university of groningen

faculty of science and engineering

Non-intrusive appliance load monitoring using breakpoint identification and external features

Bachelor thesis

Author:
Job Heersink

Supervisors:
Viktoriya Degeler
Alexander Lazovik

University of Groningen
The Netherlands
June 7, 2020

Abstract

Monitoring of home appliances provides useful information on how to improve user consumption habits and refine energy conservation. This information could also be used to reduce energy consumption overall. Many have tried to measure the power consumption per device using sub-meters at plug level, this however is impractical and economically infeasible. In this paper we explore the possibility of obtaining the state of devices from aggregate power consumption data using breakpoint identification with the help of neural networks. We will explore an existing non intrusive appliance load monitoring method and analyse how well it does on a standard data set as well as a data set containing the most frequently present devices in a household. We will extend this method to take external features like temperate, wind and weather data into account. In addition to that we will explore the opportunity of generalizing this implementation across houses by letting the neural network learn on a specific set of household consumption data and testing it on another data set.

Contents

Chapter 1

Introduction

Due to the increased electric energy demands over the last few years, more and more electric utilities are upgrading their traditional power grids to a more sophisticated, advanced and self-healing smart grid[1][2]. With the increasing presence of smart grids, it is now possible to manage, control and monitor the energy consumption on real time usage basis to enhance the efficiency or reduce the power consumption of a power system network.

Using Appliance load monitoring techniques, information on the power consumption of individual devices can be gathered on a real time basis, which is useful for reducing the overall power consumption. Many intrusive techniques have been developed and implemented[3], but intrusive methods measure the power consumption of appliances on socket level, thus requiring a substantial amount of hardware. This can be exceptionally costly and time consuming to implement on a large scale. Non intrusive appliance load monitoring (NIALM) techniques, using disaggregation techniques to gather the states of appliances from an aggregate energy signal, at most only require measuring equipment in the setup fase. This makes NIALM techniques more cost efficient and less time consuming than the alternative.

George Hart[4] was the first to publish an article on non intrusive appliance load monitoring. Hart made the distinction between manual setup non-intrusive appliance load monitoring (MS-NIALM) and automatic setup non-intrusive appliance load monitoring (AS-NIALM), where a MS-NIALM method would require a manual setup fase to make the implementation disaggregate the aggregate power consumption signal and an AS-NIALM method only needs to be "plugged-in" to the aggregate signal. Although AS-NIALM seems to have the most potential both cost wise and time wise, no existing implementation exists thus far[5]. Some research has been done on the subject and quite some neural network implementations seem to get some decent results[6][7], but seem to work on a only on a selection of large non-varying devices. A decent working implementation thus still needs to be found.

In this paper we will look at an existing MS-NIALM Neural network implementation which uses breakpoint identification to disaggregate their aggregate power consumption using low frequency data proposed by Gao et al.[8]. This implementa-

tion shows promising results for disaggregating both small and big devices, uses an input feature which can be extended to include improvement, utilizes low frequency data instead of high frequency data, which is easier to gather per household, and shows promise to be adapted as a AS-NIALM implantation, because this neural network trains on general features of the power signal. We will test this implementation on both the reference Energy Disaggregation Data Set (REDD)[9], to create a benchmark and proof that our implementation is at least partly similar to Gao's, and our own gathered data using the Plugwise measuring devices from v. degeler[3] to gather our own power consumption data of 10 devices from a 1 person studio apartment, to test the implementation on a larger quantity of modern household devices. In addition to that we would like to improve the implementation by incorporating weather data in the input features and test on the two data sets, as Gao mentions in the future work section: "Another direction would be to incorporate more features to improve the detection accuracy, such as temperature, humidity, and water usage". Lastly, the possibility of extending this method to an AS-NIALM implementation will be researched by training the neural network on REDD and testing on our own gathered studio data.

The structure of this article is organized as follows. Chapter 2 present the research quesntions and details about the implemented NIALM method. Chapter 3 presents prior work, external datasets and libraries related to appliance disaggregation. Chapter 4 describes how our own power consumption data was gathered and what devices were used. In Chapter 5 we present the experiments on the neural network and in Chapter 6 and 7 we conclude the article.

Chapter 2

PROPOSAL

2.1 Problem Formulation

The problem of disaggregating an aggregate energy consumption signal can be formulated as follows:

$$y(t) = \sum_{n=1}^{N} y_n(t)$$

where $y(t)$ stands for the total power used at a certain point in time, t. $y_n(t)$ stands for the power consumption of appliance n and N for the amount of appliances. As we will show in the experiment of Chapter 4.3 $y(t)$ can be represented as the sum of the power consumption of all devices at a certain point in time.

The problem here is that we would like to retrieve the value of $y_n(t)$ knowing only $y(t)$. So we would like to get an approximation $M(y(t))$

$$M(y(t)) = [y_1(t), y_2(t), ..., y_N(t)]$$

Where M returns N distinct values that represents the consumption of the different appliances. We are however only interested in the activation of the appliance and not the entire consumption. We thus need to define the appliance activation $x_n(t)$ as follows, where the value 1 represents the on state and 0 the off state:

$$x_n(t) = \begin{cases} 0 & y_n(t) \leq c_n \\ 1 & y_n(t) > c_n \end{cases}$$

Where c_n is stands for the power consumption at the off state. For many appliances this value will be 0, however appliances with a cubed power supply connected to it, like the living room lamp from our data set, still consume power while turned off. These cases are also referred to as vampire appliances [10].

Using this definition we can define an operator M_x which approximates the state of each appliance. The goal of a intrusive appliance load monitoring would be to realise

a function like M or M_x, so that given an aggregate signal $y(t)$, a set of appliance consumption/states is returned.

$$M_x(y(t)) = [x_1(t), x_2(t), ..., x_N(t)]$$

Our presented implementation produces a slightly modified version of M_x, namely F: where given a set of characteristics S of the aggregate signal at time t, would return a set of activate labels of appliances N.

$$F(S(t)) = \{i | x_i(t) = 1 \wedge 0 \leq i \leq N\}$$

2.2 Research Questions

We propose to implement an existing method for energy disaggregation capable of recognizing states of known devices from overall power consumption using low frequency data. In other words, an approximation of F. We will test this method on both a standardised data set like REDD[9] and on our own data set containing devices regularly present in households. In addition to that, We will try to improve the existing method. As mentioned by Gao et al.[8], Massidda et al.[6] and researched by MacMackin et al.[11] the power consumption of appliances like fridges and air conditioners are influenced by temperature and weather could thus have an impact on the consumption patterns. In addition to that lights might be turned on sooner when it is raining or cloudy outside. We will investigate to what proportion external factors like temperature, wind and weather data have an effect on the performance of appliance disaggregation by comparing this version to the original. Another aspect of this implementation that shows promise is it's ability to be used as an AS-NIALM method, since it trains on the characteristics of the power signal and not on the signal itself. If this method could be used as an AS-NIALM method, it would remove the need for a setup fase and thus reduce the cost and time of implementing the method in a household drastically. We would thus like to test our implementation on generalizability from one set of households to another, by training the neural network on a set of devices from REDD and testing on our own data where the set of devices is present in both data sets.

this will lead to the following research questions:
- how well does the combination of the two neural nets perform
- how well does a this implementation work on modern standard household appliances.
- to what extend do external factors have effect on the accuracy of this implementation.
- to what extend does training on fake breakpoints have effect on the accuracy of the two neural networks combined.
- how well will the improved version be able to perform across houses.

If a satisfying answer is found to a number of these research questions we will have the following results:
- a more accurate algorithm, written and tested in python, to produce more reliable results for appliance dissagregation

- an implementation able to recognize unknown devices using a general dataset

2.3 Methodology

The implementation we will be testing and extending is based on Gao's multilayer perceptron classifier[8]. We choose this implementation since it shows very good results for REDD: an overall accuracy on the selected appliances of 91.2%, uses an input feature which can be extended to include improvement, utilizes low frequency data instead of high frequency data, which is easier to gather per household, and shows promise to be adapted as an AS-NIALM implantation, because this neural network trains on general features of the power signal instead of the signal itself. Other non intrusive appliance load monitoring techniques like Kelly et al.[7] use the power signal itself as an input feature, making it impossible to extend the feature input with any other data without changing the architecture. Others like [12] need high frequency data to function. They test on a sampling rate of 15.36 kHz. Measuring equipment capable of measuring at this speed can be quite expensive, while low frequency (1Hz) measuring devices are easier to come by, making the preferred implementation in households the multilayer perceptron of Gao et al.

Gao's multilayer perceptron classifier uses two neural network classifiers, the segmentation classifier and the segment labeling classifier to disaggregate an energy signal. First the segmentation classifier breaks the signal up into segments, where a segment is a set of energy consumption where no appliances are turned on or off. This is done by identifying breakpoints, a point on the signal where a device has been turned on or off. The labeling classifier then labels each segment of the segmentation classifier as a single appliance, where each appliance has its own label: a non-appliance segment represented by a 0 or a multi-appliance segment represented by $i+1$ where i is the amount of appliances. The third step consists of disaggregating multi-appliance segments into individual appliances, by using the segment labeling classifier obtained from the second step, untill no multi appliance segments remain.

we do not scale the data, note this somewhere

Feature	Description
mean	average power from the last breakpoint till the current breakpoint
delta	The absolute difference of power values between the current point and the previous point
label	A binary value indicating the breakpoint

Table 2.1: Features used for the breakpoint identifier

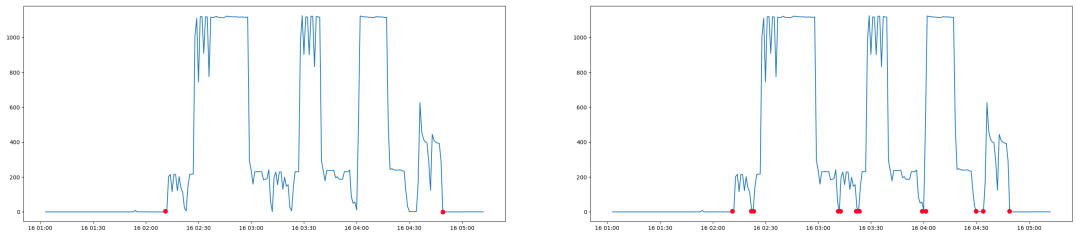
Feature	Description
average	average power of the current segment
min	min power of the current segment
duration	the duration in seconds of the segment.
hour of day	the (mean) hour of day of the segment ranging from [1,24] indicating when the segment took place.
shape	1: a constant segment. 2: a step-down segment or a 3: a varying segment.
previous	The label of the previous segment.
*average temperature	average temperature at the start of the segment
*wind	wind speed at the start of the segment
*weather state	the state of the weather at the start of the segment that could be either 0:rain, 1:cloudy, 2:mist, or 3:sunny.
label	The label of each segment. 0: Empty or Phantom load, 1-N: the N individual appliances; N+1: multiappliance segments.

Table 2.2: Features used for segment labeling. Features marked with * are only used in the improved version of the segment labeler.

Breakpoint identifier

The first multilayer perceptron classifier in Gao's implementation is the breakpoint identifier. This neural network should be able to identify whether an appliance state has changed on a point on the signal. In other words if that specific point is a breakpoint. Using these breakpoints we can create segments, where each segment represents a different combination of appliances. For each point in the data set the classifier will output either a 1, "this point is a breakpoint", or a 0, "this point is not a breakpoint". The breakpoint identifier is trained based on the two features defined in table 2.1, that is, the mean power between this point and the last breakpoint and power difference (delta) between this and the previous point, at each data point of the aggregate data. The resulting feature will be the label. These features are extracted from the aggregated energy signal and the true breakpoints that are trained on are extracted from the separate appliance consumption signals. Goa identified these true breakpoints manually, this is however unfeasible for large quantities of appliances and almost impossible to reproduce. Since we will test this implementation on 10 appliances, 6 more than was tested in the original paper, we decide to identify the breakpoints using a naive algorithm. This algorithm would detect when a appliance was turned on for example by checking if the current value is above a certain power threshold and the previous value is below the same power threshold . For most appliances this would produce correct results, since spikes are averaged out by the sample rate of 1 every minute and appliances are only stated as on if they actually consume power. For pull-sating appliances like the dishwasher, which consumes power in bursts, the definition of a breakpoint correct becomes a bit vague. Take for example a slice of the data from the dishwasher appliance at figure 2.1. If one were to manually identify breakpoints, one would probably go for Gao's, shown in figure 2.1a, since it does to the human eye look like one state. However, one could see also perceive the power consumption as an interval where the

include the table with power threshold data



(a) Gao's true breakpoints (b) our true breakpoints

Figure 2.1: true breakpoint definitions examples for a part of dish washer

dish washer is turned off for a few minutes, before it consumes power again. This approach is shown in figure 2.1b. Testing both variations proved that the breakpoint identifier had a hard time disaggregating the second aproach, mainly due to the increased amount of breakpoints vs non-breakpoints in contrast to the orginal implementation, but proved to increase accuracy for the segment labeler. That is why, apart from the ease of use for larger appliances, we chose to implement the naive method shown in figure 2.1b.

segment labeler

The second multilayer perceptron classifier is the segment labeler. the segments, a part of the power signal seperated by two consecutive breakpoints, created by the breakpoint identifier are labeled by this neural network. The classifier should return a 0, when no appliance is active, a label ranging from 1-N where N is the amount of appliances, representing an appliance, or N+1, representing a multi appliance segment where more than one appliance is active. This neural network is trained on the features defined in table2.2, namely average of the power consumption in the segment, minimum power consumption, shape of the segment, duration of the segment, hour of day it is active and the previous label. The resulting feature will be the label. We create the training and testing input by extracting this information from the signal and create the target output by getting the state of each device from the same time period as the segment.

Multi appliance dissagregation

At this step we try to dissagregate the mutli appliance segments. We do this by first subtracting neighbouring single appliance segments as follows: If the nearest left neighbour that is a single appliance is the same appliance as the nearest neighbour on the right, we will presume the multiappliance segment is lifted by the consumtion of that appliance and subtract its average consumption from the multiapplaince consumption. If the two neighbours are not the same, we will presume the multi applaince segment is lifted by both, and subtract both consumption from the signal. The remaining consumption will be put through the segment labeler again, and the same process will repeat until no multi appliance segments are left. However there are occasions when all neighboring segments have been subtracted and the segment labeler still identifies the segment as a multi appliance segment, which is most likely caused by two appliances turning on at the same time in the dataset. unfortunately

Parameter	value
number of hidden nodes	16
activation	relu
alpha	10^{-5}
learning rate	adaptive

Table 2.3: parameter values for scikit-learns Multilayer perceptron classifier

this implementation is not able to disaggregate those instances.

training the neural network
The classifiers are trained and created to match the specifications in Gao et al.[8]. We use 10-fold cross validation to train the neural network and measure the accuracy and we apply random oversampling[13] to balance each category in the training set by randomly duplicating the appliances with less occurrences in the data. To make the architecture as similar to Gao as possible we use a Multi-layer Perception classifier, the default training algorithm in Python's scikit-learn package[14]. The specifications are listed in table below 2.3. The other parameters are left to the default value of scikit-learns Multilayer perceptron. Note that α is the weight of L2 regularization term in the loss function below.

$$Loss(\hat{y}, y, W) = -yln\hat{y} - (1-y)ln(1-\hat{y}) + \alpha\|W\|_2^2$$

Chapter 3

Related work

shoudl Gao be mentioned here as well?

3.1 Hart

One of the pioneers in this research area was George Hart in the late 1980s with his article on non-intrusive appliance load monitoring (NIALM)[4]. He was the first to differentiate between manual setup non-intrusive appliance load monitoring (MS-NIALM), requiring a one time intrusive setup to get the power consumption data of each individual device in a household, and automatic setup non-intrusive appliance load monitoring (AS-NIALM), where the implementation would recognize the devices and their power consumption without the need for human interference in the setup process. In his article he proposed a energy disaggregation method that looks for sharp edges in both the real and reactive power signals. The method clustered these devices based on their ON/OFF states, Where a device can either be on, consuming a constant amount of power, or off, consuming no power.
This model works for simple appliances like light bulbs, hairdryers or other On/Off devices, but for appliances with multiple states like a TV (On, Standby, Off, Eco mode), this method does not work. An article on NIALM implementations[15] stated that this method can relatively easy detect and track the on-off appliances, but has apparent problems in detecting multi-state and variable-load appliances. Next to that was noted that similar power consumption from two or more different devices may not be separated and due to many appliances change their resistance after they turn on, can create a mismatch within the algorithm as high as 10%. Similar to this we also segment the energy signal using on/off states, but our implementation will try to recognize multi-state appliances as well by training a neural network on it's characteristics.

3.2 Kelly

One of the first succesfull neural network AS-NIALM implementation was the Autoencoder of Kelly et al[7]. In this paper the performance of 3 neural network architectures are tested and compared against eachother as well as the well known combinatorial optimisation or factorial hidden Markov models.
The most notable approach the autor has taken is that, instead of creating one neu-

ral network dissagregating the aggregate signal, it creates a neural network for each appliance. The network should first train on a mix of real data and syntactically created data to make it a more generalize implementation and could then be applied to any house to measure the power consumption of a similar device. Tested on all aplliances the Auto Encoder got an f1 score of .53, precicion of .47 recall of .94 and an accuracy of .93. In addition to that the neural networks use a naive form of syntetic data generation. It does not account for relations between devices or time. For example light might turn on more often when it gets dark around a certain time and if the TV is turned on, chances are that the gaming console will be turned on as well a few seconds later. Another issue is that the train data only contain 5 big devices, but lack most common appliances in households or offices like lights and laptops, which are harder to dissagregate. On the plus side, the Auto Encoder does not give the state of the device, but the actual consumption of the device. That would mean that for multi state appliances like computers it could detect more than just the on and off state. It shows a lot of potential, but needs to be further improved before it can be applied in houses or offices. In this paper we will test our implementation on generalizability. This article proposes a improvement to training the neural network: training on synthetic and real data, where synthetic data is generated to hold 0-2 devices at the most to train the neural network on single appliance signatures as well. Using data sets with a large amount of multi appliance segments vs single appliance segments, this addition could be beneficial to the multi appliance segment disaggregation. This could be researched in the future.

3.3 REDD and python tools

Disaggregation algorithms need to learn a how appliances consume energy from existing data. Such algorithms generally require appliance-level data from either the building which will be disaggregated (sometimes referred to as supervised) or buildings other than the one which will be disaggregated (sometimes referred to as unsupervised). A dataset containing both supervised and unsupervised data is the The Reference Energy Disaggregation Data Set (REDD)[9].
REDD is a freely available data set containing detailed power usage information from six residential houses, with the aim of furthering research on energy disaggregation. An example of the data in REDD can be seen in figure 3.1, excluding unlabeled data like sockets and data with the appliance unknown. REDD is a widely used dataset for energy disaggregation research and makes comparing energy disaggregation algorithms easier. In addition to that REDD provides a general geographic location, namely Massachusetts, US, and timestamps of where and when the measurements where taken. This allows for the inclusion of temperature, wind and weather state data in training and testing NIALM implementations. Do note however that REDD does contain corrupt data in some houses and only measures power in the summer. We will be using this dataset and our own gathered data to train and test the method presented in this paper. REDD will make it easier to compare other NIALM implementations with our implementation.

The non-intrusive load monitoring toolkit (nilmtk)[16] was designed for the purpose of parsing the REDD dataset (and others) as well as build a framework for comparing multiple NIALM algorithms. Although this toolkit is as of now still very

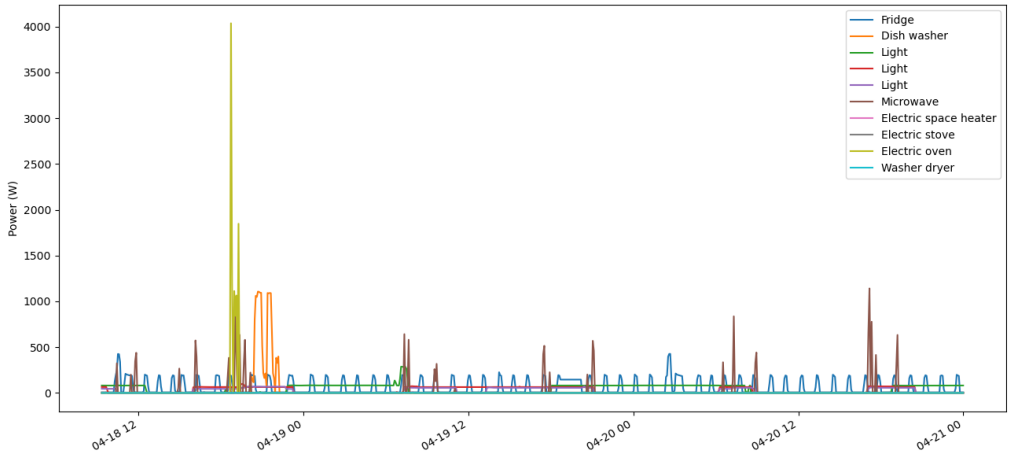


Figure 3.1: example data from the 1 house of REDD. Not included are unlabeled appliances like sockets and unknowns

REDD	max	mean on	mean off	mean on time	activ.	labels	distribution
fridge	441.81 W	197.21 W	7.34 W	1160.09	1285	936	60.35%
microwave	1771.25 W	988.01 W	4.27 W	173.62	564	302	19.47%
washer dryer	3252.12 W	1737.81 W	0.06 W	765.68	176	154	9.93%
dish washer	1171.53 W	638.96 W	0.21 W	1248.54	178	159	10.25%

Table 3.1: characteristics off the data in house 1 of REDD

much in development, much of its components are working and very usefull. We will be using this toolkit for loading and parsing REDD and plotting certain fractions of the REDD data.

for the creation of our Multi-layer Perception classifier we use the scikit-learn[14] package, so that the architecture of our implementation is similar to that of Gao et al.[8]. Scikit-learn also provides built in functionality to measure the performance of the neural network, which we will be using.

To avoid the fact that a model would give a high accuracy due to the imbalance of the data distribution, we apply a random oversampling technique[13] to balance each category in the training set by randomly duplicating the less frequent occurring appliances. This balanced data can then be used for training the neural network.

3.4 WEATHER DATA

For this research we will need access to historical weather data from Massachusetts from 18-4-2011 untill 20-5-2011 for the inclusion of temperature, wind and weather state data in REDD and we will need access to weather data from Groningen from 02-05-2020 till 23-05-2020 for the inclusion of temperature, wind and weather state data in the consumption data we will be gathering ourselves. We will be using the data from timeanddate.com[1] for both datasets. From here we will get the

[1]https://www.timeanddate.com/

min temperature, max temperature, wind speed and weather state (sunny, cloudy, raining, mist) at 0:00, 6:00, 12:00 and 18:00 each day.

Chapter 4

MEASUREMENTS

4.1 DATASET DESCRIPTION

The goal of this dataset is to try to accurately represent the power consumption of the average current household by including appliances in the dataset most often found in households. This means that, apart from big appliances like washing machines, microwaves and water heaters, this dataset also contains relatively low power consuming devices like lights and phone chargers. We chose to also use a selection of devices which would be considered as standard within a present day household like laptops, mobile phone chargers and gaming consoles, but are not present in REDD[9]. Mostly the data present in REDD consists of high power appliances with distinct signatures, lacking low powered appliances like chargers, which according to Y. Gao[8] is much more difficult to disaggregate. Our own gathered data will thus be harder to disaggregate but it will provide a better overview of the efficiency of our algorithm for more difficult devices. Do note however that the gathering of this data took place mostly in the month of may 2020, which was at the peak of the corona virus pandemic, where people were instructed to stay home. Because of the quarantine most appliances ran all day, where normally they would just be on for a few hours. Because of the increased usage, the dataset might be slightly more difficult to disaggregate than a normal household.
For an overview of all appliances and a their consumption characteristics see table 4.1, where the corresponding maximum power consumption, average consumption when on and average consumption when off are given for each appliance, as well as the average duration of their on state, number of times the appliance has been turned on/off (activ.), the amount of segments in which the appliance is on (and thus contains its label) and the distribution in the dataset.

We gathered the consumption data for the following devices:

Figure 4.1: TV and ps4

A TV
or a Philips 40PFL4358H/12 3D TV to be precise, which can be seen in figure 4.1. We chose this appliance because a TV is present in almost every household today and consumes a decent amount of power. According to the label on the back of this device, this TV will consume 41W-60W when on. As you can see on table 4.1 this predicting is quite accurate. In this data set the TV was used mostly only during the day, that includes mornings and evenings and is on for around an hour or more. Regarding relations to other devices, when the TV is turned on, the Play Station 4 almost always follows, since the Play Station 4 is used for almost all activities with this TV.

Figure 4.2: phone charger

phone charger
or a Samsung galaxy A-70 with a thunderbolt charger to be precise, which can be seen in figure 4.2. We chose this appliance, because the phone charger is a rather new, but frequently present, device in the current household and because a phone charger is a very low power consuming device. One of the lowest in this data set. Including a low power consuming device allows us to see how well the implementation does on such devices. This device was mostly plugged in before bedtime, and consumed power until the phone was fully charged.

Figure 4.3: desk lamp

desk lamp
or a 30W lamp to be precise, which can be seen in figure 4.3. We chose this appliance because lights are relatively low power consuming, but in large quantities present in households. In addition to that we expect lights to be influenced by the weather, since one might turn on a lamp when its dark and cloudy outside. This lamp, as with all lights, are mainly used in the evening or during cloudy or rainy weather.

Figure 4.4: living room lamp

living room lamp
or a 50W lamp to be precise, which can be seen in figure 4.4. This lamp is rather old and has several options for lighting: dim, full, off. Because of that it makes use of a power brick, which consumes a little bit of power even when off, as you can see from table 4.1. We chose this appliance for the same reasons as the desk-lamp, but also to include more than one light. A household will most likely contain a

multiple of lights, with this device included in our data set we can see how well the implementation will disaggregate between the two lights. This lamp was mostly used during the evening or on cloudy afternoons and this lamp was not as frequently used as the desk-lamp.

Figure 4.5: washing machine

washing machine
or a BEKO WTV71483CSB with energy label A+++ to be precise, which can be seen in figure 4.5. We chose this machine for its presence in most households, but also for its energy consumption pattern. A washing machine will consume power in bursts, making it more challenging to detect. The inclusion of this device will give the opportunity to see how well the implementation does on these kind of devices. The washing machine was used at any time of the day, but mostly in the afternoon.

Figure 4.6: fridge

fridge
or the ikea LAGAN fridge/freezer with energy label A++ to be exact, which can be seen in figure 4.6. We chose this machine for its presence in most households, but also for the fact that a fridge is one of the only frequently available household devices which turns itself on periodically. This device will run day and night and

the power consumption could be influenced by weather and temperature.

Figure 4.7: water heater

water heater
or the OK. OWK 103 to be exact, which can be seen in figure 4.7. According to the label the water heater can consume up to 2200W, as we can see from table 4.1 the water heater doesn't reach this value, but is one of the most high power consuming devices in the data set. We chose this appliance to have a short but high power consuming device present in the data set. The water heater consumes a lot of power at once, but only for about 3 minutes and is used only once or twice a day. Making it a perfect example for short but high power consuming devices.

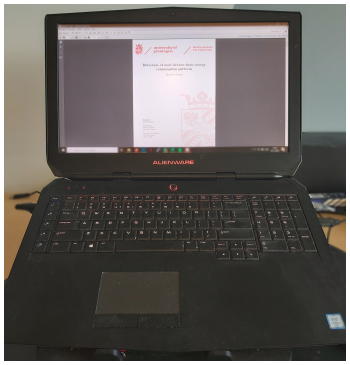


Figure 4.8: laptop

laptop
or an Alienware R17 Gaming laptop to be precise, which can be seen in figure 4.8. We chose this appliance because since the 90's, computers can be considered a standard household appliance. This is a high performance laptop, meaning that it will consume more power than most. Especially when it is used for gaming or training a neural network. It is also very varying in power consumption. ranging from 150W to 50W or less. The laptop is most likely to be the first device to be turned on and will be on for most of the day.

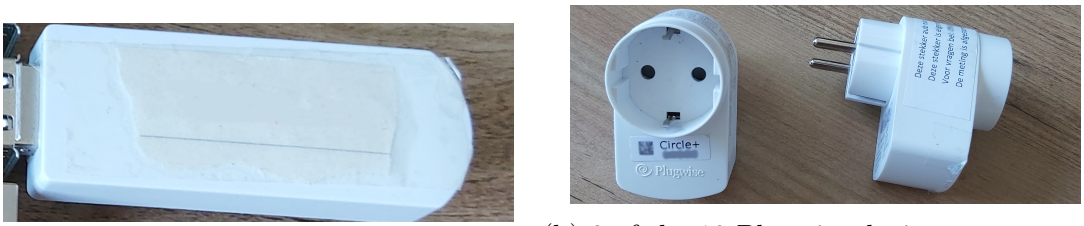
PlayStation 4
or the PlayStation 4 original 500GB to be precise, which can be seen in figure 4.1. We choose this device to see how well the implementation is able to disaggregate devices that rely on each other. This PlayStation for example, will almost never be turned on if the TV isn't turned on. And if the TV is turned on, the PlayStation will likely follow. However when the tv is turned off, the Play Station may still remain on (because the resident is forget-full sometimes). This inclusion of the Play Station 4 gives the ability to test how well an implementation is able to disaggregate a segment where two appliances are turned on at once.

Figure 4.9: microwave

microwave
or a KOENIC KMW 4441 microwave/oven to be precise, which can be seen in figure 4.9. This device consumes between 1450-2500W of power when on, according to the label. This is pretty accurate when compared to the data from table 4.1. As you can see this device consumes the most amount of power at once in the data set. We chose this appliance mostly for its presence in the household, but also for its high power consumption. The microwave was at any time of the day, but long activation's where most present during the evening.

studio data	max	mean on	mean off	mean on time	activ.	labels	distribution
TV	63.73 W	57.73 W	0.16 W	3831.79	78	687	12.79%
phone charger	25.55 W	9.73 W	0.23 W	126.01	746	404	7.52%
desk lamp	36.08 W	34.80 W	0.00 W	3287.65	34	87	1.62%
couch lamp	53.23 W	44.49 W	4.07 W	5457.50	8	76	1.41%
washer	2066.00 W	885.62 W	1.68 W	48.88	734	381	7.09%
fridge	703.65 W	55.08 W	0.04 W	1594.75	644	863	16.06%
water heater	1988.55 W	1959.33 W	0.05 W	180.00	14	11	0.20%
laptop	143.96 W	58.97 W	0.18 W	32226.00	41	1616	30.08%
PlayStation 4	152.00 W	97.12 W	8.31 W	11886.00	60	1011	18.82%
microwave	2509.10 W	1960.11 W	0.92 W	38.57	446	236	4.39%

Table 4.1: characteristics of appliance power consumption



(a) the Stick

(b) 2 of the 10 Plugwise devices.
(1 Circle+, 1 Circle)

Figure 4.10: measuring devices

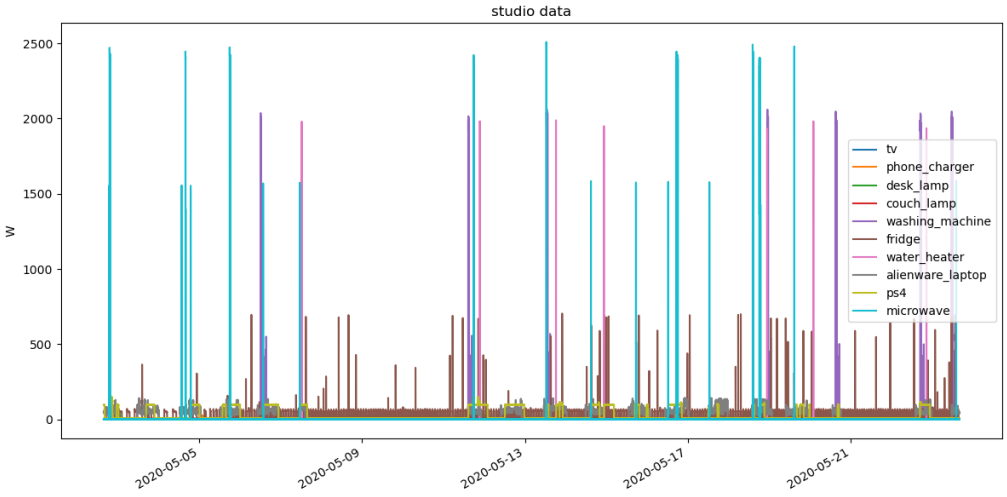


Figure 4.11: plot of the studio data including all individual appliances

4.2 APPROACH

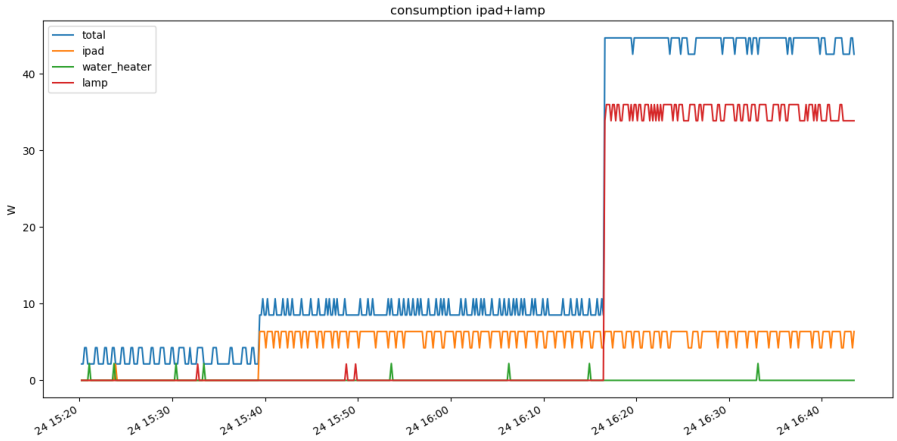
Apart from the REDD dataset we will be using our own dataset for training and testing our implementation. This dataset contains the power consumption data of 10 devices from a 1 person studio apartment in Groningen. We measured the power consumption data every 10 seconds, since any higher gathering per second increases the changes of NaN values significantly, and gathered data for a duration of 3 weeks. The data was gathered using a total of 10 Plugwise[1] devices and a USB stick called a 'Stick' as shown in figure 4.10. The set of these Plugwise devices communicated over a wireless ZigBee mesh network around one coordinator, shown in figure 4.10b with the 'Circle +' label. The coordinator then in turn communicates with the 'Stick' allowing the gathering of data by plugging the Stick into a pc. A very similar setup was previously implemented for research on optimizing energy costs for offices connected to the smart grid[3]
The python-Plugwise library[2] was used to create a small script to communicate with the Stick and request the data from the Plugwise devices. This script was run on a dedicated laptop for 3 weeks.
For the application of the data to our neural network, we chose a sampling period of 60 seconds for more accurate comparison with gao's tests on REDD, which also

[1]http://www.plugwise.com
[2]https://bitbucket.org/hadara/python-plugwise/wiki/Home

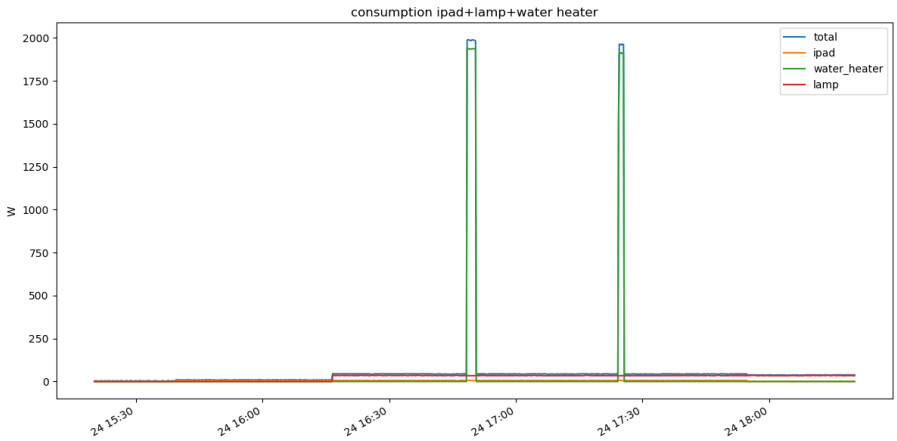
use a sampling period of 60 seconds. We use foward filling on empty or NaN values in the data and store the values in a pandas dataframe[17]. A plot of the result of the data gathering can be seen in figure 4.11

4.3 AGGREGATE DATA SUMMATION EXPERIMENT

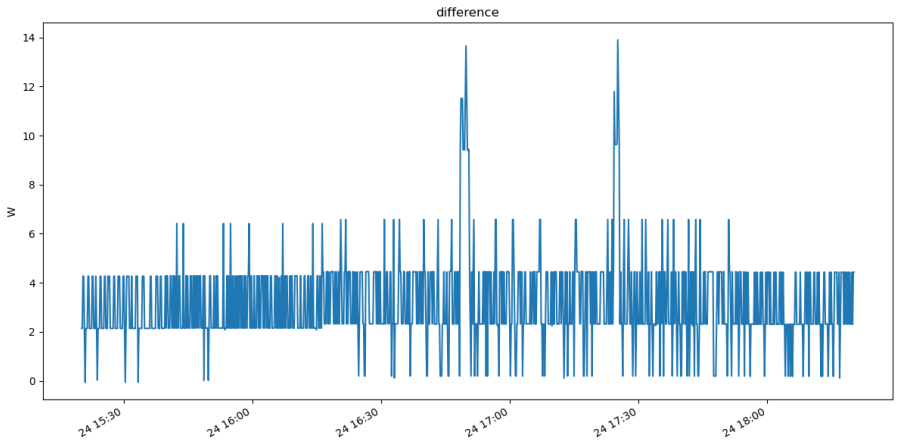
The aggregate data was calculated by summing up the total appliance consumption. To evaluate whether or not this method would produce correct aggregate data we will execute a small experiment:
3 devices connected to Plugwise devices (Circle's), an iPad charger, a lamp and a water heater, are added to a extension cord which in turn is connected to a plug-wise device, the Circle+, so that the Plugwise device was gathering the power consumption information of the 3 devices. See figure 8.1 for the actual setup.
The consumption of each device and the total consumption was measured for a total of 3 hours every 10 seconds. In these 3 hours some of the devices are turned of or on to measure the change in total and individual power.
The results of this experiment can be seen in figure 4.12. in figure 4.12a it is already visible that the total consumption is always around 2.5W higher than everything else. Even when all appliances are turned off. Also in figure 4.12b and figure 4.12c it is clear that difference lies between 2W and 4W most of the time. It is also clear that the difference seems to grow larger when a large appliance is turned on. The difference of 2W and 4W can be explained by the power consumption of the Plugwise devices. We have 3 Plugwise devices connected to an extension cord, one Plugwise devices has an average consumption of 1.1W [3], which means that the average consumption on the total would at least be 3.3W Explaining the small difference.
The spikes could however indeed be an sign of the consumption of power due to the resistance in the extension cord. Since the Plugwise devices should not be part of the equation in a real household and the spike only represents 0.5% of the total power consumption, we will use the summed appliances for our aggregate power consumption data. Since implementations like Gao et al.[8] and Kelly et al.[7] use the same approach for their aggregate data, we will be able to create and test an implementation of the algorithms closer to the originals.



(a) consumption iPad+lamp



(b) consumption ipad+lamp+waterheater



(c) difference actual vs summed appliance consumption

Figure 4.12: actual total consumption vs summed appliance consumption outcomes

Chapter 5

Experiments

5.1 Experiment 1: Original implementation

First we test our implementation on performance and compare it to mutlilayer perceptron presented in the article by Gao et al.[8]. As stated before, the method presented in this article consists of 3 steps: the breakpoint identifier, the segment labeler and the multi appliance disaggregation step. We will test all 3 of these steps on REDD and compare the results with Gao et al. to see how similar our implementation is. We will also test these steps on our own gathered studio data to spot how well the implementation performs on a bigger quantity and more varying set of modern household appliances. In addition to that we will test how well the segment labeler does given the breakpoints which are identified by the breakpoint identifier on both data sets in the combined implementation section. This result will tell us how well the implementation as a whole will perform on household appliances, which is not stated in the article of Gao et al.

5.1.1 REDD

To spot how similar our implementation is to Gao et al., we will be testing the implementation on REDD, in which we choose the following 4 appliances: a fridge, a microwave, a washer dryer and a dish washer. Note that Gao et al. tests on the following 4 appliances: a fridge, a microwave, a washer dryer and an air conditoner, but due to the fact that house 6 and 5 containt corrupted data[8] and measures power on dates where we do not have weather data on, we choose not to include those houses. This means we do not have data on the air-conditioning. Instead, we decided to disaggregate the dish washer.
Just like Gao et al., for each appliance, we down-sample the power signal from 1 Hz to one measurement per minute. We obtain our experimental data by aggregating and adding up the power signals from all chosen appliances in the same house.
The results should be as similar to the results in Gao et al., since we used the same architecture. But the testing is done on slightly different data. We do not test on house 6 and include the dish washer instead of the air conditioner. Another point of difference is that the dish washer consumes power in pulses, so it could be harder to disaggregate. Lastly Gao et al. manually identifies breakpoints, while we let an

classification report	precision	recall	f1-score	support
non-breakpoint	1.00	0.96	0.98	10988
breakpoint	0.53	0.93	0.67	532

Table 5.1: classification report of the breakpoint identifier using REDD with an accuracy of 95.83%

algorithm detect the breakpoints.

Breakpoint identifier
The first classifier we test is the breakpoint identifier. The classification report of the breakpoint identifier can be seen in table 5.1 depicting the performance of the implementation. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[14] we measured an accuracy of **98%** with a standard deviation of **1%**. Gao doesn't specifically note the accuracy of the breakpoint identifier, but states that the breakpoint identifier "performs nearly perfectly in identifying breakpoints and non-breakpoints". Keeping in mind that Gao has a lower ratio of breakpoint vs non-breakpoint than our implementation due to the inclusion of the dishwasher and the fact that we do not manually label the breakpoints, we can conclude that this implementation is sufficiently similar to gao's breakpoint identifier.
In the confusion matrix of table 8.1 we see that most breakpoints are correctly identified, but 377 non-breakpoints are labeled as breakpoints. However the segment labeler might be able to solve the incorrect marked breakpoints. We will analyse the combined accuracy later.

segment labeler
The second classifier, the segment labeler, is up next. The classification report of the segmnet labeler can be seen in 5.2 depicting the performance of the implementation. The confusion matrix can be seen in table 8.2. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[14] we measured an accuracy of **88%** with a standard deviation of **3%**. In the article of Gao et al. an accuracy of **0.945** was measured. This difference in accuracy is mainly due to the absence of house 6 in the training data, resulting in less training points for fridge. Considering only the segments labeled as single appliance or empty by the segment labeler, **417** out of **482** appliance or empty instances were labelled correctly, where **48** were mislabeled and **17** not labeled at all (due to the segment being labeled as a single appliance while being a multi appliance segment).

multiappliance labeler
The last step of the method proposed by Gao et al. is disaggregating the multi-appliance segments further. We test this step on the **55** multi appliance segments from the testing data. From the total of **220** single appliance labels within the multi appliance segments **156** were correctly identified. Combing these results by adding

classification report	precision	recall	f1-score	support
empty	1.00	0.98	0.99	213
fridge	0.95	0.82	0.88	187
microwave	0.78	0.82	0.80	44
washer dryer	0.11	0.18	0.13	11
dish washer	0.21	0.27	0.24	22
multi	0.57	0.69	0.62	55

Table 5.2: classification report of the segment labeler using REDD with an accuracy of 83.46%

up the total and correct predicted labels like Gao we end up with **603** out of **710** correct. It got an accuracy of **70.9%** Compared to Gao's **91.2%**. Note that **13** multi appliance segments were not able to be disaggregated at all by the method proposed by Gao et al.

a table with disaggregated appliances would be nice

combined implementation

These results are quite promising, but what is left out of the article from Gao et al. is the performance of the implementation as a whole. Or in fact, how well the segment labeler does on the output of the breakpoint identifier. This test is important, since it is the accuracy of the combined implementation that matters when the neural nets are put into practice.

To test the implementation as a whole, we first gave the input to the breakpoint indentifier as we did before. With the output from the breakpoint indentifier we look up the states of each of the given breakpoints and use that as our target for the segment labeler. Using the breakpoint from the breakpoint identifier and the aggregate signal, the input is created and tested on the segment labeler. the results can be seen in table 8.3 and table 5.3. As you can see, the accuracy of **70%** for the combined test is considerably lower than the standalone implementation. It is also visible that combined implementation performs very poorly on identifying multi-label appliances, indicating that the multi-appliance disaggregation will be nearly impossible to do accurately. To test the multi appliance disaggregation we apply the same original tactic of multi appliance disaggregation, but with the breakpoints given by the breakpoint identifier and the labels given by the segment labeler. From the **121** segments labeled as multi appliance by the segment labeler only **345** out of **528** of the labels were found correctly. This is an accuracy of **65%**.

conclusion

With an accuracy of **0.98** and **0.90** for the breakpoint identifier and the segment labeler respectivly and an overall accuracy of **84%**, we conclude that the implementation is similar and efficient enough to use as a benchmark against our proposed improvements. We do regonize that, although we use the same dataset, we train and test only on 4 houses, instead of 6. We may also split the point between training and testing data differently then Gao. Resulting in less points to test on then Gao. Regarding the results of the combined method, it can be seen that the current implementation is far from perfect and still needs work before it can be put into practice

classification report	precision	recall	f1-score	support
empty	1.00	0.69	0.81	365
fridge	0.71	0.86	0.77	363
microwave	0.55	0.54	0.55	72
washer dryer	0.12	0.15	0.13	13
dish washer	0.34	0.18	0.24	55
multi	0.38	0.68	0.49	73

Table 5.3: classification report of the segment labeler with breakpoints given by the breakpoint identifier using REDD with an accuracy of 70.46%

5.1.2 STUDIO DATA SET

Here we propose to test our implementation of the multilayer perceptron classifier against our own data set and compare it with the results of experiment 1. Our data set is created with the intention to represent a modern day household. Where REDD only includes household appliances like fridge, washing machine and a microwave and some general appliances like lights, other common appliances our data set also includes appliances like a laptop, phone charger, TV and gaming console. With this experiment we investigate how well this implementation works on these devices and how well it is able to handle a large quantity of varying devices, 10 to be exact. Due to the increased amount of different appliances, and the inclusion of a laptop, which is on almost the entire day, the amount of multi-appliance segments will be substantial. Thus with this experiment we can also see how well the multistate disaggregation performs as well.
We down-sample the power signal from 0.1 Hz to one measurement per minute so that the sampling will be similar to REDD. We obtain our experimental data by aggregating and adding up the power signals from all appliances.

Breakpoint identifier
The first classifier we test on our own data is the breakpoint identifier. The classification report of the breakpoint identifier can be seen in table 5.4 depicting the performance of the implementation. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[14] we measured an accuracy of **98%** with a standard deviation of **2%**. Gao doesn't specifically note the accuracy of the breakpoint identifier, but states that the breakpoint identifier "performs nearly perfectly in identifying breakpoints and non-breakpoints". Keeping in mind that Gao has a lower ratio of breakpoint vs non-breakpoint than our implementation due to the inclusion of the dishwasher and the fact that we do not manually label the breakpoints, we can conclude that this implementation is sufficiently similar to gao's breakpoint identifier.
In the confusion matrix of table 8.4 we see that most breakpoints are correctly identified, but 339 non-breakpoints are labeled as breakpoints. However the segment

classification report	precision	recall	f1-score	support
non-breakpoint	0.99	0.95	0.97	7213
breakpoint	0.48	0.90	0.62	347

Table 5.4: classification report of the breakpoint identifier using the studio data with an accuracy of 95.04%

classification report	precision	recall	f1-score	support
empty	1.00	0.87	0.93	67
tv	0.50	1.00	0.67	1
phone charger	0.43	1.00	0.60	9
desk lamp	0.00	0.00	0.00	4
washing machine	0.00	0.00	0.00	1
fridge	0.93	0.93	0.93	59
water heater	0.00	0.00	0.00	1
alienware laptop	0.72	0.73	0.73	45
ps4	0.00	0.00	0.00	0
multi	0.98	0.89	0.93	160

Table 5.5: classification report of the segment labeler using the studio data with an accuracy of 86.17%

labeler might be able to solve the incorrect marked breakpoints. We will analyse the combined accuracy later.

segment labeler
The second classifier, the segment labeler, is up next. The classification report of the segmnet labeler can be seen in 5.5 depicting the performance of the implementation. The confusion matrix can be seen in table 8.5. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[14] we measured an accuracy of **85%** with a standard deviation of **9%**. One can see that although the accuracy remains the same, most of the appliances have very little to no data point. This is because due to the large quantity of devices in the dataset, a lot of the segments in this dataset are multi-appliance segments. This is why with the studio data as training data the implementation seems to be very good at mutli state dissagregation. Since a lot of single states are missing from the training data as well, the segment labeler will most likely have a lot of trouble dissagregating the single states that are present. Furthermore the microwave doesn't seem to be present in the reports, meaning there was not one single single-appliance label of a microwave in the result data. This does not look good for the multi appliance disaggregation.

multiappliance labeler
The last step of Gao's multilayer perceptron is disaggregating the multi-appliance segments further. We test this step on the **160** multi appliance segments from the testing data. From the total of **1600** single appliance labels within the multi appliance segments **1206** were correctly identified. so it got an accuracy of **75.3%**. Do

classification report	precision	recall	f1-score	support
empty	0.98	0.79	0.88	72
tv	0.06	1.00	0.11	1
phone charger	0.09	0.67	0.15	3
desk lamp	0.00	0.00	0.00	5
fridge	0.85	0.28	0.42	101
water heater	0.00	0.00	0.00	1
alienware laptop	0.68	0.24	0.36	111
ps4	0.00	0.00	0.00	0
multi	0.85	0.91	0.88	357

Table 5.6: classification report of the segment labeler with breakpoints given by the breakpoint identifier using the studio data with an accuracy of 67.74%

note that **22** multi appliance segments were not able to be disaggregated at all by the method proposed by Gao et al. This accuracy is quite surprising, but is mainly due to the fact that the TV, Alienware laptop and fridge occupy a considerable portion of the single appliance labels in the multi appliance segments, which the segment labeler performs well on. the segment labeler seems to struggle with appliances like the ps4, washing machine and other small devices.

combined implementation

We saw that the segment labeler experiences quite some problems with the current way of how it is trained. But how does the combined implementation hold up? It turns out it performs relatively the same as with the REDD dataset, as can be seen in table 5.6. This test on the combination of the two neural networks also gives us insight into the breakpoint identifier. For example we can see that the breakpoint identifier produces far less breakpoints for the phone, and way to much for the laptop. This is because the laptop has a very varying power consumption, while the phone consumes almost no power. To test the multi appliance disaggregation we apply the same original tactic of multi appliance disaggregation, but with the breakpoints given by the breakpoint identifier and the labels given by the segment labeler. From the **385** segments labeled as multi aplliance by the segment labeler only **2961** out of **3850** of the single appliance labels were found correctly. This is an accuracy of **76.9%**. Not considerably worse than the stand alone implementation, but still not sufficiently accurate.

conclusion

The accuracy of the implementation on the STUDIO data remains almost the same as REDD, however the method seems to not be able to disaggregate a handfull of appliances. This is mainly due to the lack of single appliance activations in the data set. The segment labeler simply does not have enough to train on. Considering these shortcomings the multi appliance labeler and the combined implementation still perform relatively well. With a respective accuracy of **75.3%** and **76.9%**.

classification report	precision	recall	f1-score	support
empty	1.00	0.96	0.98	213
fridge	0.93	0.86	0.89	187
microwave	0.73	0.73	0.73	44
washer dryer	0.33	0.27	0.30	11
dish washer	0.24	0.50	0.33	22
multi	0.54	0.56	0.55	55

Table 5.7: classification report of the improved segment labeler using REDD with an accuracy of 83.08%

5.2 Experiment 3: external features

considering the results from experiment 1 and 2, we would like to propose some changes to the original implementation to try and improve it to work better for multiple appliances and to make the two combined neural networks work better together. The first change to the implementation will be the inclusion of temperature wind and weather state data in the features of the segmentation labeler as shown in table 2.2 as the features marked with *. With this inclusion we expect appliances influenced by the temperature, like a fridge, or appliances influenced by cloudy, sunny or rainy weather, like lights, easier to disaggregate. The second change is to make the segment labeler train on false breakpoints. For the training data we use a 50/50 distribution between false and actual breakpoints. To make a false breakpoint we choose a random point on the signal and place a breakpoint there. We hope that this will increase the accuracy of the combination of the two neural networks. We will test the changes of the segment labeler on both REDD and the studio data. Lastly, we will check the possible changes to the overall implementation and check if the added features have an impact on the output by analysing the weights.

5.2.1 REDD

segment labeler
The classification report of the segment labeler can be seen in 5.7 depicting the performance of the implementation. The confusion matrix can be seen in table 8.7. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[14] we measured an accuracy of **88%** with a standard deviation of **4%**. Compared to the original implementation there is not much change. This is mainly due to the fact that the fridge is one of the only appliances that could be influenced by temperature, but when looking at the fridge itself there is only a slight increase in accuracy.

multi appliance The last step of Gao's multilayer perceptron is disaggregating the multi-appliance segments further. We test this step on the **55** multi appliance segments from the testing data. From the total of **220** single appliance labels within the multi appliance segments **176** were correctly identified. so it got an accuracy of **80.0%**. Note that only **4** multi appliance segments were not able to be disaggregated by the method proposed by Gao et al.

classification report	precision	recall	f1-score	support
empty	1.00	0.68	0.81	352
fridge	0.63	0.90	0.74	264
microwave	0.64	0.32	0.43	115
washer dryer	0.00	0.00	0.00	121
dish washer	0.25	0.11	0.15	84
multi	0.39	0.84	0.53	184

Table 5.8: classification report of the improved segment labeler with breakpoints given by the breakpoint identifier using REDD with an accuracy of 60.54%

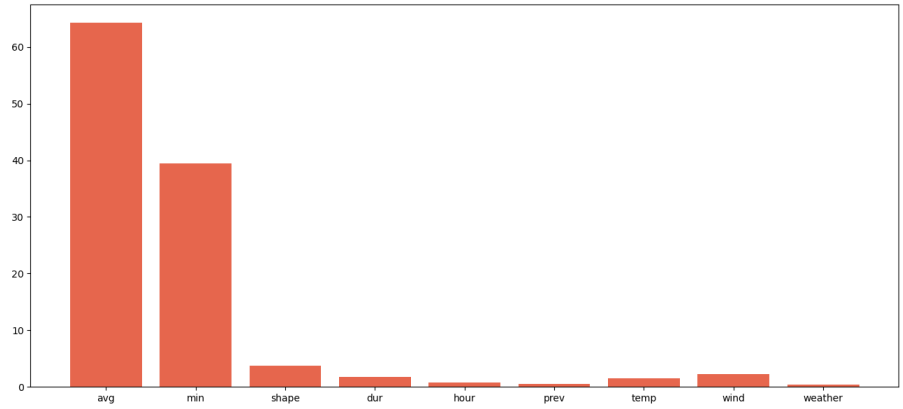


Figure 5.1: analysis of the weights of the improves segment labeler on REDD

combined implementation
from the performance metrics depicted in table 5.8 and the confusion matrix in table 8.8 it can be seen that no significant increase in performance was detected. So it seems that training on false breakpoints does not increase the accuracy. The multi appliance disaggregation actually performed worse. From the **398** segments labeled as multi aplliance by the segment labeler only **789** out of **1592** of the single appliance labels were found correctly. This is an accuracy of **49.6%**.

analysis of weights
The accuracy of the implementations alone are not enough to see if the weather, wind and temperature feature in the neural network actually have an impact in making the neural network produce a the correct result, because with the inclusion of new features, the architecture of the neural net changes as well. To see if the features have an impact, we choose to analyse the absolute weights of the first layer. We average the weights of each feature and multiply that by the average value of the feature to get an indication of the influence to the outcome. the result can be seen in figure 5.1.

As expected do the average and min power consumption have the most influence on the second layer. What also can be seen is that although shape has a maximum value of 3, it still has a decent influence in the outcome. Regarding the temperature and wind they have about as much influence in the outcome as shape and duration. Apparently the weather state is as good as ignored. Note that this test reflects how

classification report	precision	recall	f1-score	support
empty	1.00	0.93	0.96	67
tv	1.00	1.00	1.00	1
phone charger	0.75	1.00	0.86	9
desk lamp	0.00	0.00	0.00	4
washing machine	0.00	0.00	0.00	1
fridge	0.88	0.95	0.91	59
water heater	0.00	0.00	0.00	1
alienware laptop	0.78	0.62	0.69	45
ps4	0.00	0.00	0.00	0
multi	0.97	0.91	0.94	160

Table 5.9: classification report of the improved segment labeler using the studio data with an accuracy of 86.74%

much a feature contributes to an outcome, not just the correct outcome.

5.2.2 STUDIO

segment labeler
Because the proposed changes only affect the segment labeler, the breakpoint identifier will not be changed. The classification report of the segment labeler can be seen in 5.9 depicting the performance of the implementation. The confusion matrix can be seen in table 8.9. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[14] we measured an accuracy of **88%** with a standard deviation of **11%**. Compared to the original implementation this is still fairly similar.

multi appliance
The last step of Gao's multilayer perceptron is disaggregating the multi-appliance segments further. We test this step on the **160** multi appliance segments from the testing data. From the total of **1600** single appliance labels within the multi appliance segments **1216** were correctly identified. so it got an accuracy of **76.0%**. Note that **26** multi appliance segments were not able to be disaggregated at all by the method proposed by Gao et al.

combined implementation
Looking at the combined implementation of the improved version, of which the results can be seen in table 5.10 and table 8.10, we detect no real change in accuracy. So it seems that training on false breakpoints does not increase the accuracy. The multi appliance disaggregation performed as follows: From the **352** segments labeled as multi appliance by the segment labeler only **2519** out of **3520** of the single appliance labels were found correctly. This is an accuracy of **71.6%**.

classification report	precision	recall	f1-score	support
empty	1.00	0.74	0.85	84
tv	0.08	1.00	0.14	1
phone charger	0.50	1.00	0.67	4
desk lamp	0.00	0.00	0.00	6
washing machine	0.00	0.00	0.00	1
fridge	0.84	0.65	0.73	108
water heater	0.00	0.00	0.00	1
alienware laptop	0.76	0.42	0.54	132
ps4	0.00	0.00	0.00	0
multi	0.86	0.91	0.88	333

Table 5.10: classification report of the improved segment labeler with breakpoints given by the breakpoint identifier using the studio data with an accuracy of 73.88%

classification report	precision	recall	f1-score	support
non-breakpoint	1.00	0.97	0.98	29528
breakpoint	0.38	0.81	0.52	712

Table 5.11: classification report of the improved breakpoint identifier using a combination of the 2 datasets with an accuracy of 96.47%

5.2.3 CONCLUSION

Overall the results don't seem to show any improvement in the accuracy. This is mainly due to the lack of temperature dependant devices in the household and lack of difference in temperature in the data set. It can also be seen that training on fake breakpoints does not increase the performance of the combined implementation, and on one occasion even causes a decrease in accuracy.

5.3 EXPERIMENT 4: GENERALIZEABILITY

Here we will test the generalize ability of the improved implementation. Or in other words, how well the implementation is able to disaggregate appliances from an unkown household. To test this we will train the two Multilayer perceptrons on the big devices that REDD and the studio data have in common, namely the fridge, the microwave and the washing machine. The result will tell us how well this implementation will perform as a AS-NIALM method. Due to a considerable difference in the power consumption of the devices, as can be seen in table 3.1 and table 4.1, a low accuracy is predicted. However if we do receive an accuracy on par with the other test, this means that the implementation is very well suited to be used as a AS-NIALM method.

breakpoint identifier The breakpoint identifier performs better than expected, correctly identifying a good amount of breakpoints as can be seen in table 5.11 and table 8.11. The accuracy of **96.47%** is also very good, considering the difference in testing and training appliances.

segment labeler

classification report	precision	recall	f1-score	support
empty	0.94	0.70	0.81	327
fridge	0.73	0.95	0.82	319
microwave	0.60	0.17	0.26	18
washing machine	0.69	0.47	0.56	19
multi	0.62	0.72	0.67	29

Table 5.12: classification report of the improved segment labeler using a combination of the 2 datasets with an accuracy of 79.49%

classification report	precision	recall	f1-score	support
empty	0.60	0.56	0.58	580
fridge	0.48	0.65	0.55	581
microwave	0.32	0.17	0.22	70
washing machine	0.18	0.01	0.02	193
multi	0.28	0.45	0.34	87

Table 5.13: classification report of the improved segment labeler with breakpoints given by the breakpoint identifier using a combination of the 2 datasets with an accuracy of 49.70%

The segment labeler also does considerably well, having an accuracy of **79.49%**. As you can see from table 5.12 and table8.12 the appliances are being disaggregated with a decent accuracy.

multi appliance disaggregation
We test the multi appliance disaggregation step on the **29** multi appliance segments from the testing data. From the total of **87** single appliance labels within the multi appliance segments **61** were correctly identified. so it got an accuracy of **70.0%**. Note that **1** multi appliance segments was not able to be disaggregated at all by the method proposed by Gao et al. This is still surprisingly accurate

combined implementation
Looking at the combined implementation, of which the results can be seen in table 5.13 and table 8.13, we detect an accuracy of **49.70%**. Which is a bit lower than expected considering the results of the individual tests, but still very promising for a AS-NIALM method. The multi appliance disaggregation performed as follows: From the **140** segments labeled as multi appliance by the segment labeler only **185** out of **420** of the single appliance labels were found correctly. This is an accuracy of **41.1%**.

5.3.1 CONCLUSION

As we can see from the tests the performance of the individual neural networks are quite good. Proving that the power consumption of the appliances over the data sets do not have to be exactly the same, the method is able to learn the characteristics of the appliances. The combined test however does not perform as well. Although

this is still on par with other AS-NIALM methods like the one presented by Kelly et al.[7].

Chapter 6

Conclusion

anwser research questions

overall the implementations seem to produce correct results for big appliances, but does not work well on large quantities of appliances. This is mostly due to the large quantity of multi state appliances vs single state appliances in the training dataset. The generalization also did quite well. Is it really AS NIALM? because we do need to train on a big dataset. mention this!

Chapter 7

Future improvements

syntetic data generation might be beneficial to multi appliance dissagregation. "We found that synthetic data acts as a regulariser. In other words, training on a mix of synthetic and real aggregate data rather than just real data appears to improve the net's ability to generalise to unseen houses. For validation and testing we use only real data (not synthetic)." neural net article. There was no rain in the month that was gathered. Also no cold/ very warm temperatures. More data needs to be gathered (on year basis). To get mroe accurate results.

Chapter 8

Appendix

confusion matrix	non-breakpoint	breakpoint
non-breakpoint	10544	444
breakpoint	36	496

Table 8.1: confusion matrix of the breakpoint identifier using REDD

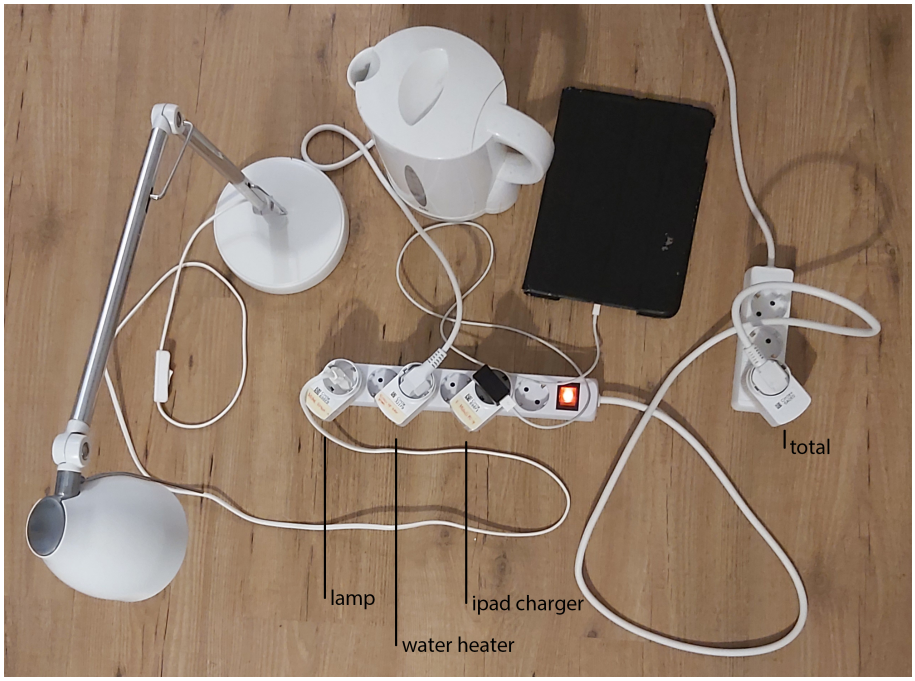


Figure 8.1: setup sum of individual consumption experiment

confusion matrix	empty	fridge	microwave	washer dryer	dish washer	multi
empty	209	3	0	0	0	1
fridge	1	153	0	1	16	16
microwave	0	0	36	2	1	5
washer dryer	0	0	0	2	3	6
dish washer	0	4	4	7	6	1
multi	0	1	6	7	3	38

Table 8.2: confusion matrix of the segment labeler using REDD

confusion matrix	empty	fridge	microwave	washer dryer	dish washer	multi
empty	251	110	0	1	0	3
fridge	1	311	19	2	8	22
microwave	0	3	39	1	1	28
washer dryer	0	2	1	2	1	7
dish washer	0	10	7	6	10	22
multi	0	4	5	5	9	50

Table 8.3: confusion matrix of the segment labeler with breakpoints given by the breakpoint identifier using REDD

confusion matrix	non-breakpoint	breakpoint
non-breakpoint	6874	339
breakpoint	36	311

Table 8.4: confusion matrix of the breakpoint identifier using the studio data

confusion matrix	empty	tv	phone charger	desk lamp	washing machine	fridge	water hea
empty	58	0	9	0	0	0	0
tv	0	1	0	0	0	0	0
phone charger	0	0	9	0	0	0	0
desk lamp	0	0	0	0	0	1	0
washing machine	0	0	0	0	0	0	0
fridge	0	0	1	1	0	55	0
water heater	0	0	0	0	0	0	0
alienware laptop	0	1	0	0	0	1	0
ps4	0	0	0	0	0	0	0
multi	0	0	2	0	0	2	0

Table 8.5: confusion matrix of the segment labeler using the studio data

confusion matrix	empty	tv	phone charger	desk lamp	fridge	water heater	alienware lapt
empty	57	4	6	2	2	0	0
tv	0	1	0	0	0	0	0
phone charger	1	0	2	0	0	0	0
desk lamp	0	1	0	0	1	0	3
fridge	0	10	12	5	28	0	9
water heater	0	0	0	0	0	0	0
alienware laptop	0	2	0	0	1	0	27
ps4	0	0	0	0	0	0	0
multi	0	0	3	2	1	0	1

Table 8.6: confusion matrix of the segment labeler with breakpoints given by the breakpoint identifier using the studio data

confusion matrix	empty	fridge	microwave	washer dryer	dish washer	multi
empty	205	7	0	0	1	0
fridge	0	160	0	1	22	4
microwave	0	0	32	0	2	10
washer dryer	0	0	0	3	1	7
dish washer	0	4	1	1	11	5
multi	0	1	11	4	8	31

Table 8.7: confusion matrix of the improved segment labeler using REDD

confusion matrix	empty	fridge	microwave	washer dryer	dish washer	multi
empty	241	111	0	0	0	0
fridge	0	237	2	2	17	6
microwave	0	4	37	0	0	74
washer dryer	0	2	1	0	1	117
dish washer	0	20	5	3	9	47
multi	0	4	13	4	9	154

Table 8.8: confusion matrix of the improved segment labeler with breakpoints given by the breakpoint identifier using REDD

confusion matrix	empty	tv	phone charger	desk lamp	washing machine	fridge	water hea
empty	62	0	3	0	0	2	0
tv	0	1	0	0	0	0	0
phone charger	0	0	9	0	0	0	0
desk lamp	0	0	0	0	0	1	0
washing machine	0	0	0	0	0	0	0
fridge	0	0	0	0	0	56	0
water heater	0	0	0	0	0	0	0
alienware laptop	0	0	0	1	0	1	0
ps4	0	0	0	0	0	0	0
multi	0	0	0	2	0	4	0

Table 8.9: confusion matrix of the improved segment labeler using the studio data

confusion matrix	empty	tv	phone charger	desk lamp	washing machine	fridge	water hea
empty	62	0	4	10	0	6	0
tv	0	1	0	0	0	0	0
phone charger	0	0	4	0	0	0	0
desk lamp	0	0	0	0	0	3	0
washing machine	0	0	0	0	0	0	0
fridge	0	11	0	13	0	70	0
water heater	0	0	0	0	0	0	0
alienware laptop	0	1	0	3	0	2	0
ps4	0	0	0	0	0	0	0
multi	0	0	0	4	0	2	0

Table 8.10: confusion matrix of the improved segment labeler with breakpoints given by the breakpoint identifier using the studio data

confusion matrix	non-breakpoint	breakpoint
non-breakpoint	28596	932
breakpoint	134	578

Table 8.11: confusion matrix of the improved breakpoint identifier using a combination of the 2 datasets

confusion matrix	empty	fridge	microwave	washing machine	multi
empty	230	96	0	1	0
fridge	12	303	0	3	1
microwave	0	7	3	0	8
washing machine	1	3	2	9	4
multi	1	7	0	0	21

Table 8.12: confusion matrix of the improved segment labeler using a combination of the 2 datasets

confusion matrix	empty	fridge	microwave	washing machine	multi
empty	323	247	4	0	6
fridge	193	375	6	5	2
microwave	1	15	12	1	41
washing machine	21	110	8	2	52
multi	4	34	7	3	39

Table 8.13: confusion matrix of the improved segment labeler with breakpoints given by the breakpoint identifier using a combination of the 2 datasets

Bibliography

[1] B. Naghibi and S. Deilami. Non-intrusive load monitoring and supplementary techniques for home energy management. In *2014 Australasian Universities Power Engineering Conference (AUPEC)*, pages 1–5, 2014.

[2] Jakob Stoustrup, Anuradha M. Annaswamy, Aranya Chakrabortty, and Zhihua Qu. chapter Smart grid control : overview and research opportunities. Power electronics & power systems. Springer, Cham, Switzerland, 2019.

[3] G. A. Pagani T. A. Nguyen A. Lazovik I. Georgievski, V.Degeler and M. Aiello. Optimizing energy costs for offices connected to the smart grid. *IEEE Transactions on Smart Grid*, 3(4):2273–2285, 2012.

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

[5] Khaled Chahine. Towards automatic setup of non intrusive appliance load monitoring – feature extraction and clustering. *International Journal of Electrical and Computer Engineering (IJECE)*, 9:1002, 04 2019.

[6] Luca Massidda, Marino Marrocu, and Simone Manca. Non-intrusive load disaggregation by convolutional neural network and multilabel classification. *Applied Sciences*, 10:1454, 02 2020.

[7] Jack Kelly and William J. Knottenbelt. Neural NILM: deep neural networks applied to energy disaggregation. *CoRR*, abs/1507.06594, 2015.

[8] A. Schay Y. Gao and D. Hou. Home appliance detection from aggregated energy consumption data on a single circuit. *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1–8, 08 2017.

[9] J. Z. Kolter and M. J. Johnson. Redd: A public data set for energy disaggregation research. *Workshop on Data Mining Applications in Sustainability (SIGKDD)*, 25:59–62, 2011.

[10] Kalyana. Pentapati and Saurabh. Kumar. Vampire power in dentistry: Should we be concerned? *Journal of Dental Research and Review*, 2(3):141–142, 2015.

[11] [Nick MacMackin], Lindsay Miller, and Rupp Carriveau. Modeling and disaggregating hourly effects of weather on sectoral electricity demand. *Energy*, 188:115956, 2019.

[12] J. Gillis and W. G. Morsi. Non-intrusive load monitoring using orthogonal wavelet analysis. In *2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–5, May 2016.

[13] Guillaume Lemaundefinedtre, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *J. Mach. Learn. Res.*, 18(1):559–563, January 2017.

[14] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[15] Michael Zeifman and Kurt Roth. Nonintrusive appliance load monitoring: Reviewandoutlook. *Consumer Electronics, IEEE Transactions on*, 57:76 – 84, 03 2011.

[16] O. Parson H. Dutta W. Knottenbelt A. Rogers A. Singh M. Srivastava N. Batra, J. Kelly. Nilmtk: An open source toolkit for non-intrusive load monitoring. *5th International Conference on Future Energy Systems (ACM e-Energy)*, 2014.

[17] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.