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
July 1, 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..

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 measuring equipment. 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 only on a selection of large non-varying devices. A decent working implementation still needs to be found.

In this paper we will look at an existing Neural network implementation which uses breakpoint identification to disaggregate their aggregate power consumption using low frequency data proposed by Gao et al.[8]. Breakpoints are points in time where the state of an appliance is changed or in other words, when an appliance is turned on or off. This implementation 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 Research Questions

We propose to improve an existing method for energy disaggregation capable of recognizing states of known devices from overall power consumption using low frequency power consumption data.

The implementation we will be using is the Gao's multilayer perceptron classifier[8] consisting of two neural nets, one to identify breakpoints and one to label each power consumption segment separated by breakpoints. We choose this implementation for several reasons. First of all, it presents very good results: an overall accuracy on the selected appliances of 91.2% using REDD[9].

Secondly, it uses an input feature which can be extended to include external features. 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.

Thirdly, it utilizes low frequency data instead of high frequency data. implementations like that of Gillis et al.[10] need high frequency data to function. They test on a sampling rate of 15.36 kHz. Measuring equipment capable of measuring at this speed requires special instrumentation, while low frequency (1Hz) measuring devices are easier to come by[11]. Low frequency is thus more feasible for implementation on a large scale.

Finally, it shows promise to be adapted as an AS-NIALM implantation. Because of the fact that this neural network trains on general features of the power signal instead of the signal itself, it could be possible to apply this implementation to other households without first training on the present devices. A pre-trained model could in that case just be hooked up to the aggregate energy signal without needing a one time intrusive setup fase at all. It would reduce the cost and time of implementing this model in a household drastically. Therefore, we would like to test our implementation on generalizability from one set of households to another.

We will try to improve this implementation with the inclusion of temperature, wind and weather data in the input features. As mentioned by Gao et al.[8], Massidda et

al.[6] and researched by MacMackin et al.[12] the power consumption of appliances like fridges and air conditioners are influenced by temperature and weather and these factors could 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 the improved version to the original.

In addition to that we will train the neural network on synthetic breakpoints to try and improve the overall accuracy. The purpose of this improvement is to gain a better accuracy on the implementation as a whole, where all the steps are linked together. In this case, the second neural network will get the output of the first one as its input. But this input, the breakpoints in this case, could consists of false breakpoints. We speculate that if we train the second neural network on false breakpoints, it'll be able to better predict the state of appliances even though no state has been changed. Normally, the segment labeler is trained and tested on input which always have distinct appliance states.

We will test this method on both a standardised data set, namely REDD[9], and on our own data set. We will create this data set with the goal of including more modern and regularly present devices in households. Since we do not have a way of measuring the aggregate data in this dataset and we will be using a subset of appliances in REDD, we need to find out if we can get this data by summing up the power consumption of the devices. We will undertake a small experiment to see whether we can do this.

To encapsulate this, we would like to find answers to the following research questions:
- can the existing implementation be improved with inclusion of weather related external factors.
- can the existing implementation be improved with training on fake breakpoints

To find an answer to the research questions, we must first find an answer to the following:
- How well does the combination of the two neural nets perform with and without improvements.
- How well does the implementation work on modern standard household appliances with and without improvements.
- to what extend do external factors have an effect on the result of the implementation.
- to what extend does training on fake breakpoints have effect on the accuracy of the two neural networks combined.
- how well will the implementation perform as an AS-NIALM method.

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 disaggregation.
- an implementation able to recognize unknown devices using a general data set.

2.2 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 [13].

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.

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

2.3 Methodology

The method we will be using to solve the previously defined problem is Gao's multilayer perceptron classifier[8]. This method uses two neural network classifiers: The segmentation classifier, which we will refer to as the breakpoint identifier, and the segment labeling classifier, which we will refer to as the segment labeler.

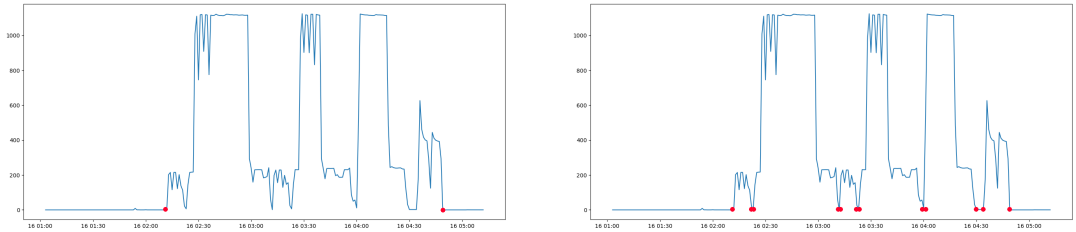
This method works as follows: First the segmentation classifier breaks the signal up into segments, where a segment is a set of energy consumption where no appliances are turned on or off. This is done by identifying breakpoints, a point on the signal where a device has been turned on or off. The labeling 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, until no multi appliance segments remain.

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

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 activated 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. The resulting feature will be the label indicating that this point in time is a breakpoint or not.

These features are extracted from the aggregated energy signal and the true breakpoints are extracted from the individual 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 by checking if the current value is above a certain power threshold and the previous value is below the same power threshold for example. For most appliances this would produce correct results, since spikes are averaged out by the sample rate of 1 minute and appliances are only stated as on if they



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

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

actually consume power. For pulsating appliances like the dishwasher, which consumes power in bursts, the definition of a correct breakpoint becomes a bit vague. Take for example a slice of the data from the dishwasher 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 also perceive the power consumption as an interval where the 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 original 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.

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.

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 the segments from the aggregate signal using the true breakpoints and create the target output by retrieving the state of each device from the same time period as the segment.

Multi appliance disaggregation
At this step we try to disaggregate 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 multi appliance segment is lifted by the consumption of that appliance and subtract its average consumption from the multi appliance consumption. If the two neighbours are not the same, we will presume the multi appliance segment is lifted by both, and subtract both the respective power 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 data set. Unfortunately this implementation is not able to disaggregate those instances.

training the neural network
The classifiers are created and trained 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[14] 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[15]. 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.

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

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

Chapter 3

Materials

3.1 Related work

Gao et al.
Our base implementation is based on the one presented by Gao et al.[8]. In his article he created the energy disaggregation method which classified the segments using breakpoint identification. The implementation was tested on REDD and a subset of Clarkson University's Smart Housing dataset, comprising of the power consumption from 6 apartments or 25 bedrooms. On REDD the calculated accuracy was 91.2% at the end. Showing great potential for implementation in households. We choose to use this implementation for it's good results, extendable input feature, the use of easily gathered low-frequency data and the promise of it begin adapted as a AS-NIALM algorithm.

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[16] 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. We will use the terminology of MS-NIALM and AS-NIALM presented by Hart in this article to differentiate between NIALM methods requiring manual setup versus NIALM that do not.

Kelly et al.
One of the first successful neural network AS-NIALM implementation was the Autoencoder of Kelly et al.[7]. In the 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 author has taken is that, instead of creating one neural network disaggregating 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 appliances 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 synthetic data generation. We will use a form of this synthetic data generation for the inclusion of fake breakpoints in our training data. It is worth noting however that in this case the synthetic data generation 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. This is not a problem for the breakpoints however, since we assume that the breakpoint occurrence is independent from weather.

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.

3.2 DATA SETS

REDD
Disaggregation algorithms need to learn 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 unsu-

pervised). A data set containing both supervised and unsupervised data is the The Reference Energy Disaggregation Data Set (REDD)[9].
REDD is a freely available and widely used data set containing detailed power usage information from six residential houses, with the aim of furthering research on energy disaggregation and making comparisons between energy disaggregation algorithms easier. Moreover, REDD provides a general geographic location for their houses, 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, although this data does need to be retrieved from another source. Do note however that REDD does contain corrupt data in some houses and only measures power in the summer of 2011.
We will be using this dataset to train and test the method presented in this paper. REDD will make it easier to compare Gao's original implementations with our implementation.

studio data
Apart from REDD we will also use our own gathered data to test and train the implementation. This data will represent the power consumption of a one person studio apartment in 2020. For this reason we will reference to this data set as the studio data set. The set of devices is chosen with the goal of trying to represent frequently present modern devices in a household. For example, with the inclusion of a phone charger, gaming console and a tv. The data has been gathered with a sampeling rate of once every 10 seconds and the measurements took place for 3 weeks. From the 2th of May to the 23th of May.

weather data
For this research we will need access to historical weather data from Massachusetts from 18-4-2011 until 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 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. More information on both REDD, the studio data and the weather data can be found in Chapter 4.

3.3 TOOLS AND LIBRARIES

The non-intrusive load monitoring toolkit (nilmtk)[17] 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 much in development, much of its components are working and very useful. We will be using this toolkit for loading and parsing REDD and plotting certain fractions of REDD data.

for the creation of our Multi-layer Perception classifier we use the scikit-learn[15] package, so that the architecture of our implementation is similar to that of Gao et

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

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[14] 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.

Chapter 4

Data

4.1 REDD

one of the data sets used by Gao et al. is the The Reference Energy Disaggregation Data Set (REDD)[9]. We will be using this data set too, to be able to accurately compare our implementation with the one presented in the paper by Gao et al.[8].

4.1.1 dataset description

REDD contains the low-frequency power consumption of individual appliances and the low or high-frequency aggregate power consumption of 6 houses. The measurements have been taken at roughly the same time, from the 15th of april up till the 1st of June, for a little more than a month per house. In this paper, we are only interested in the low-frequency power consumption per appliance. Like Gao, we will only test on a subset of usefull devices and will thus not be able to use the given aggregate data. A subset is chosen since appliances like "unkown" and sockets present in most houses are not particularly use full. This low frequency data is sampled every 3 seconds.

Devices present in REDD range from standard big appliances like a Fridge, Dish washer, Microwave, electric oven and washer dryer, but also small appliances like lights and smoke detectors. As previously mentioned REDD also contains a lot of "unknown" and socket appliances. These could be for example a vacuum cleaner or a hair dryer, but since there is no way to know for sure which appliance these sockets represent, they are left out. Moreover, some subset of appliances are only present in one household and not in others. Since we will be training and testing

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 4.1: characteristics off the data in house 1 of REDD

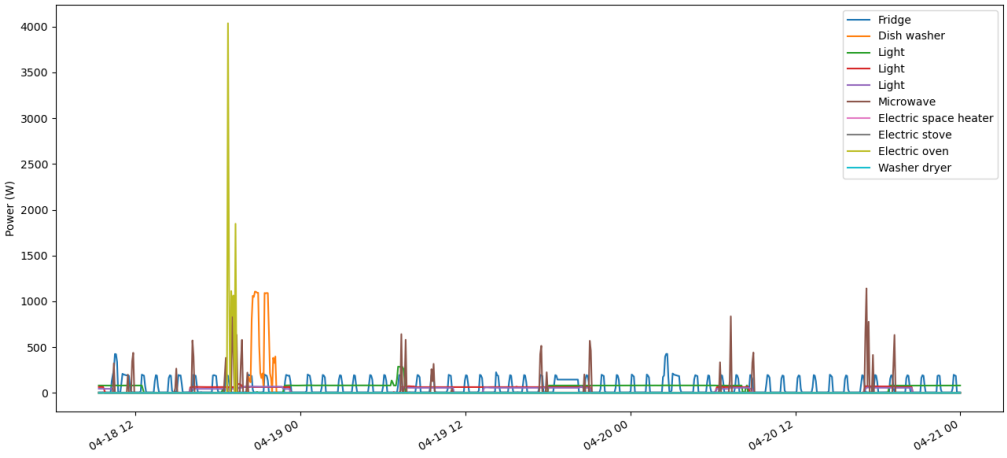


Figure 4.1: example data from the 1 house of REDD. Not included are unlabeled appliances like sockets and unknowns

on a combination of the houses, it is important that the devices are present in the training set as well as in the house picked as a testing set. That is why we, just as Gao et al., picked a small subset of appliances that is most common across houses. This subset can be seen in table 4.1 along with the corresponding power consumption characteristics of the appliances.

One disadvantage of REDD is the corrupted data in a number of houses. In these houses there is a gap in energy consumption data for around a week. We solved this issue by selecting only the dates on which this corrupted data is not present for training and testing. An example of the data in REDD can be seen in figure 4.1, excluding unlabeled data like sockets and data with the unknown appliances.

4.1.2 APPROACH

As described in the article by Kolter et al.[9]: For the gathering low-frequency plug-level data, a wireless plug monitoring system developed by Enmetric[1] was used. Appliances can be connected to an extension cord which would send the power consumption data via DHCP to a central server at a rate of 1Hz. Because the system reports at a sufficient rate it was mostly left as-is.

Since whole house level data and high-frequency data is also present in REDD, the whole setup was more complicated. For the gathering of such data certain "REDDboxes" where used per house to have low-frequency measuring equipment, high-frequency measurement equipment and a hub for processing the incoming data all in one place. Since we won't be using the data gathered by the "REDDboxes", the details are out of the scope of this article.

[1] http://www.enmetric.com

4.2 Studio data

Apart from REDD, we also use our own gathered data from a one person studio apartment in Groningen. A description of the data set itself and the appliances is shown below. In addition to that we will discuss the approach of how we got to the data in the section following it.

4.2.1 data set description

The goal of this data set is to try to accurately represent the power consumption of the average current household by including appliances most often found in households. This means that, apart from big appliances like washing machines, microwaves and water heaters, this data set 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 data set might be slightly more difficult to disaggregate than a normal household.

For an overview of all appliances and a their consumption characteristics see table 4.2, 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.

included devices:
Here follows a list of of appliances that were included in the data set. The corresponding details, expected power consumption, approximate usage pattern and possible relations to other devices will be explained per appliance.

Figure 4.2: TV and ps4

A TV
or a Philips 40PFL4358H/12 3D TV to be precise, which can be seen in figure 4.2. 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 4.2 this prediction is quite accurate. In this data set the TV was used mostly during the day. that includes mornings and evenings. it is on for around an hour or more per day. 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 4.3: phone charger

phone charger
or a Samsung galaxy A-70 with a thunderbolt charger to be precise, which can be seen in figure 4.3. We chose this appliance because the phone charger is a rather new, but frequently present, 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 in the data set 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 4.4: desk lamp

desk lamp
or a 30W lamp to be precise, which can be seen in figure 4.4. 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 4.5: living room lamp

living room lamp
or a 50W lamp to be precise, which can be seen in figure 4.5. 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 4.2. 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 4.6: washing machine

washing machine
or a BEKO WTV71483CSB with energy label A+++ to be precise, which can be seen in figure 4.6. 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 4.7: fridge

fridge
or the ikea LAGAN fridge/freezer with energy label A++ to be exact, which can be seen in figure 4.7. 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 4.8: water heater

water heater
or the OK. OWK 103 to be exact, which can be seen in figure 4.8. According to the label the water heater can consume up to 2200W, as we can see from table 4.2 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.

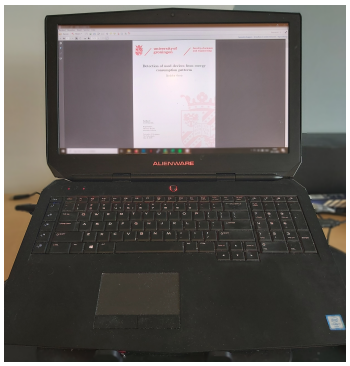


Figure 4.9: laptop

laptop
or an Alienware R17 Gaming laptop to be precise, which can be seen in figure 4.9. We chose this appliance because since the beginning of the 21st century, 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.

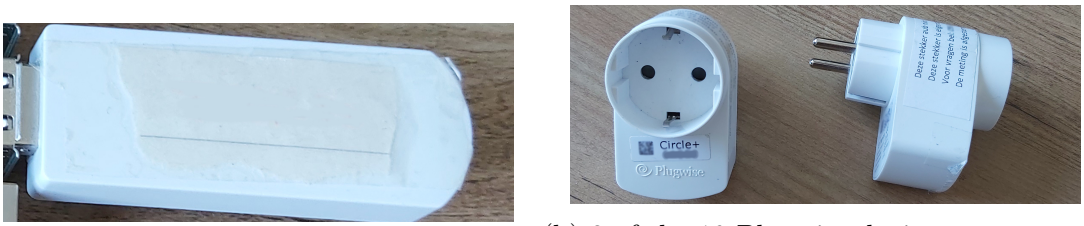
PlayStation 4
or the PlayStation 4 original 500GB to be precise, which can be seen in figure 4.2. We choose this device to see how well the implementation is able to disaggregate devices that rely on each other. This PlayStation for example, will almost never be turned on if the TV isn't turned on. And 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 PlayStation 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 4.10: microwave

microwave
or a KOENIC KMW 4441 microwave/oven to be precise, which can be seen in figure 4.10. 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 4.2. 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 active 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 4.2: characteristics of appliance power consumption



(a) the Stick

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

Figure 4.11: measuring devices

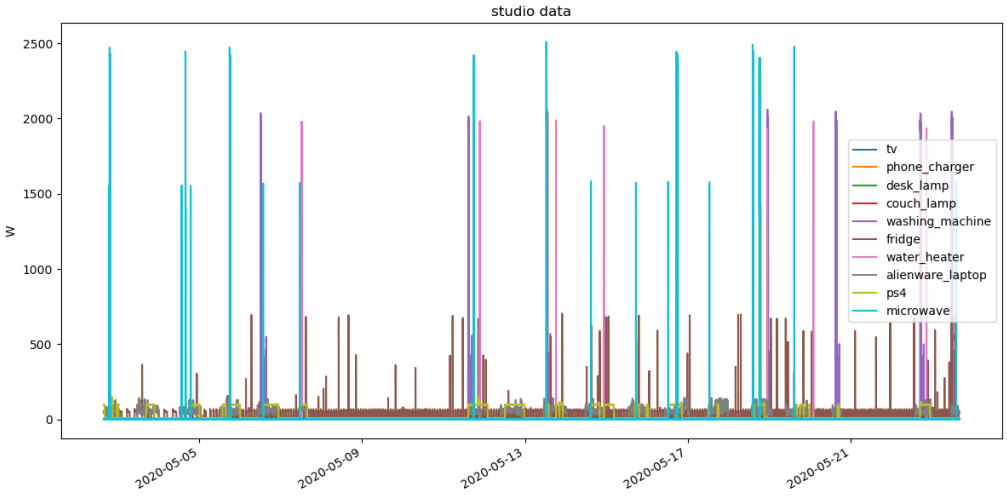


Figure 4.12: plot of the studio data including all individual appliances

4.2.2 APPROACH

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[2] devices and a USB stick called a 'Stick' as shown in figure 4.11. The set of these Plugwise devices communicated over a wireless ZigBee mesh network around one coordinator, shown in figure 4.11b 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[3] 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 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[18]. A plot of the result of the data gathering can be seen in figure 4.12

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

	avg temp	std temp	avg wind	std wind	rain	cloud	mist	sun
REDD	12.6	4.1	17.6	7.2	9.6%	41.0%	35.9%	13.5%
studio	10.7	5.5	12.5	6.7	0%	22.7%	11.4%	65.9%

Table 4.3: characteristics of the weather data set

4.3 WEATHER DATA

The weather data used in this article was retrieved from timeanddate.com[4] for the location of Massachusetts from 18-4-2011 until 20-5-2011 and Groningen from 02-05-2020 till 23-05-2020 to fit our data. For each day the data is split up as follows: each day has a measurement point. One at 0:00, at 6:00, at 12:00 and one at 18:00. At each of these points in time the corresponding max temperature, min temperature, wind speed and weather state (sunny, cloudy, raining, mist), is available. This data was formatted into a .csv file.

The usage of this data within the implementation proceeds as follows: When the power consumption of a specific point in time was about to be disagregated, the nearest known time to contain weather data was picked from the data set and put into the input feature. The weather state was converted into integer numbers ranging from 0: raining, 1: cloudy, 2: mist, 3: sunny.

A description of the dataset can be seen in table 4.3. Here the average and standard deviation of the temperature and wind speed is displayed for the data corresponding to a particular energy consumption data set. The distribution of the weather state over the data is also given. Note that it hasn't rained once during the time of measurement of the studio data.

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

Chapter 5

Experiments

Here we will present the experiments with which we will try to answer the research questions. We will first begin with a small experiment to see if we can get the aggregate energy consumption data from the sum of the individual power consumption of the devices. After that we will test our benchmark implementation on REDD and compare it to Gao, as well as test it on our studio data and compare it to the previous results on REDD. Following that experiment is the accuracy test of the addition of the stated improvements. These results will be compared to the benchmarks which have been created in the previous experiment. Lastly, we will test the improved version on generalizability by training on a subset of data in REDD and testing it on a subset of devices in our studio data.

For the calculation of the performance scores, the following equations were used, where tp is the number of true positives, fp the number of false positives and fn the number of false negatives.

$$accuracy = \frac{\#correctly\ detected\ instances}{\#instances}$$

$$precision = \frac{tp}{tp + fp}$$

$$recall = \frac{tp}{tp + fn}$$

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

5.1 aggregate data summation experiment

Before we start with testing the implementation, we first need to evaluate if we can create the aggregate data by summing up the power consumption of the individual devices. To evaluate whether 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 gathers the power consumption information of the 3 devices. See figure 8.1 for the actual setup.

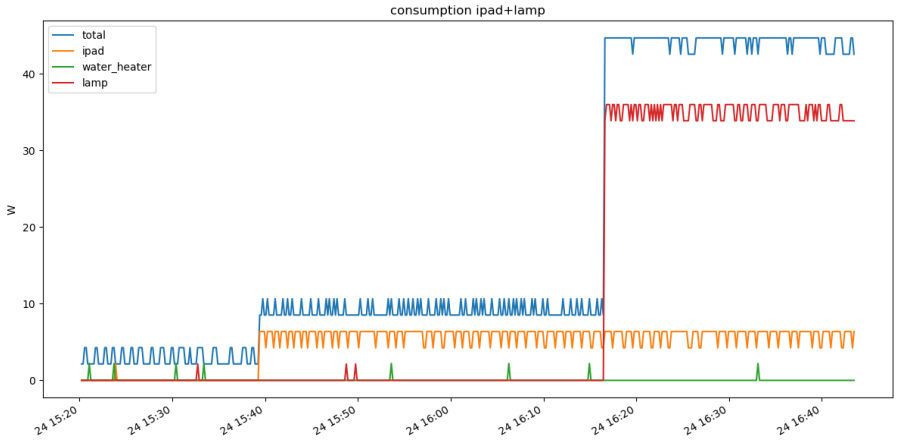
The consumption of each device and the total consumption was measured for a total of 3 hours every 10 seconds. In these 3 hours some of the devices are turned of or on to measure the change in total and individual power.

The results of this experiment can be seen in figure 5.1. In figure 5.1a 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 5.1b and figure 5.1c 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.

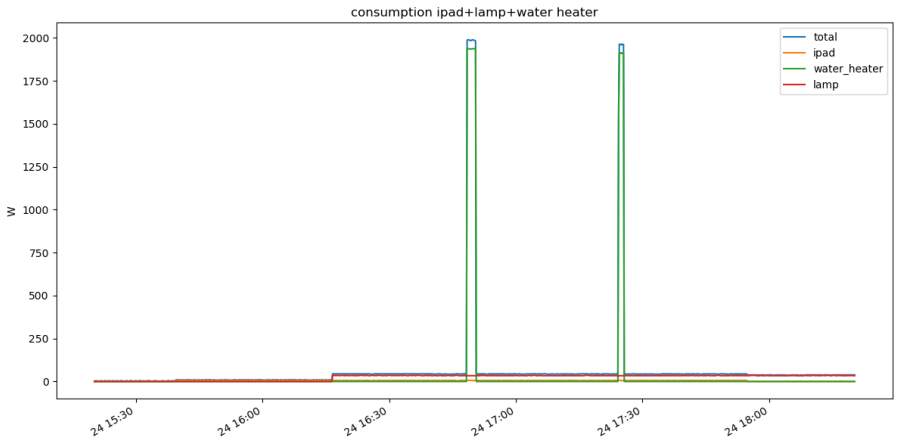
The difference of 2W and 4W can be explained by the power consumption of the Plugwise devices. We have 3 Plugwise devices connected to an extension cord, one Plugwise device has an average consumption of 1.1W [3], which means that the average consumption on the total would at least be 3.3W Explaining the small difference.

The spikes in difference between the sum and the actual total power could however indeed be an sign of resistance in the extension cord. Meaning that when a lot of power is consumed, the sum will not precisely represent the aggregate data.

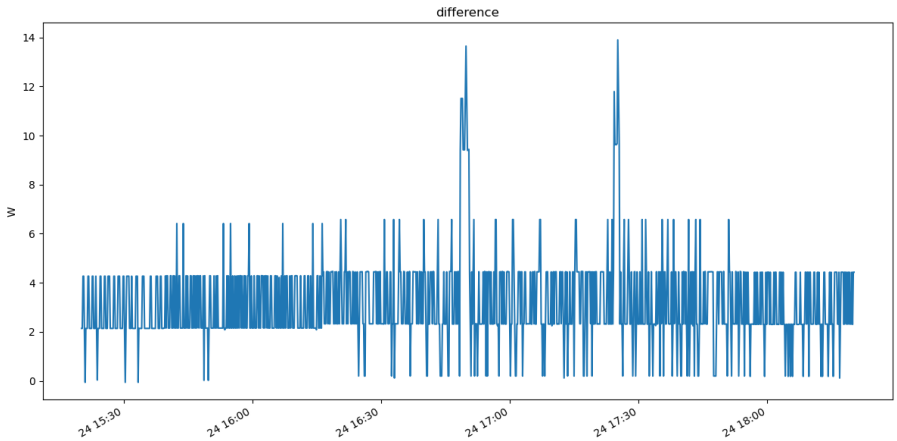
Since the Plugwise devices should not be part of the equation in a real household and the spike only represents 0.5% of the total power consumption, we will use the summed appliances for our aggregate power consumption data. Since implementations like Gao et al.[8] and Kelly et al.[7] use the same approach 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 5.1: actual total consumption vs summed appliance consumption outcomes

5.2 The original implementation

Here we test our implementation on performance and compare it to multilayer perceptron presented in the article by Gao et al.[8]. The purpose of these experiments is to create a benchmark for the improved version. 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 performs 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.

5.2.1 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 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 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[15] we measured an accuracy of **97%** with a standard deviation of **3** using the data with random oversampling and **98%** with a standard deviation of **1** without. 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 con-

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%

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%

clude that this implementation is sufficiently similar to gao's breakpoint identifier. In the confusion matrix of table 8.1 in the appendix 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 segment labeler can be seen in 5.2 depicting the performance of the implementation. The confusion matrix can be seen in table 8.2 in the appendix. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[15] we measured an accuracy of **88%** with a standard deviation of **3** without using random over-sampling. In the article of Gao et al. an accuracy of **94.5%** 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. Using the same cross validation method on random over-sampled data we get an accuracy of **66%** with a standard deviation of **5**. In the over-sampled data, appliances that are normally not frequently present are now more present in the training/testing data. So this result does show some over-fitting. 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).

multi appliance disaggregation

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. This is an accuracy of **70.5%**. Combing these results by adding up the total and correct predicted labels like Gao

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%

we end up with **603** out of **710** correct. So the total accuracy is **84.5%** 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.

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 identifier as we did before. With the output from the breakpoint identifier we looked up the states of each of the given breakpoints and used that as our target for the segment labeler. Using the breakpoints 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 in the appendix and table 5.3. As you can see, the accuracy of 70.5% 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. Putting all of the steps together, the implementation is able to disaggregate **3316** out of the **3764** labels correctly. This is an accuracy of **88.1%**

conclusion
The point of this experiment was, as mentioned, to see how well this implementation is able to mirror the results presented in the article by Gao et al.[8]. The breakpoint identifier seems to perform well considering both the cross validation result of **97%** accuracy and the classification report with an accuracy of **95.8%**. Gao et al. does not specifically mention the exact accuracy of the breakpoint identifier, only that it "performs near perfectly". It should also be noted that we used a naive algorithm to find the true breakpoints, while Gao et al. identified them manually. This probably

caused a slight drop in accuracy. Our implementation may not be perfect, but with a result of **96-97%** we can conclude that it is sufficiently similar to Gao's implementation to use as a benchmark in this article.

The next step in the implementation presented by Gao et al. is the segment labeler. We performed 3 tests on the neural network to see how well it compares to Gao et al. and to see if there are signs of over-fitting. The first test, the Cross-validation test, showed an accuracy of **88%** and the second test, the classification report, showed an accuracy of **83.5%**. In the article of Gao et al. an accuracy of **94.5%** was measured. As you can see there is a difference in accuracy compared to Gao et al. of around **10%**. This difference is mainly due to the fact that we use slightly different and less training data. For example we do not train on house six and thus do not have an air condition in our data set. Instead we use the dishwasher. The results are still accurate enough to be used as a benchmark, but the difference between this implementation and the one presented by Gao et al. should be kept in mind. The third test was the cross-validation on the random over-sampled data used for training the neural net to detect over-fitting. This test resulted in an accuracy of **66%** with a standard deviation of **5**. This indeed is a sign of over-fitting. Because the data as-is is mostly dominated by appliance like the fridge and the microwave, the washer dryer is under represented. Random-oversampling tries to attack this by randomly duplicating examples from the minority class and adding them to the training data set. But because the implementation does not seem to be able to disaggregate the washer dryer well, the overall accuracy is thus decreased. The solution to this problem would be to include more training data.

The third step in the implementation presented by Gao et al. is the multi appliance disaggregation. This step showed an accuracy of **70.46%**. combined with the correctly identified labels of the second step this resulted in an accuracy of **84.5%**. Compared to the **91.2%** from Gao et al. this is quite a difference. However, since the multi appliance disaggregation step is just the segment labeler applied over and over again on each multi appliance label, a big drop in accuracy was expected because the segment labeler showed a difference in accuracy already. With this in mind, this step seems to be sufficient to use as a benchmark.

A step not present in the article by Gao et al. but still very important to mention is the combined implementation. Where the breakpoint identifier, the segment labeler and the multi appliance disaggregation are combined together as they would in real life. This experiment will give us better insight into how the implementation will work as a whole in real life. Since Gao et al. does not present this in their article, we can not compare the results to the original implementation. The point of this test is just to get an indication of how well it will perform overall. The final resulting accuracy was **88.1%**. We will use this measurement to compare the improved combined version to this one.

One of the planned improvements is the inclusion of fake breakpoints in the training data. The goal of that improvement is to increase the accuracy of this test. Because at the end of the day it is the accuracy of the combined implementation that counts.

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%

5.2.2 STUDIO

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, a phone charger, a TV and a 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 multi state disaggregation performs.

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[15] we measured an accuracy of **94%** with a standard deviation of **2** with over-sampling and an accuracy of **98%** with a standard deviation of **2** without over-sampling. Considering the accuracy on REDD **96-97%** the application of the breakpoint identifier on the studio data seems to work well too.

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 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 segment labeler can be seen in 5.5 depicting the performance of the implementation. The confusion matrix can be seen in table 8.5 in the appendix. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[15] we measured an accuracy of **85%** with a standard deviation of **9**. One can see that although

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%

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 data set, a lot of the segments in this data set are multi-appliance segments. This is why with the studio data as training data the implementation seems to be very good at multi state disaggregation. 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 disaggregating 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. using over-sampled data and 10fold cross validation we reach an accuracy of **94%** with a standard deviation of **3**. Compared to REDD over-sampling now seems to increase the accuracy of the implementation. This can be explained by the fact that the studio data set has almost no single appliance data points, except for 2 appliances. Random over-sampling most likely sees for example the laptop as minority and duplicates it, while the others are neglected even more.

multi appliance disaggregation
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 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

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%

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. Putting all of the steps together, the implementation is able to disaggregate **5313** out of the **6510** labels correctly. This is an accuracy of **81.6**%. Not considerably worse than the stand alone implementation.

conclusion
We've just tested the implementation of Gao's et al. on our own studio dataset. The breakpoint identifier seems to compare well against the results of REDD. The test on the studio data seems to get an accuracy of around **94-98%** where REDD seems to get an accuracy of around **97%**. We can conclude that the breakpoint identifier seems to work well on both data sets.

The segment labeler scores and accuracy of **85%** without sampling witch compares well to REDD's **88%** accuracy. The test using random oversampling does show an increase in accuracy. Where the score of **66%** on the 10-fold cross validation on the over-sampled data of REDD clearly indicates over-fitting, the studio data gets a score of **94%**. This is mainly due to the almost complete absence of single appliance labels for most appliances.

The multi appliance disaggregation step performs at an accuracy of **75.3%**. Compared to the accuracy of 70.5% on REDD this is actually an improvement. The studio data mostly consists of multi appliance segments because of the larger number of appliances and a low number of single appliance data points for a lot of appliances. It was thus expected to have a lower accuracy on the multi appliance disaggregation, but it turns out to not perform worse than REDD, but even better. The reason for this can be explained the same way as the accuracy of the segment labeler. A lot of multi appliance states consists of the laptop segment and the fridge segment.

As with the experiment with REDD we tested the combined implementation to see how well all the steps together would perform. In this case it is also not much different. We measured an accuracy of **81.6%** which is slightly worse than the accuracy of **88.1%** on REDD.

overall the implementation seems to work well on studio data. However the results are clouded by the distribution of data points in the data set. The results of the segments labeler mainly reflect how well the implementation is able to perform on the fridge and the laptop, since all the other appliances are hidden away under multi-state segments. However the multi appliance disaggregation step seems to still perform well considering this shortcoming.

In the next two experiments we will try to increase this accuracy with the inclusion of synthetic breakpoints in the training data. and increase the accuracy of the segment labeler with the inclusion of weather data.

5.3 The external features

In this section we will try to answer the first 4 sub-research questions. First of all, how well the combination is able to perform with improvements. We will analyse this using REDD. Second of all, how well it does on modern devices. We will analyse this using our studio data. Thirdly, to what extend the external factors have an effect on the result of the implementation. We will test this with a small weight analyses. Last of all, to what extend the training on fake breakpoints improve the accuracy as a whole. We will test this on both REDD and our studio data by combining the steps of Gao together.

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.

5.3.1 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-

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%

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%

learn[15] we measured an accuracy of **68%** with a standard deviation of **5** using the data with random oversampling and **88%** with a standard deviation of **4** without. 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.

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. Putting all of the steps together, the implementation is able to disaggregate **3412** out of the **4480** labels correctly. This is an accuracy of **76.2%**.

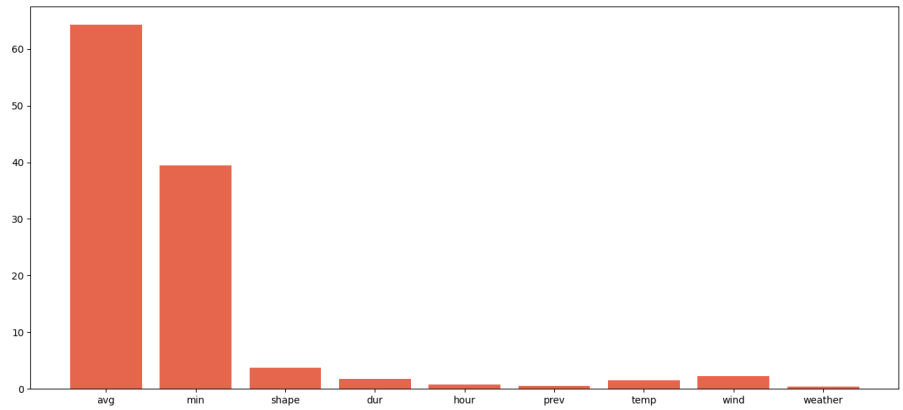


Figure 5.2: analysis of the weights of the improves segment labeler on REDD

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 5.2.

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

conclusion

The first improvement we tested was the inclusion of weather data in the segment labeler. The segment labeler showed an accuracy of **68%** with a standard deviation of **5** using the data with random oversampling and **88%** with a standard deviation of **4** without. Meaning that there are still signs of over-fitting. The implementation without the improvements got an accuracy of **66%** with a standard deviation of **5** using the data with random oversampling and **88%** with a standard deviation of **3** without. This is not considerably different from each other. This would show that the inclusion of weather data in the input features does not increase the accuracy.

The multi appliance dissagregation step did perform a bit better however, with an accuracy of **80.0%** compared to the **70.5%**. This would be a sign that the inclusion of weather data would improve the multi appliance disaggregation step. Since this is just a difference of 20 appliance states, it is hard to say if it is not just a matter of chance.

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%

The improvement of including synthetic breakpoints in the input feature will be testing with the combined implementation. This step did not perform well at all. With an accuracy of **76.2%** compared to the original accuracy of **88.1%**. This shows that training on synthetic breakpoints does not increase accuracy, if not make it worse.

To see if the inclusion of weather data actually has an impact on the resulting output, we measured the average influence of the weights of each input node and multiplied that by the average value of the input. The result was that temperature and wind seem to have a decent effect on the output. The average weight of the weather state seems to be very close to 0, so the fact that it is raining or sunny outside does not seem to matter in this model.

5.3.2 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 in the appendix. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[15] we measured an accuracy of **96%** with a standard deviation of **2** using the data with random oversampling and **88%** with a standard deviation of **11** without. Compared to the original implementation and the improved version of REDD 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.

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%

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. Putting all of the steps together, the implementation is able to disaggregate **5457** out of the **6700** labels correctly. This is an accuracy of **81.5%**.

conclusion
The segment labeler got an accuracy of **88%** with a standard deviation of **11** while the original implementation showed an accuracy of **85%** with a standard deviation of **9**. Just as with REDD no real changes in accuracy are measured.

the multi appliance step had an accuracy of **76.0%**. Compared to the **75.3%** accuracy of the original, no real improvement was measured. This would indicate that the inclusion of weather data in the input feature does not improve the accuracy for the studio data as well.

The combined implementation showed an accuracy of **81.5%**. Compared to the **81.6%** accuracy of the original implementation, we see almost no difference. The inclusion of fake breakpoints does not seem to worsen performance as it does for REDD, but it does not improve it for studio either.

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%

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%

5.4 The generalize ability experiment

Here we will test the generalize ability of the improved implementation and try to answer the last sub-research question. Or in other words, how well the implementation is able to disaggregate appliances from an unknown 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 an AS-NIALM method. Due to a considerable difference in the power consumption of the devices, as can be seen in table 4.1 and table 4.2, 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 **96.5%** is also very good, considering the difference in testing and training appliances.

segment labeler
The segment labeler also does considerably well, having an accuracy of **79.5%**. 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

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%

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 **49.7%**. 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. Putting all of the steps together, the implementation is able to disaggregate **3484** out of the **4533** labels correctly. This is an accuracy of **76.2%**.

5.4.1 CONCLUSION

As we can see from the tests the performance of the individual neural networks are quite good. Take the accuracy of the breakpoint identifier for example, which hovers at **96.5%** . compared to the **94%** to **98%** measured for both data sets separately, this is surprisingly good. This proofs that at leat the breakpoint identifier is able to recognize characteristics very well and thus is able to be applied across data sets.

The segment labeler received an accuracy of **79.5%**. Also not considerably different from **83.1%** and **87.7%** shown by the classification report of REDD and the studio data set respectively. It seems that the segment labeler is able to recognize appliance characteristics very well too .

The multi appliance disaggregation step received an accuracy of **70.0%**, compared to the **80%** and **76%** accuracy, this is not to bad either. We have thus shown that the neural nets on their own can be used as an AS-NIALM method. But what about the complete picture, or in other words, the combination of the neural networks?

The combined implementation measures an accuracy of **76.2%**. Compared to the **76.2%** accuracy on REDD and the **81.5%** accuracy on the studio data this seem to hold up very well to. So it seems that in the end some work does need to be done to make this implementation as a whole an AS-NIALM method, but it does show quite a lot of potential.

Chapter 6

Conclusion

In this article we've presented several experiments and improvements on the multilayer perceptron of Gao et al.[8]. In addition to that we've introduced a new, more present day dataset representing the power consumption of a 1 person studio apartment. In this article we've tried to answer whether or not the the existing implementation can be improved with inclusion of weather related external factors or whether or not the existing implementation can be improved with training on fake breakpoints. We first set out to create a benchmark for ourselves, were we created the implementation as presented in the article by Gao et al. and tested it on both REDD on our own studio data set. To find out if the inclusion of weather data does some good, we've changed the input feature in such a way so that it includes temperature, wind speed and weather state at a specific point in time. We will use the accuracy from the classification report to compare the results. The results are as follows:

For REDD we received an accuracy of **83.1%** for the improved version and an accuracy of **83.5%** for the benchmark. Considering this slight increase in accuracy, we can conclude that the inclusion of temperature, wind and weather state data in the input feature does not improve performance for REDD. The multi appliance dissagregation step did perform a bit better however, with an accuracy of **80.0%** compared to the **70.5%**. But this accuracy is measured of far fewer data points and might be influenced by chance.

For the studio data we received an accuracy of **86.7%** for the improved version and an accuracy of **86.2%** for the benchmark. The multi appliance step had an accuracy of **76.0%** and the benchmark measured an accuracy of **75.3%**. Just as with REDD the accuracy is fairly similar, so we can conclude that the inclusion of temperature, wind and weather state data in the input feature also does not improve performance for the studio data.

Because of the small difference in accuracy, we undertook a small experiment on the weights of the input nodes of the segment labeler. This experiment did indeed show that at least the temperature and wind speed have some influence on the outcome of the neural network. So the added input features are not just simply ignored.

This small difference could be due to the absence of relation between weather and

appliance power consumption, but it might also be because of the lack of difference in the weather data. Both the measurements of REDD and the studio data took place during the summer, especially for studio is barely rained during the measuring period.

The improvement of training on false breakpoints was mainly meant to improve the accuracy overall, where the segment labeler would get the input directly from the breakpoint identifier. To find out if the training on false breakpoints improved performance we linked the processes together and compared the correct labels at each point in time, to the predicted labels at each point in time. We will use the accuracy from the classification report to compare the results. The results are as follows:

For REDD we received an accuracy of **76.2%** for the improved version and an accuracy of **88.1%** for the benchmark. For studio we received an accuracy of **81.5%** for the improved version and an accuracy of **81.6%** for the benchmark. Although REDD shows an considerable drop in accuracy, the studio data seems to show no difference. We can thus conclude that training on fake breakpoints does not increase accuracy. It might even make it worse.

Last of all, we wanted to know how well the improved version is able to generalize across data sets. We tested this by training on REDD with the selection certain devices present in both datasets, and testing on our own studio data. The results are as follows:

the breakpoint identifier showed an accuracy of **96.5%** . compared to the **94%** to **98%** measured for both data sets separately, this is surprisingly good. The segment labeler received an accuracy of **79.5%**. Also not considerably different from **83.1%** and **87.7%** shown by the classification report of REDD and the studio data set respectively. The multi appliance disaggregation step received an accuracy of **70.0%**, compared to the **80%** and **76%** accuracy, this is not to bad either. The combined implementation measures an accuracy of **76.2%**. Compared to the **76.2%** accuracy on REDD and the **81.5%** accuracy on the studio data there is no discernible difference. It seems that the improved implementation for Gao, and the original too for that fact, will work fine as an AS-NIALM implementation.

Regarding the gathered studio data, the power consumption of 10 devices have successfully been gathered for 3 weeks. In addition to that we research if it is possible to use the summation of power consumption of the individual appliances as aggregate data. As our experiment shows, we only have a slight spike in power difference if a large amount of power is consumed due to the resistance in the extension cord. since this counts for only 0.5% we've elected to ignore it.

To answer the stated research questions, we have concluded that external factors like weather data do not increase the accuracy of this implementation. Training on fake breakpoints seem to does not seem to increase accuracy either, if not make it worse. What we were able to show however is the ability for the Gao's multilayer perceptron and the improved adaptation for it to be used as an AS-NIALM method. Compared to other AS-NIALM implementations like the auto-encoder of Kelly et al.[7], which utilizes an entire different strategy then Gao et al., the performances are very similar.

Chapter 7

Future improvements

As we previously concluded, the addition of weather data in the input feature and the training on fake breakpoints did not seem to improve accuracy. There are however a few things that might be good to be looked into in future research regarding the implementation. First of all, in this research we implemented a interpretation of Gao's multilayer percentron, not an exact copy, since we had to make it from scratch. If it is possible to access the original implementation, which was measured to have a slightly higher performance overall, the accuracy of the improved version might increase as well. Because of that it could give a clearer image on the usefulness of weather data in the input feature.

Secondly, we chose our devices within REDD to be as close to Gao's research as possible. But REDD contains more devices than was used in our research. For example, the lights were not included, since the test on the implementation in the article of Gao et al. did not include them as well, and the air-conditioner was not used since only one house contained data for it, which did not overlap with our temperature data. In the future this implementation could be tested on more appliances in REDD.

Thirdly, we had to automate the parsing of the true breakpoints from the individual appliance data since Gao's manual approach is imprecise and unfeasible for large data sets. We used a naive method for this, where if a power consumption goes below or above a certain power threshold, the point is marked as a breakpoint. This has the downside of generating a lot of breakpoints for pulsating devices. In the future a more advanced algorithm needs to be developed to identify true breakpoints by taking the existence of Finite State Machines[4] into account, where an appliance can have more than just an on and off state.

Lastly, we've only looked at the breakpoint identifier and the segment labeler of Gao to analyse the increase in performance with inclusion of weather data in the input feature, but this could also be tested on other breakpoint or state detection algorithms, like the one by Dash et al.[19]. Gao's implementation also has some major shortcomings in the multi appliance segment disaggregation step, since it is not even able to disaggregate a segment where two appliances are turned on at once. To get a more general overview of temperature, wind and weather state influence on power consumption or energy disaggregation, more models need to be researched.

The same can be said about the inclusion of fake breakpoints in the training data. Another model might show more promising results for this inclusion than what the model presented in this article showed.

In this article we presented our own gathered data from a one person studio apartment. We used this data for our comparisons next to REDD. This data consisted of the power consumption of 10 devices and was measured for 3 weeks. The intention of the data set is to contain the most frequently present modern devices in a household. We believe this criteria has been met, but a lot of improvements could be made on the data set itself. For instance, take a look at the data set description in table 4.2. Here it can be seen that the distribution of the data is very uneven. The laptop, PlayStation 4 and fridge occupy a large portion of the overall data points, while the desk lamp, couch lamp and water heater only occur a fraction of the time. This is of course due to the fact that some appliances are used more often than others, but the resulting problem is that these appliances are completely underrepresented in the training and testing data. So the model won't be able to disaggregate them no mater how hard it learns. The measured accuracy will therefore mostly reflect how well the model is able to disaggregate the more frequently used devices.

We tried to solve this using random over-sampling, but the distribution was still not even enough. In the future, more data points for the less frequently used devices are needed. This can be done by simply measuring for a longer period, where a year or more would be enough to create enough data points for the model to accurately train the less frequently used appliances on.

Another argument for a longer period of data gathering is the use of weather data. Both REDD and the studio data were measured at around the same time during the summer. We saw hardly any rain or cloudy weather during the measurements of the studio data and the temperature was always quite warm. This can be seen in table 4.3. A year round measurement is needed to get a more accurate representation of the influence of temperature and weather state on the power consumption of devices or at least more difference in temperature and weather state. Freezing versus very hot temperatures could give a clearer image of the influence temperature has on the fridge and rainy weather versus sunny weather a clearer image of the influence of light on the power consumption of lamps.

Apart from more data, another solution to the shortage of data points might be the potential improvement of training on synthetic data (not only synthetic breakpoints, but synthetic segments too). As stated in the article by Kelly et al.[7]: "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.". We have shown that training on synthetic breakpoints does not seem to better performance, but creating synthetic segments of combinations of less frequently used devices might improve the accuracy of those devices. Research does need to be done in the case of preserving usage patterns within the synthetic data. For example lights go on when it is dark and if the TV is turned on, the gaming console or dvd player is also most likely to be turned on.

Another thing to mention about the data, is the extraordinary period of time the measurements took place. The data was measured during the beginning of the first

wave of the corona virus, so most of the power consumption actually reflect a one person studio apartment in quarantine. Due to this quarantine a substantial amount of time of the resident was spend at home, meaning that the energy consumption and the appliance used are considerably more than normal. In the future a more accurate representation of a household could be created where the resident isn't obligated to stay home all day.

The first thing that is checked before we try to answer the research questions is whether the aggregate data can be taken from the summation of the power consumption of the individual appliances. The answer was yes due to only a 0.5% difference in summed and true total power consumption. This was concluded to be caused by the resistance in the extension cord. It should be noted however that in real application the resistance might be a lot higher, since we're not talking about a short extension cord anymore, but a whole house filled with wires. Further research needs to be done in order to see if the spike in difference of measured power consumption becomes significant when applied to an electricity grid within a household.

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

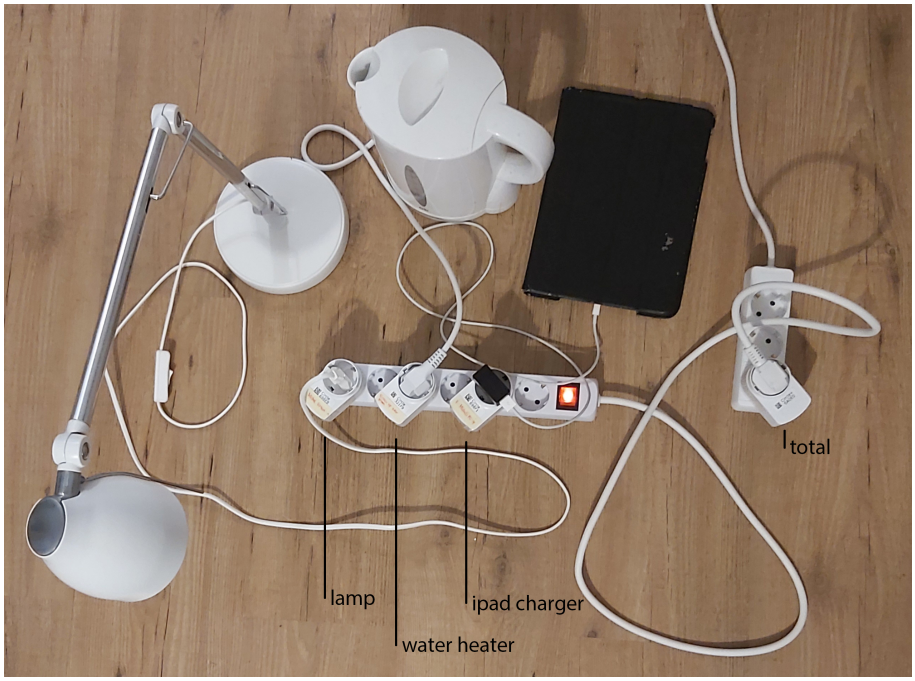


Figure 8.1: setup sum of individual consumption experiment

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] 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.

[11] Hyungsul Kim, Manish Marwah, Martin Arlitt, Geoff Lyon, and Jiawei Han.

Unsupervised disaggregation of low frequency power measurements. volume 11, pages 747–758, 04 2011.

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

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] S. Dash, K. Gandhi, and R. Sodhi. An automatic state detection algorithm for non-intrusive load monitoring. In *2019 IEEE 16th India Council International Conference (INDICON)*, pages 1–4, 2019.