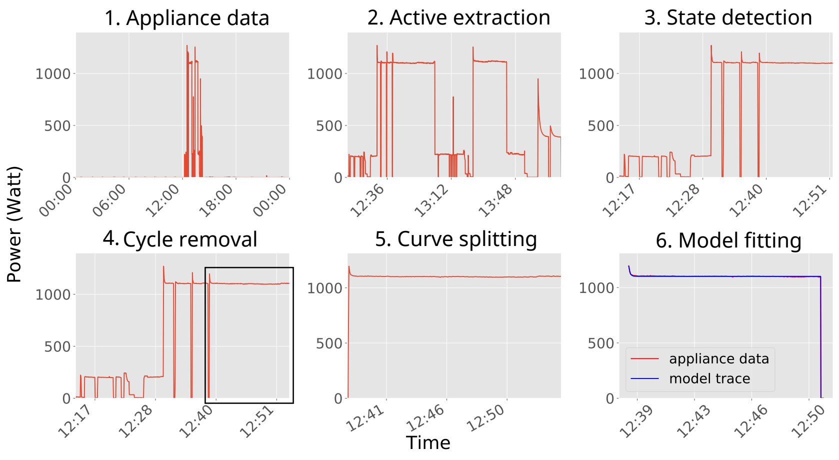
During training there could be multiple brands and models of appliance of the same type. This could result in huge number of models and parameters



(a) Fridge



(b) Dishwasher, load 1



(c) Dishwasher, load 2

Figure 4.2: Step-by-step output of model builder for fridge and dishwasher appliance.

Table 4.1: Different models and parameters for various loads.

Model	Mathematical model	Parameters
On-off	$ax + b$	a, b
On-off decay	$ae^{(-bx)} + c$	a, b, c
On-off growth	$ae^{(bx)} + c$	a, b, c
Random range	$Norm(\mu, \sigma^2)$	μ, σ^2
Stable min/max	$Gamma(\alpha, \beta, \mu)$	α, β, μ

for each appliance. Therefore clustering is performed to identify the most suitable models and parameters for each appliance. We apply a density-based clustering algorithm – DBSCAN – to develop a general model for each type of appliance.

Since there are six types of mathematical model as shown in Table 4.1, the clustering algorithm is executed for each model per appliance. Parameters which have similar values in a mathematical model are clustered together. The final general model for an appliance consists of several cluster mean values for each mathematical model type. Therefore, each cluster mean value is a parameter of the general model.

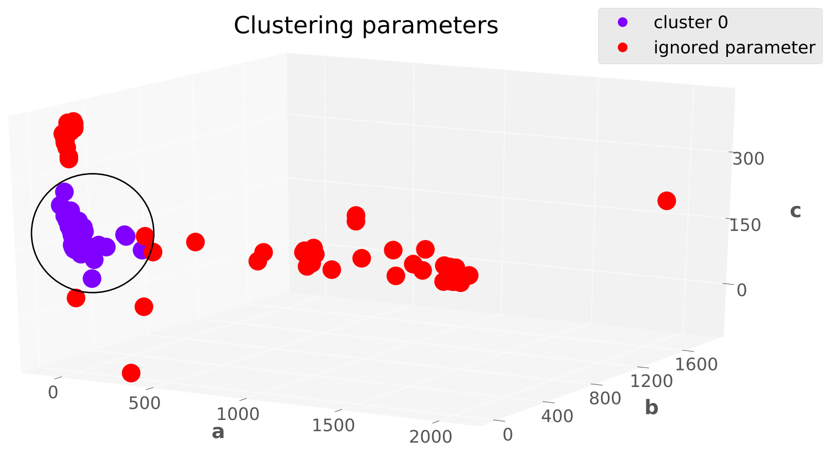
Figure 4.3 illustrates the clustering parameters for fridge and dish washer based on the appliance-level data from REDD houses 1 and 2. For simplicity, only clustering for decay model is shown. From Figure 4.3, a general decay model for fridge therefore has one model per cluster and general decay model for dish washer has two models per clusters. Note that the red dots are ignored by the clustering algorithm since they rarely occur.

4.8 Summary of general appliance modeling

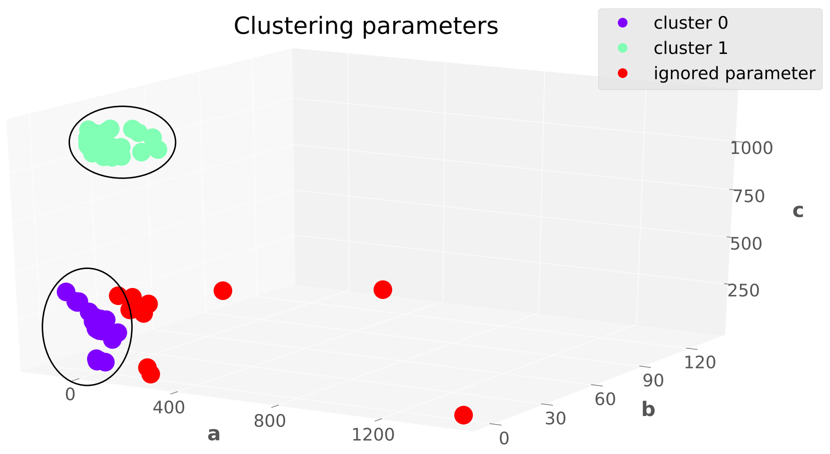
The general appliance modeling starts with the idea that an appliance consists of multiple operational states. Each state has a unique electrical power signature which can be modeled mathematically. The model builder extracts the unique signatures using several steps (the active period extraction until curve splitting steps) – finding a model and parameters for each signature using the model fitting, and finally finding general appliance models using a clustering algorithm.

At this stage, for each appliance we have the possible states and the corresponding models and parameters. For example, a washing machine has OFF, pump ON, and motor ON states where the pump ON state corresponds to random range and on-off models, similarly motor ON state corresponds to random range and on-off decay models. The general appliance models includes the final models and its parameters for each state of an appliance.

The disaggregator component of UniversalNILM uses these general appliance models to disaggregate smart meter data (without any appliance-level data) in the test house from the same dataset (blind disaggregation, same dataset) and different datasets (blind disaggregation, different dataset).



(a) Fridge, decay model



(b) Dishwasher, decay model

Figure 4.3: Clustering parameters towards developing general model for each appliance. Each black circle denotes a cluster and each mean values of a cluster is a parameter of the general appliance model

Chapter 5

Energy Disaggregation

The objective of the disaggregator component is to employ the general appliance models on a test house with aggregated smart meter data and derive appliance-level energy consumption information.

To this end, the disaggregator consists of two modules *viz.,* disaggregation based on general model and model tuner as shown in Figure 3.3. The disaggregation based on general model is explained in Section 5.1. Section 5.2 explains the model tuner.

5.1 General model based disaggregation

As mentioned earlier, in each test house we only have the aggregated energy consumption information of the entire household (aggregated smart meter data). From the general appliance models, we know that the corresponding states and power consumption of each appliance. General model based disaggregation uses these information to disaggregate the aggregated smart meter data of the test house. There are two broad steps, (i) first, we need to identify the regions and extract chunks where one or more appliance could be active in the aggregated smart meter data (super-state detection) and (ii) second, assign each chunk to a valid appliance state based on the general model (CO disaggregation). Since we use CO and general model to disaggregate we call this CO-Disaggregation() as shown in Algorithm 1.

5.1.1 Detection of super-states

In the test house, there are only the aggregated smart meter data and the list of appliances in the household. In order to apply our mathematical models on the smart meter data, we first split the whole smart meter data into several chunks and try to find the most similar model for each chunk. Each chunk should represent a cycle chunk, because that is the base of our general appliance models as mentioned in 4.6. Note that multiple appliances can

Algorithm 1 General model based combinatorial optimization

```
1: function CO-Disaggregation($smart\_meter, model$)
2:     $ss \leftarrow$ GetSuperStates($smart\_meter$)
3:     $i \leftarrow 0$
4:     Loop iter
5:         $j \leftarrow 0$
6:         Loop superstates
7:             $disagg[j] \leftarrow$ CO-LeastRMSE($ss[j], model, disagg[j-1]$)
8:             if $j = length(ss)$ then
9:                 break Loop superstates
10:            end if
11:            $j \leftarrow j + 1$
12:        End Loop
13:        if $i = max\_iter$ or $length(ss) = 0$ then
14:            break Loop iter
15:        end if
16:        $i \leftarrow i + 1$
17:    End Loop
18:    return $disagg$
19: end function
```

be active simultaneously in the aggregated smart meter data. Thus, each identified chunk is called as *super-state*, which is basically a combination of appliance states.

The detection of super-states is similar to the state detection described in 4.3. The key difference is that we use sample entropy (*SampEn*) instead of approximate entropy in order to detect super-state changes. The reason is that *SampEn* is independent of data length and less sensitive to data variance than approximate entropy [26]. This makes *SampEn* suitable for smart meter data which has higher data variance and length compared to appliance active chunk. Thus, at the end of this step, the aggregated smart meter data is split into chunks of super-states as described in Algorithm 2. The next step is to disaggregate these super-states into corresponding appliance states.

5.1.2 Combinatorial Optimization disaggregation

Once we have super-state chunks from super-state detection, we perform CO based disaggregation on each chunk using appliance-level data which is generated from our general models. Combinatorial Optimization (CO) finds the optimal combination of appliance states, which minimizes the difference between the sum of predicted appliance power and the observed aggregated power. Let $\hat{y}_t^{(n)}$ be the estimated energy consumed and $y_t^{(n)}$ be the actual

Algorithm 2 Super-states detection

```
function GetSuperStates($smart\_meter$)
    $i \leftarrow 0$
    $window \leftarrow 20$
    Loop entropy window
        $sm\_window \leftarrow smart\_meter[i : window + i]$
        $entropy[i] \leftarrow$ SampEntropy($sm\_window$)
        if $i = length(smart\_meter)$ then
            break Loop entropy window
        end if
        $i \leftarrow i + 1$
    End Loop
    $super\_states \leftarrow$ FindPeaks($entropy$)
    return $super\_states$
end function
```

energy demand of each appliance n at time t. $\overline{y}_t$ represent the aggregate energy reading of the household. The ground truth state of an appliance is represented by $x_t^{(n)} \in Z \geq 0$ and $\hat{x}_t^{(n)}$ represents the appliance state estimated by the disaggregation algorithm. CO based disaggregation can now be defined as,

$$\hat{x}_t^{(n)} = \arg\min_{\hat{x}_t^{(n)}} \left| \overline{y}_t - \sum_{n=1}^{N} \hat{y}_t^{(n)} \right| \tag{5.1}$$

where N is the set of all appliances in the household and t is the current time period. The predicted energy consumption of an appliance $\hat{y}_t^{(n)}$ is then mapped to the closest appliance state $\hat{x}_t^{(n)}$. More details on CO can be found in [27].

Thus, CO finds the combination of appliance states from general appliance models that closely matches to the detected super state in the aggregated smart meter data. One drawback is that the computational complexity in CO increases exponentially with the number of appliances. The computational complexity for T time is $O(T|S|^{|N|})$ where $|S|$ is the number of appliances states and $|N|$ is the number of appliances in the household. Since there are multiple brands and models of a type of appliance, $|S|$ can be large and CO disaggregation becomes computationally intractable.

To reduce the computational complexity, we employ two techniques. The first is *priority combination* [27]. Priority combination is basically the set of appliances that are assumed to be currently running, based on the last iteration of CO. If the difference between the sum of priority combination set and the super-state chunk is within a threshold δ, then the set is prioritized for disaggregation and the priority combination is retained. Second, for each super-state chunk, we only generate and compare combinations of models that have similar mean value with the super-state chunk.

With these techniques, the computational complexity for T time becomes $O(T|S_{cm}|^{|N_{cp}|})$ where $|S_{cm}|$ is the number of mean-constrained appliances states and $|N_{cp}|$ is the number of priority-constrained appliances in the household.

Note that $S_{cm} \subseteq S$ and therefore $|S_{cm}| \leq |S|$. Further, note that $N_{cp} \subseteq N$ and therefore $|N_{cp}| \leq |N|$. In practice, $|S_{cm}| < |S|$ almost always applies and $|N_{cp}| << |N|$. These modifications to the original CO algorithm allows CO-Disaggregation() to have a low computational complexity and run in real-time. These techniques are implemented in function GenerateTrace().

Finally, we find a combination of appliance models which has the least root mean squared error (RMSE) compared to the super-state chunk. Eq. 5.1 with the least RMSE is implemented in function LeastRMSE(). The full CO-LeastRMSE is described in Algorithm 3.

Algorithm 3 Least RMSE combinatorial optimization

```
1: function CO-LeastRMSE($ss$,$model$,$priority\_combination$)
2:     $model\_traces \leftarrow$ GenerateTrace($ss, model, priority\_combination$)
3:     $min\_rmse \leftarrow$ LeastRMSE($ss, model\_traces$)
4:     $trace\_id \leftarrow$ where $min\_rmse$ in $model\_traces$
5:     $disagg\_ss \leftarrow model\_traces[trace\_id]$
6:     return $disagg\_ss$
7: end function
```

5.2 Model Tuner: Fine-tuning general appliance models at test house level

In a test house, it is very likely that there are appliances with different models and brands than those used in the training phase. For example, the model builder uses fridge from brand A model 1, brand B model 2 and 3, and brand C model 4. In the test house, there is a fridge from brand E model 1. Therefore it is important to tune the general appliance models according to specific appliances in the test house without any user intervention. In this section, we describe how we fine-tune the general appliance models based on the aggregated smart meter data of a test house.

The model tuner is only executed once during the initialization phase, without users intervention, at a test house. The main idea is to assign convincing chunks of aggregated smart meter data (i.e., super-state chunk) to each appliance. The model tuner algorithm is shown in Algorithm 4. This has three steps: (i) provides initial appliance-level information in the test house using CO-Disaggregation() and general appliance models; (ii) fit convincing chunk: for each super-state chunk, the algorithm finds the most suitable model and updates the model parameters based on the test house data; and (iii) cluster parameters: since there might be multiple models and parameters for each state, clustering is employed to find the most suitable model. We now describe these steps in detail.

Algorithm 4 Model tuner

```
1: procedure ModelTuner
2:     $disagg \leftarrow$ CO-Disaggregation($smart\_meter$,  $general\_model$)
3:     $fit\_params \leftarrow$ FitConvincing($disagg$)
4:     $tuned\_models \leftarrow$ EMClustering($fit\_params$)
5: end procedure
```

5.2.1 Fitting convincing chunk

The goal of this step is to find the appliance state model that matches the super-state chunk. A super-state chunk is considered convincing if, (i) the chunk only has one appliance trace dominating in the disaggregated data chunk and (ii) the *on_time* of the dominant appliance trace fits the typical usage duration of that type of appliance (i.e. appliance is active).

Let $\hat{y}_t^{(n)}$ be the estimated energy consumed of each appliance n at time t using CO-Disaggregation() and general appliance models. $\overline{y}_t$ represent the aggregate energy reading of the household. The criteria of a convincing chunk for appliance n can now be defined as,

$$accept_n(\overline{y}_t) = \begin{cases} true, & \text{if } \hat{y}_t^{(n)} > 0.5 \times \overline{y}_t \text{ and } length(t) \approx on\_time(n). \\ false, & \text{otherwise.} \end{cases} \tag{5.2}$$

where N is the set of all appliances in the household and t is the current super-state chunk. If $accept_n$ at the super-state chunk t is *true*, then the super-state chunk $\overline{y}_t$ is a convincing chunk for appliance n. The convincing chunks will be used to tune the general appliance models.

We determine the typical *on_time* for each type of appliance based on a previous research about appliance modeling [20]. We also measure the average usage duration of each type of appliance using several datasets to confirm the validity of that research.

These criteria are used based on common understanding and our observation that same appliances in different houses are often similar in terms of power consumption and usage duration, which is also confirmed in the literature [20]. Therefore, it makes sense to assume that a chunk of the aggregated smart meter data belongs to an appliance if the power consumption and usage duration of that general appliance model is similar to that chunk of smart meter data. Thus it makes more sense, rather than assuming that the chunk belongs to a combination of other appliances.

We then find a model and parameter by applying different curves and distributions fitting on the convincing chunk as described in *model fitting* in Section. 4.6. The details of the fitting convincing chunk module is explained in Algorithm 5.

Algorithm 5 Find convincing chunk and fit it

```
1: function FitConvincing($smart\_meter, disagg$)
2:     $min \leftarrow 0.1$
3:     $max \leftarrow 2.0$
4:     Loop $i$ in $appliances$                    ▷ i is appliance idx
5:         Loop $j$ in $disagg[i]$                 ▷ j is active chunk idx
6:             $ratio \leftarrow disagg[i][j]/smart\_meter[j]$
7:             if $ratio > 0.5$ then
8:                 $dominant \leftarrow True$
9:             end if
10:            $len \leftarrow length(disagg[i][j])$
11:            $min\_ot \leftarrow min * on\_time[i]$
12:            $max\_ot \leftarrow max * on\_time[i]$
13:            if $min\_ot < len < max\_ot$ then
14:                $fitrange \leftarrow True$
15:            end if
16:            if $dominant$ & $fitrange$ then
17:                $mp[i] \leftarrow$ ModelFitting($disagg[i][j]$)
18:            end if
19:        End Loop
20:    End Loop
21:    return $mp$    ▷ mp is the collection of models and parameters from convincing chunks
22: end function
```

5.2.2 Clustering tuned parameters

The *fitting convincing chunk* module generates a lot of mathematical models and parameters because there are a lot of convincing chunks for each appliance. To generate a tuned model for each appliance, a clustering algorithm is employed to group similar parameters for each mathematical model. This also removes outliers from the tuned appliance models. This is similar to the *clustering general parameters* step which is explained in Section. 4.7. However, in this step, we employ Expectation Maximization (EM) clustering algorithm to ensure each model parameter is given the same leve of importance during clustering. We use the GaussianMixture module in scikit-learn package to implement the EMClustering() function [24]. Finally, for each super-state detected we identify the corresponding fine-tuned model and its parameters.

This fine-tuning is performed on some historical data (say one week) in each test house. This is to ensure that all appliances are used at least once and their power consumption is reflected on the aggregated smart meter data. Additionally, model tuner can be executed again if new appliance is added later in the house or regularly e.g every 3 months to update the tuned appliances models. The model tuner algorithm is shown in 4.

5.3 Summary of energy disaggregation

UniversalNILM energy disaggregation module aims to disaggregate smart meter data in a test house using only general appliance models. First, it disaggregates smart meter data using the CO-disaggregation algorithm and general appliance models as we explained in Section 5.1. This provides initial appliance-level information for the test house. Second, the model tuner utilizes this information to tune the general appliance models using as we explain in Section 5.2. Finally, the tuned appliance models are used with the CO-disaggregation algorithm to predict fine-grained appliance-level data in the test houses. Note that, other disaggregation modules such as HMMs [14] can be used with the tuned appliance models.

Chapter 6

Evaluation and Results

This chapter starts with the explanation about the experiment: datasets, metrics, and implementation details in Section 6.1. In Section 6.2 we present and discuss the experiment results for both appliance load modeling and energy disaggregation. In Section 6.3 we measure and discuss the execution time of UniversalNILM to see its real-time capability. We perform a stress test on UniversalNILM in Section 6.4 by increasing the number of appliances. Finally, we investigate the effect of different amounts of data on the disaggregation accuracy in Section 6.5.

6.1 Experimental evaluation

In this section, we present details about the datasets used, metrics employed and implementation details.

6.1.1 Datasets

We use three publicly available and well-known energy datasets:
REDD [16]. The Reference Energy Disaggregation Data Set was released by MIT in 2011. The dataset contains several weeks of aggregated smart meter data and appliance-level power data for *six* different houses in the United States. In our evaluation, we use only 4 households as the remaining two households have data for less than two weeks.
UK-DALE [13]. This dataset records the power demand from *five* houses in UK. In each house, both the aggregated total mains power consumption as well as power consumed by individual appliances are recorded.
DRED [27]. The Dutch Residential Energy Dataset includes one house aggregated smart meter data along with the appliance-level power data for more than five months.

We choose the above datasets mainly due to the following reasons, (i) all of the above datasets record both appliance and smart meter data every second; (ii) moreover, all the above datasets share at least 5 appliances so that we can perform blind disaggregation across the households and the datasets; and (iii) we use three datasets to ensure the proposed solution can be used anywhere and can be used to compare the results with state-of-the-art algorithms (scalability and applicability).

6.1.2 Metrics

We employ three standard metrics to evaluate UniversalNILM across the three datasets.

Root mean square error. This is a common measure to calculate the differences between ground truth values and values generated by models.

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(p_t - \hat{p_t})^2}, \tag{6.1}$$

where p_t is the appliance-level data at time t, $\hat{p_t}$ is the predicted appliance-level data at time t, and T is the length of appliance-level data.

F1-score. F1-score is used to measure the accuracy of an algorithm.

$$F1\text{-}score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{6.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{6.4}$$

True positives (TP) indicates that an appliance is ON in ground truth and the disaggregation result also indicates appliance iw ON. FP is the false positives, i.e., an appliance is OFF but the disaggregation reports that the appliance is ON. FN is the false negatives, i.e., an appliance is ON but the disaggregation reports that the appliance is OFF.

Proportion of total energy correctly assigned (PTE). Apart from classifying whether an appliance is on or off, one might be interested in observing the amount of actual energy being correctly predicted by an energy disaggregation algorithm. PTE is a common metric [12, 16] employed to predict the amount of power consumed by each appliance.

$$PTE = 1 - \frac{\sum_t^T \sum_i^N \left| p_t^{(i)} - \hat{p_t}^{(i)} \right|}{\sum_t^T \bar{p}_t}, \tag{6.5}$$

where T is the length of appliance/smart meter power data and N is the total number of appliances.

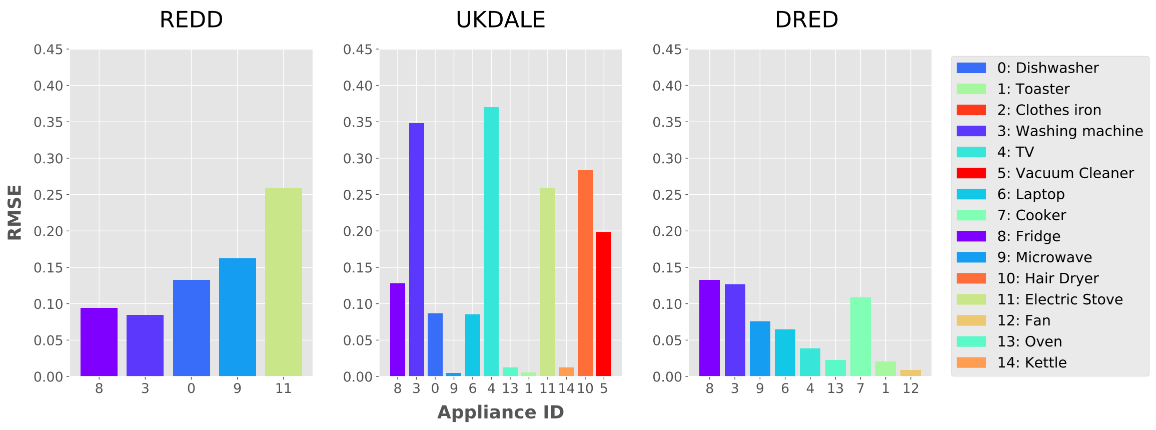


Figure 6.1: Evaluation for general appliance model builder across different appliances and datasets.

6.1.3 Implementation details

The complete UniversalNILM framework is implemented in Python. Specifically, Scikit-Learn and PyEEG [4] libraries are used for state detection and super-state detection as mentioned in Section. 4 and 5. Furthermore, we have integrated our framework with NILMTK – Non-intrusive load monitoring toolkit [7] – used by energy disaggregation researchers. NILMTK currently supports multiple energy datasets and acts as a common platform to evaluate various energy disaggregation algorithms.

6.2 Results

In this section, we present detailed evaluation results of UniversalNILM. We show how general appliance models learnt on a few training houses are transferred to disaggregate the aggregated smart meter data of houses in both the same and different datasets (different geographic locations). Specially, we discuss, (i) how accurate is the general appliance models that are constructed from a few training houses, (ii) disaggregation performance of tuned appliance models on unseen test houses in the same dataset, and (iii) disaggregation performance of tuned appliance models on unseen test houses in a different dataset. All experiments use real household data with 1-Hz sampling frequency. The disaggregator provides fine-grained appliance-level data at 1-Hz frequency.

Table 6.1: General appliance model evaluation.

Dataset	Train houses	Test houses
REDD	1, 2	3, 5, 6
UK-DALE	1, 2	3, 4, 5
DRED	1	1

6.2.1 General appliance models builder

General appliance model builder takes a few training houses in a dataset and builds a general model for each appliance that can be applied across other households. In our evaluation, we used *two* households for training and *threee* households for testing in REDD dataset. Similarly, in UK-DALE *two* houses and *three* households were used for training and testing, respectively. In DRED dataset, since it has only *one* household, we use one week data from training and another week data for testing. Table. 6.1 shows the training and testing houses employed and in total 16 appliances were modeled from all of these houses.

To test the accuracy of the general appliance models, we use root mean squared error (RMSE) between the ground truth appliance-level data and the predicted appliance-level data generated from our general appliance models.

Figure 6.1 shows the average RMSE values for each appliance across all the test houses in the datasets. A low RMSE value indicates that the general appliance models were able to predict the appliance data in the test house. It can be seen that the RMSE values for most appliances are low ($< 30\%$) in all datasets. This shows that the general appliance models are able to capture broad features that can be used to model appliances of various brands and models. Furthermore in DRED, the RMSE values are very low. Since DRED has only one house, the general appliance models are able to capture the appliances features accurately. This shows that the proposed model can be used to generate both brand-specific and general model of an appliance. Finally, the high RMSE values in UK-DALE for the television and the washing machine is mainly due to sparse appliance model parameters resulting in clustering inaccuracies.

6.2.2 Energy disaggregation on test houses in the same dataset

As seen previously, general appliance models are not always accurate. Hence, we fine-tune the general models to obtain tuned appliance models. Fine tuning is executed only once during an initialization phase as described in

Table 6.2: Disaggregation evaluation scenario

Dataset	Train houses	Test houses
REDD	1, 2	3, 5
UK-DALE	1, 2	4, 5
DRED	1	1

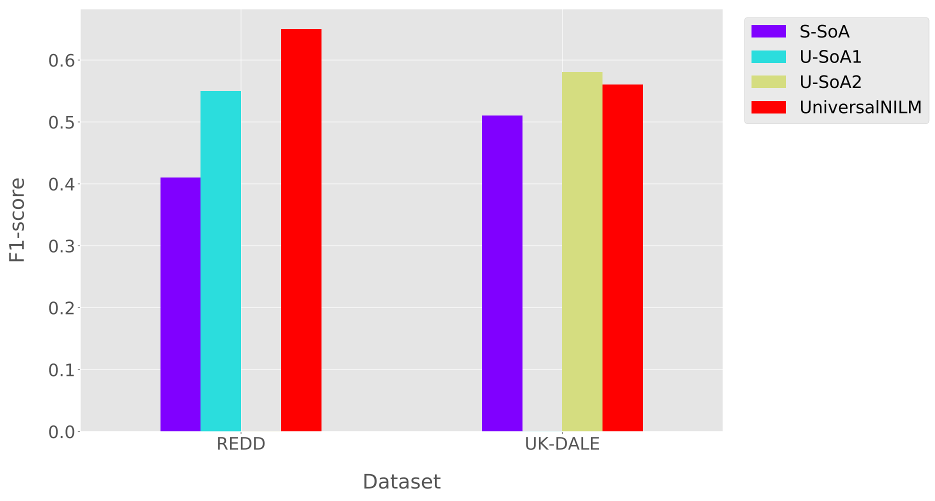


Figure 6.2: F1-score comparison

Section 5.2. Table 6.2 shows the traning and testing houses used to evaluate the tuned models. The training phase from which the general appliance models are derived uses one week of data. For the testing phase, one week of aggregated smart meter data is extracted from the test houses.

F1-score: Dataset level. Since the number and type of appliances in each household and dataset may vary (i.e. a type of appliance is available in a house, but it is not available in the other houses), we show the results for the top-5 appliances across the datasets. F1-score is used to measure the accuracy of disaggregation using tuned appliance models. We compare the disaggregation accuracy of our proposed tuned appliance models with the state-of-the-art energy disaggregation solutions as shown in Table. 6.3 and Figure 6.2. We compare UniversalNILM with (i) a supervised technique (S-SoA) used as baseline based on FHMM models in NILMTK, (ii) an unsupervised technique (U-SoA1) based on approximate inference in FHMM [15], which represents an unlabeled unsupervised disaggregation method, and (iii) an unsupervised technique (U-SoA2) based on NeuralNILM [12], which represents a labeled unsupervised algorithm.

It can be seen that UniversalNILM outperforms the current state-of-the-

Table 6.3: F1-score comparison

Disaggregator	REDD	UK-DALE
S-SoA	0.41	0.51
U-SoA1	0.55	-
U-SoA2	-	**0.58**
UniversalNILM	**0.65**	0.56

art baseline supervised and unsupervised NILM techniques. The results are presented in the form of the average F1 score of all test houses in both the REDD and UK-DALE datasets. Note that, we ensured all the algorithms are evaluated on the same test data. Furthermore, UniversalNILM has much lower computational complexity compared to the other algorithms and runs in real-time across the datasets.
F1-score: Appliance level. We now discuss the F1-score per appliance across datasets. Table. 6.4 shows the F1-score for the top-5 appliances in each dataset, where (n/a) indicates that the appliance is not present in the house and (-) indicates that the appliance is never ON in the test data.

Table 6.4: F1-score results

Appliance	REDD		UK-DALE		DRED
	3	5	1	4	1
Dishwasher	0.78	0.42	0.27	n/a	n/a
Fridge	0.82	0.63	0.85	0.75	0.84
Microwave	0.65	-	-	-	0.43
Washing machine	0.76	-	0.47	0.53	-
Kettle	n/a	n/a	0.23	0.67	n/a
Television	n/a	n/a	-	-	-
Cooker	n/a	n/a	n/a	n/a	0.72
Laptop	n/a	n/a	n/a	n/a	0.22

In general, it can be seen that the proposed tuned models have high F1-score for simple appliances such as fridge, microwave, cooker, and kettle (average F1-score is $> 50\%$). Furthermore, the tuned models have high accuracy ($> 50\%$) for some complex appliances such as dishwasher and washing machine. However, there are a few cases where F1-scores for complex appliances are $< 40\%$. This is because, (i) there are other appliances with similar mean value as the complex appliances (inherent problem with CO-based disaggregation) and (ii) the composite appliances are active only once during

Table 6.5: Proportion of total energy correctly assigned against various approaches for REDD dataset.

Disaggregator	REDD
S-SoA	**0.93**
U-SoA1	0.48
UniversalNILM	0.77

tuning period and therefore the chance to tune their models accurately is very limited.

Proportion of total energy correctly assigned. As mentioned earlier, proportion of total energy correctly assigned is a key metric for energy disaggregation algorithms. We use the same training and testing scenario as described earlier in Table. 6.2.

Table. 6.5 shows the PTE metric for the test houses in REDD dataset. We compared UniversalNILM against S-SoA based on [7] and U-SoA1 based on [16]. It can be seen that, S-SoA has the highest PTE, understandably since the models of appliances are based on actual appliance power data in the test houses. However, UniversalNILM outperforms the state-of-the-art unsupervised technique. UniversalNILM has an average PTE of 77% as compared to 48% of U-SoA1. Furthermore, 77% PTE is considered to be an adequate result, since UniversalNILM does not need any training data at the test house.

Figure 6.3 illustrates the energy assigned to each appliance in ground truth and UniversalNILM for a day in a house in the REDD dataset. It can be seen that, the energy assigned to each appliance closely resembles the ground truth data.

Disaggregation using tuned models. Figure 6.4 illustrates the proposed disaggregation based on tuned models on House-3 in REDD and corresponding ground truth. It can be seen that the disaggregation result of UniversalNILM for each type of appliance closely resembles to the ground truth.

6.2.3 Energy disaggregation on test houses in a different dataset

In the previous sections, we described how UniversalNILM outperforms the state-of-the-art techniques when applied on the households in the same dataset. In this section, we present results where UniversalNILM is tested on a dataset and trained on other datasets. The key idea is that the general appliance models from training houses should be able to be tuned to unseen

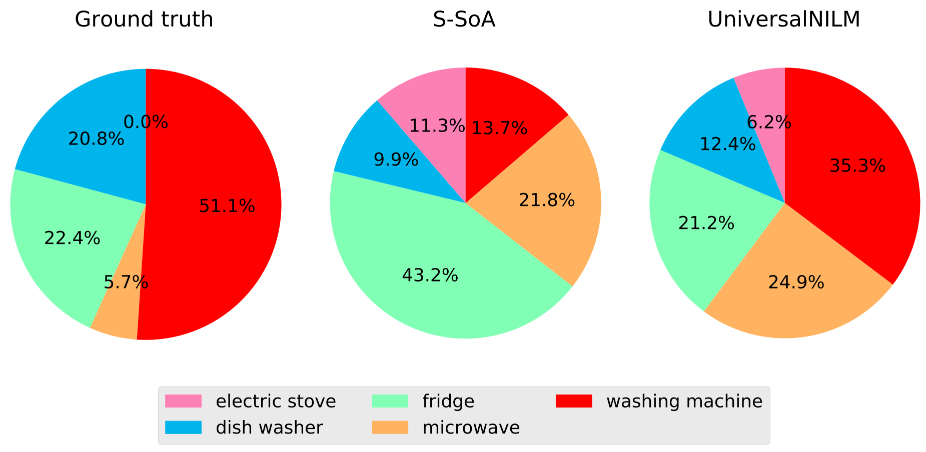


Figure 6.3: Proportion of appliance energy in ground truth and tuned model CO output

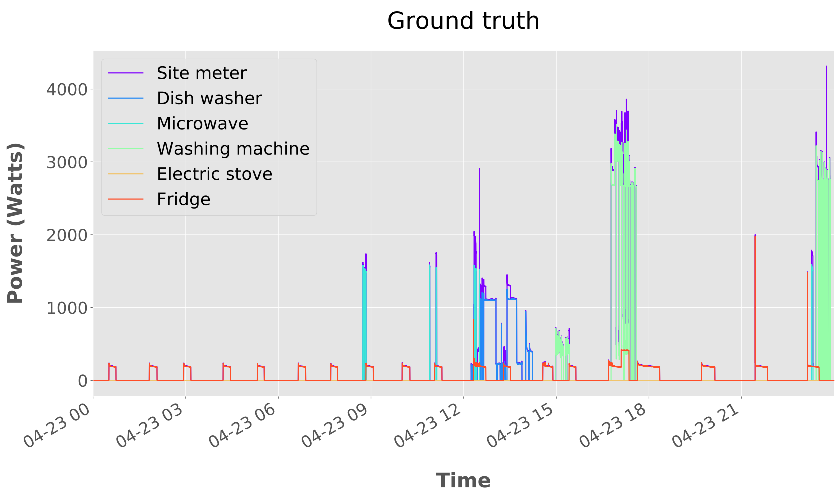
households from different datasets and locations. For example, even if we train UniversalNILM in some houses in Netherlands, it still works in houses in UK or Germany which demonstrates the scalability of UniversalNILM.

Table. 6.6 shows the average F1 score of UniversalNILM and the state-of-the-art approaches on the three datasets. Note that while UniversalNILM uses different datasets for training and testing, both U-SoA1 and U-SoA2 use the same dataset for training and testing (different houses in the same dataset).

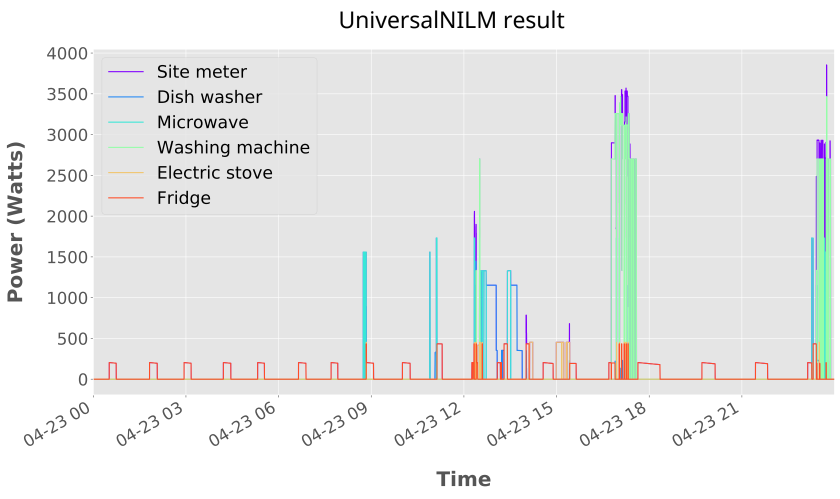
Table 6.6: Disaggregation on test houses in a different dataset.

	Scenario-1	Scenario-2	Scenario-3
UniversalNILM	**0.57**	0.55	0.53
U-SoA1	0.55	n/a	n/a
U-SoA2	n/a	**0.58**	n/a

Scenario 1: Training on UK-DALE and DRED, Test on REDD. In this scenario, we employ house 1 and house 2 data from UK-DALE and house 1 data from DRED to build general appliance models. These general appliance models are then fine-tuned into tuned appliance models on the test houses, i.e., REDD house 1, 2, and 3. It can be seen that Universal-NILM outperforms the state-of-the-art unsupervised technique (U-SoA1). UniversalNILM achieves F1-score of 57% as compared to 55% of U-SoA1



(a) Ground truth



(b) UniversalNILM tuned model output

Figure 6.4: Disaggregation with tuned model CO example in a house in REDD dataset

(see Table. 6.6).

Scenario 2: Training on REDD and DRED, Test on UK-DALE. In this scenario, we employ house 1 data from REDD and house 1 data from DRED to build general appliance models. These general appliance models are then fine-tuned into tuned appliance models on the test houses, i.e., UK-DALE house 1, 2, 4, and 5. In this scenario, UniversalNILM has a lower accuracy to the unsupervised state-of-the-art algorithm (U-SoA2). UniversalNILM achieves F1-score of 55% as compared to 58% of U-SoA1 (see Table. 6.6). The lower F1-score is due to the very low usage of a certain appliance in the training data (the toasters are rarely active in the training houses). We will discuss more about this in Section 6.5.

Scenario 3: Training on REDD and UK-DALE, Test on DRED. In this scenario, we employ data from REDD and UK-DALE to build general appliance models. These general appliance models are then fine-tuned into tuned models on the test houses, i.e., DRED house 1. There is no previous attempt of unsupervised disaggregation on DRED house 1. UniversalNILM achieves a consistent F1-score in this scenario as in previous scenarios (around 50%). This result strengthens the argument that UniversalNILM can be applied to any household in any neighborhood/city.

Thus, the proposed UniversalNILM framework can perform blind disaggregation without training data, i.e., disaggregation on households from same dataset and disaggregation on households from different dataset. In both these scenarios, UniversalNILM outperforms state-of-the-art supervised and unsupervised techniques as shown in Table 6.7. In this report we extensively evaluated our hypothesis on real-world datasets, where general appliance models learnt on a few training households can be successfully transferred to other households from same dataset and also from a different dataset without any manual efforts.

Table 6.7: Detailed comparison of various results. Same dataset means that the training and test houses are different, but within the same dataset. Different dataset means that the training and test houses are different, and they come from different datasets.

	Same dataset		Different dataset		
	REDD	UK-DALE	Scenario-1	Scenario-2	Scenario-3
UniversalNILM	**0.65**	0.56	**0.57**	0.55	0.53
S-SoA	0.41	0.51	n/a	n/a	n/a
U-SoA1	0.55	n/a	0.55*	n/a	n/a
U-SoA2	n/a	**0.58**	n/a	**0.58***	n/a
Changes	+38%	+6.5%	+4%	-5.5%	n/a

Note that U-SoA1 and U-SoA2 results (*) use same dataset for training and testing, only different houses as described in Subsection 6.2.3

6.3 Real-time capability

Another goal of this thesis is to develop a universal energy disaggregation framework that can be run on an embedded systems. This is important so that disaggregation can be performed locally in a household to minimize privacy concerns. There is no specific and hard constraint about a minimum disaggregation output interval. However, a study on real-time effectiveness of feedback [10] suggests that 10 minutes output interval is enough to give an effective feedback.

We measure the computational time using real-household data with 1 Hz sampling frequency. The disaggregator outputs fine-grained appliance power data at 1 Hz frequency. The computational time of our Python-processing algorithm on an i7, @2.30 GHz machine is shown in Table 6.8.

Table 6.8: Average computational time of UniversalNILM disaggregator component for 7 days of data in all datasets (in minutes)

	REDD	UK-DALE	DRED
Model tuner	17	19	16
CO disaggregation	7.3	7	5.6

We only measure modules from the disaggregator because it is the only component that runs in the test households. The computational time of the model tuner is more than twice of the CO disaggregation. This is acceptable since the model tuner runs only once during the initialization phase. The average computational time of the CO disaggregation is 6.6 minutes for 7 days of real household data. Although this experiment is not done on an embedded board, this result suggests that a real-time implementation is feasible.

6.4 Stress test

Most unsupervised energy disaggregation algorithms disaggregate only a limited number of appliances (mostly 5 appliances) [6,12,14,19]. As the number of disaggregated appliances increases, the difficulty increases. Therefore, it is interesting to see how UniversalNILM reacts to increase in the number of appliances. In this experiment, we perform disaggregation on test houses in a different dataset with 5, 6, 7, and 8 appliances. We only test using

UK-DALE dataset because it has enough number of similar appliances with the other datasets. REDD and DRED have a limited number of similar appliances across datasets, as mentioned in 6.2.2.

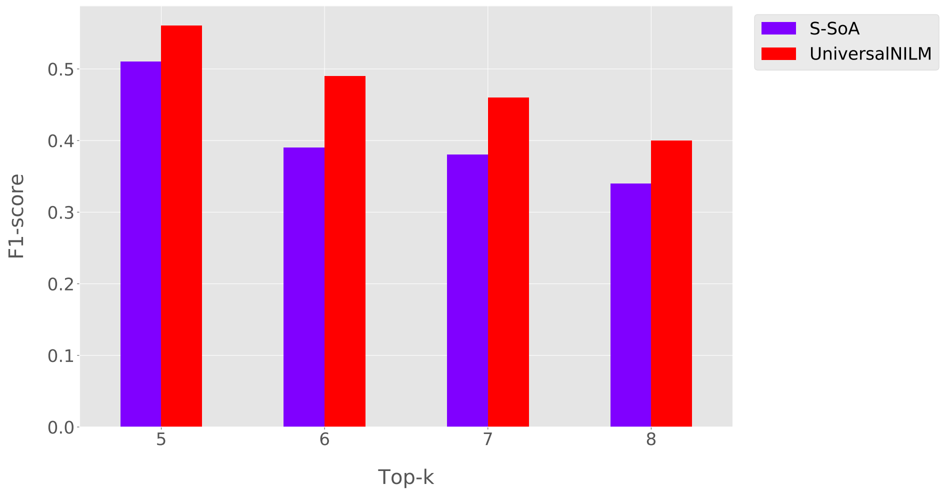


Figure 6.5: F1-score comparison with increasing number of appliances. Train on REDD and DRED, test on UK-DALE.

We employ all of the houses data from REDD and DRED to build general appliance models. These general appliance models are then automatically fine-tuned using UniversalNILM algorithm on the test houses, i.e., UK-DALE house 1, 2, 4, and 5. The maximum number of appliances is 8 because (i) the number of similar appliances with the other datasets is limited; and (ii) the types of appliances in households generally amount to such. It can be seen that UniversalNILM outperforms the state-of-the-art supervised technique (S-SoA). UniversalNILM accuracy is consistently higher than S-SoA for all number of appliances (see Figure 6.5).

However, this result also shows that the dissaggregation accuracy of UniversalNILM drops as the number of appliances increases. This is also the case with most energy disaggregation algorithms (for example, see S-SoA result in Figure 6.5) which is why most energy disaggregation algorithms are only evaluated on limited number of appliances, especially the unsupervised ones. As the number of appliances increases, there is a higher probability that there are appliances with similar signatures which are difficult to differentiate from each other.

6.5 Effect of amount of data on disaggregation accuracy

In the previous sections, we measured the accuracy of UniversalNILM using a certain amount of data. For example, we employed house 1 data from REDD and house 1 from DRED to train the general appliance models in scenario 2 in Subsection 6.2.3. We could have used a different amount of training data. For example, we could have used three houses from REDD and one house from DRED to build the general appliance models. Although the overall results suggest that the disaggregation accuracy of UniversalNILM is consistently higher that the related works, we also come up with a question: how to find the optimum amount of data to generate good general appliance models?

In this section, we investigate the effect of amount of data on disaggregation accuracy. We vary the amount of data by using different amounts of houses for training. As before, we use the top-5 appliances since most houses have those appliances. This investigation covers all of the three datasets using the following scenarios.

Scenario 1: Training on UK-DALE and DRED, Test on REDD. For training, we use one, two, three, until finally all four houses in UK-DALE. Since DRED only has 1 house, we always use that house for training. For testing, we use house 1, 2, and 3 in REDD.

Scenario 2: Training on DRED and REDD, Test on UK-DALE. For training, we use one, two, three, four, until finally five houses in REDD. Since DRED only has 1 house, we always use that house for training. For testing, we use house 1, 2, 4, and 5 in UK-DALE.

Table 6.9: Training houses in UK-DALE and REDD that are used to investigate the effect of the amount of data on disaggregation accuracy in DRED (scenario 3).

	Amount of training houses					
	2	3	4	5	6	7
UK-DALE houses	1	1, 2	1, 2	1, 2, 4	1, 2, 4	1, 2, 4, 5
REDD houses	1	1	1, 2	1, 2	1, 2, 3	1, 2, 3

Scenario 3: Training on UK-DALE and REDD, Test on DRED. For training, we use six different amounts of training houses. For simplicity, this scenario is described in detail in Table 6.9. Each cell contains the id of the training house(s). For testing, we use house 1 in DRED.

The results are shown in Figure 6.6. As a reference, let us assume that 0.5

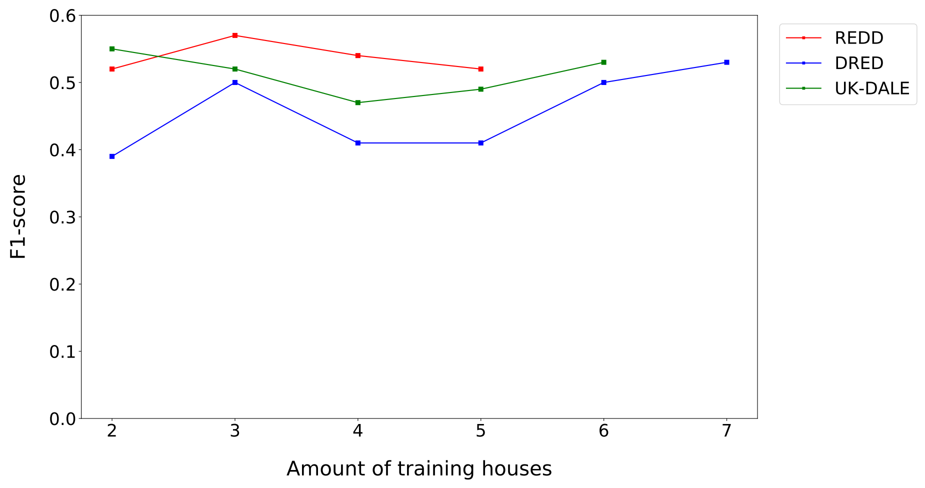


Figure 6.6: The disaggregation accuracy with different amount of training data, for each dataset

is an adequate F1-score since other unsupervised NILM algorithms usually have this level of accuracy. From this figure, we can see that the accuracy usually increases to the adequate level when we used six training houses or more. Although we cannot prove that this is the case for REDD, it is safe to assume that using six or more training houses will still give an adequate F1-score for REDD. This is because the F1-scores of REDD are always higher that 0.5.

The interesting thing is that the trend is not always positive as the amount of data increases. In all scenarios, when the amount of training houses are four or five, the accuracy drops. A possible explanation for this phenomena is that increasing the number of training data may confuse the model tuner in Section 5.2. In other words, since there are more appliance models, the model tuner becomes more vulnerable to the weakness of the combinatorial optimization algorithm (i.e., the CO algorithm picks a wrong set of appliances as in the tuning phase). This problem seems to be solved by adding even more training data to a certain threshold as we mention in the previous paragraph.

In the end, we find six houses as the minimum amount of training data to obtain an adequate disaggregation accuracy. However, we are yet to find the optimum amount (i.e., an amount of training houses, when the disaggregation accuracy stops increasing from that point on). Looking at Figure 6.6, we need more training data and houses in order to find the optimum amount. Therefore finding an optimum amount of training data is still an open problem to solve.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The key contribution of our work is the proposition of a novel semi-supervised energy disaggregation framework based on general appliance models. The proposed framework – UniversalNILM – can perform unsupervised/blind disaggregation on test houses from the same dataset as training, as well as test houses from different datasets. At the core is the idea of developing a comprehensive general appliance models that can be tuned automatically at the test house for accurate energy disaggregation.

We developed a method to build the comprehensive general appliance models automatically. Using this, we developed a method to tune the general appliance models at the test house automatically. We employed a modified CO algorithm to predict appliance-level energy consumption information based on models of appliances. Finally, UniversalNILM was extensively evaluated across three publicly available real-world datasets *viz.,* REDD, UK-DALE and DRED.

In terms of appliance load modeling, UniversalNILM general appliance models have a low RMSE values compared to ground truth appliance data ($< 10\%$) for all datasets. This shows that the general appliance models can be used to generate both brand-specific model and general model of appliances.

UniversalNILM outperforms the state-of-the-art supervised and unsupervised techniques in terms of energy disaggregation accuracy. This answers the first part of the research goal, which is to develop an energy disaggregation framework that can be applied to any household in any neighborhood/-city.

We discussed about UniversalNILM disaggregation computational complexity in Sec. 5.1.2 and concluded that the modified CO algorithm has a lower computational complexity than the original CO algorithm. Furthermore, we measured the computational time of UniversalNILM disaggregation. The average computational time is within the suggested constraint for an effective feedback on energy consumption, which suggests that a real-time implementation is feasible. This answers the second part of the research goal, which is to develop an energy disaggregation framework that can run on an embedded system in real-time locally at a household.

Additionally, we also try to push the limits of our unsupervised energy disaggregation algorithm by evaluating UniversalNILM against more appliances. Although the accuracy decreases as the number of appliances increases, UniversalNILM consistently outperforms the state-of-the-art supervised technique in every cases.

The F1-score of UniversalNILM in the REDD and UK-DALE datasets are 65% and 56%, respectively. These numbers show improvements of 38% and 6.5% compared to the related works and state-of-the-art NILM algorithms. Furthermore, UniversalNILM achieves comparable (+/- 4.5% difference) F1-scores with the state-of-the-art unsupervised NILM algorithms even if UniversalNILM uses a whole different datasets for training and testing (other algorithms use same datasets, only different houses). UniversalNILM also consistently achieves higher F1-scores (on average, 7.3% higher) compared to NILMTK FHMM, the reference supervised energy disaggregation algorithm, during the stress test. In the stress test, we used more and more appliances: from five, six, seven, to eight appliances in the test houses.

Finally, we measure the execution time of UniversalNILM to investigate whether if it is possible to run the disaggregation algorithm in real-time, locally in a test house. The average computational time of UniversalNILM disaggregator is 6.6 minutes for seven days of aggregated smart meter data. Although the test is done not on an embedded board, spending several minute to disaggregate one whole week data suggests that a real-time implementation on an embedded board is possible.

Our proposed framework can now enable wide-adoption of NILM techniques due to the reduced burden of collecting appliance-level data at each household. To the best of our knowledge, this is the first effort towards developing an unsupervised NILM technique that can be applied across datasets without training.

7.2 Future Work

While being scalable across datasets and accurate compared with other NILM algorithms, UniversalNILM faces several challenges before it can really be applied in the real-world. Much of it comes from the limitation of currently available energy datasets. There are only 4 datasets, out of 20+ energy datasets, that contain appliance-level data each consists of 1-4 houses[1]. Building general appliance models is therefore difficult since there

[1]http://wiki.nilm.eu/datasets.html

are so many appliances from different brands and models.

UniversalNILM uses CO algorithm to perform model tuning and disaggregation. In general, CO suffers if there are appliances with similar power consumption. We investigated one example of this problem in Section 6.5. Instead of relying mainly on power consumption, we can also make use of feature-rich appliance signatures (state transition, duration in each states) to tune general models from aggregated smart meter data. While developing a feature extractor for such rich signatures might be difficult, it is possible to use other methods like deep learning. However, this approach will likely require significantly bigger datasets containing thousands of appliance-level data for different appliances which we do not have at this moment.

Furthermore, the disaggregation accuracy obtained by UniversalNILM can be further improved by incorporating other disaggregation algorithm such as Hidden Markov Model. HMM relies on appliance state transition and emission probabilities, instead of relying on power consumption alone. This way, the disaggregation accuracy and number of disaggregated appliances may increase.

Bibliography

[1] CO_2 emissions from fuel combustion: overview (2017 edition). `http://www.iea.org/publications/freepublications/publication/CO2EmissionsFromFuelCombustion2017Overview.pdf`. [Online; accessed 26-October-2017].

[2] Plugwise circle. `https://www.plugwise.com/circle`. [Online; accessed 18-October-2017].

[3] Smart grids strategic research agenda towards 2035. `http://www.egvi.eu/uploads/Modules/Publications/smartgrids-sra2035.pdf`. [Online; accessed 18-October-2017].

[4] Forrest Sheng Bao, Xin Liu, and Christina Zhang. Pyeeg: an open source python module for eeg/meg feature extraction. *Computational intelligence and neuroscience*, 2011, 2011.

[5] Sean Barker, Sandeep Kalra, David Irwin, and Prashant Shenoy. Empirical characterization, modeling, and analysis of smart meter data. *IEEE Journal on Selected Areas in Communications*, 32(7):1312–1327, 2014.

[6] Sean Barker, Sandeep Kalra, David Irwin, and Prashant Shenoy. Powerplay: creating virtual power meters through online load tracking. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 60–69. ACM, 2014.

[7] Nipun Batra, Jack Kelly, Oliver Parson, Haimonti Dutta, William Knottenbelt, Alex Rogers, Amarjeet Singh, and Mani Srivastava. Nilmtk: an open source toolkit for non-intrusive load monitoring. In *Proceedings of the 5th international conference on Future energy systems*, pages 265–276. ACM, 2014.

[8] Sarah Darby et al. The effectiveness of feedback on energy consumption. 2006.

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

[10] Sébastien Houde, Annika Todd, Anant Sudarshan, June A Flora, and K Carrie Armel. Real-time feedback and electricity consumption: A field experiment assessing the potential for savings and persistence. *The Energy Journal*, 34(1):87, 2013.

[11] Srinivasan Iyengar, David Irwin, and Prashant Shenoy. Non-intrusive model derivation: automated modeling of residential electrical loads. *Power (Watt)*, 500(1000):1500, 2016.

[12] Jack Kelly and William Knottenbelt. Neural nilm: Deep neural networks applied to energy disaggregation. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 55–64. ACM, 2015.

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

[14] 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.

[15] 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, 2012.

[16] J Zico Kolter and Matthew J Johnson. Redd: A public data set for energy disaggregation research. 2011.

[17] Henning Lange and Mario Bergés. Bolt: Energy disaggregation by online binary matrix factorization of current waveforms. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, pages 11–20. ACM, 2016.

[18] 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.

[19] Masako Matsumoto, Yu Fujimoto, and Yasuhiro Hayashi. Energy disaggregation based on semi-binary nmf. In *Machine Learning and Data Mining in Pattern Recognition*, pages 401–414. Springer, 2016.

[20] Malcolm McCulloch. *Reducing domestic energy consumption through behaviour modification*. PhD thesis, University of Oxford, 2009.

[21] Office of the United Nations Environment Programme - Sustainable Buildings and Climate Initiative (SBCI). Buildings and climate change: A summary for decision-makers, 2009.

[22] Danny Parker and David Hoak. How much energy are we using? potential of residential energy demand feedback devices.

[23] Oliver Parson, Siddhartha Ghosh, Mark J Weal, and Alex Rogers. Non-intrusive load monitoring using prior models of general appliance types. 2012.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[25] Steven M Pincus, Igor M Gladstone, and Richard A Ehrenkranz. A regularity statistic for medical data analysis. *Journal of Clinical Monitoring and Computing*, 7(4):335–345, 1991.

[26] Joshua S Richman and J Randall Moorman. Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology*, 278(6):H2039–H2049, 2000.

[27] Akshay S.N. Uttama Nambi, Antonio Reyes Lua, and Venkatesha R. Prasad. Loced: Location-aware energy disaggregation framework. In *Proceedings of the 2Nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, BuildSys '15, pages 45–54, New York, NY, USA, 2015. ACM.

[28] Michael Zeifman. Disaggregation of home energy display data using probabilistic approach. *IEEE Transactions on Consumer Electronics*, 58(1), 2012.

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