university of groningen

faculty of science and engineering

Detection of used devices from energy consumption patterns

Bachelor thesis

Author:
Job Heersink

Supervisors:
Viktoriya Degeler
Alexander Lazovik

University of Groningen
The Netherlands
May 23, 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 in combination with other energy conservation software 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 neural networks. We will explore an existing method and analyse how well it does on the most frequently present devices in a household and extend it to take weather data into account. In addition to that we will explore the opportunity of generalizing this implementation across houses by letting the neural network learn on a set of household consumption data and testing it on another household.

Contents

1 Introduction **2**

2 Related work **3**
2.1 Hart . . . 3
2.2 Kelly . . . 3
2.3 Luca . . . 3
2.4 REDD and python tools . . . 4
2.5 small notes . . . 4

3 approach **6**
3.1 measurements . . . 6
3.2 Problem Formulation . . . 8
3.3 Research Questions . . . 8
3.4 Methodology . . . 9
3.5 Breakpoint identifier . . . 9
3.6 segment labeler . . . 10
3.7 Multi appliance dissagregation . . . 10
3.8 training the neural network . . . 10

4 Experiment 1: REDD **11**
4.1 Breakpoint identifier . . . 11
4.2 segment labeler . . . 11
4.3 multiappliance labeler . . . 12
4.4 conclusion . . . 12

5 Experiment 2: own data set **12**
5.1 conclusion . . . 13

6 Experiment 3: external features **13**
6.1 conclusion . . . 13

7 Experiment 4: generalizeability **13**
7.1 conclusion . . . 13

8 Conclusion **14**

9 Future improvements **15**

10 Contributors **16**

11 references **17**

1 Introduction

NOTE THE IMPORTANCE OF SMALL DEVICES
Due to the increasing use of smart grids it is now possible to monitor, control and manage the energy demands on a real time usage basis to reduce the overall consumption[1]. According to a book on the smart grid and research opportunities[2], the demand for a control-able and measure-able environment is substantial and growing.

In this paper we look at several existing manual setup nonintrusive appliance load monitoring (MS-NIALM)[3] techniques. Looking for aspects that make implementation more efficient and what shortcomings it might have. Based on that we will introduce an implementation combining, improving or introducing new components creating a more efficient algorithm. In addition to that, the possibility of extending this method to an automatic setup nonintrusive appliance load monitoring (AS-NIALM)[3] will be researched and implemented to try remove the need of knowing the power consumption of each device beforehand.

We will be using the eference Energy Disaggregation Data Set (REDD)[4] to train and test our implementation and compare it with others. Next to that we will be using the plugwise measuring devices from v. degeler[5] to gather our own power consumption data of 10 devices from a 1 person studio apartment in Groningen. This data will be used to train and evaluate our NIALM implementation and will give us insight in the efficiency of our NIALM implementation regarding more present day devices one could find in a household, like laptops, smartphone and ipad chargers and gaming consoles, which are not present in REDD.
This research will provide the ability to recognize the states of devices in a room, without having to know what the exact devices in a room are.

To create this AS-NALM technique we will dive in the neural network area of Computing science and use several techniques relevant to statistics. In addition to that, this implementation will be tested against several other MS-NALM techniques using a dataset accumulated from a 1 person studio apartment. SAY THAT WE ARE USING LOW FREquency TALK ABOUT THE STRUCTURE OF THE ARTICLE

2 Related work

2.1 Hart

One of the pioneers in this research area was George Hart in the late 1980s with his article on non-intrusive appliance load monitoring (NIALM)[3]. 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[6] 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 opr 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%.

2.2 Kelly

One of the first succesfull neural network AS-NIALM implementation was the Autoencoder of Kelly et al[7]. In this paper the performance of 3 neural network architectures are tested and compared against eachother as well as the well known combinatorial optimisation or factorial hidden Markov models.
The most notable approach the autor has taken is that, instead of creating one neural network dissagregating the aggregate signal, it creates a neural network for each appliance. The network should first train on a mix of real data and syntactically created data to make it a more generalize implementation and could then be applied to any house to measure the power consumption of a similar device. Tested on all aplliances the Auto Encoder got an f1 score of .53, precicion of .47 recall of .94 and an accuracy of .93. In addition to that the neural networks use a naive form of syntetic data generation. It does not account for relations between devices or time. For example light might turn on more often when it gets dark around a certain time and if the TV is turned on, chances are that the gaming console will be turned on as well a few seconds later. Another issue is that the train data only contain 5 big devices, but lack most common appliances in households or offices like lights and laptops, which are harder to dissagregate. On the plus side, the Auto Encoder does not give the state of the device, but the actual consumption of the device. That would mean that for multi state appliances like computers it could detect more than just the on and off state. It shows a lot of potential, but needs to be further improved before it can be applied in houses or offices. In this paper we will test our implementation on the extend it can be used as a AS-NIALM algorithm, by training it on syntetic and real data like Kelly, but with improved syntetic data generation. And test how well the implementation is able to generalize.

2.3 Luca

One of the most resent AS-NIALM implementations is the Temporal Pooling NILM (TP-NILM) architecture. it is an adaptation and simplification of the network called PSPNet (Pyramid Scene Parsing Network). It is one of the more succesfull AS-NIALM implementations with an average Precicion recall accuracy and f1 accross all appliances of: 0.93, 0.92, 0.96 and 0.92 respectivly.
This implementation has been tested on only 3 devices: A Fridge, Dishwasher and washing machine and that has a reason. Large devices with only a very temporary on spike like microwaves and ovens are very hard to detect by this implementation, let alone small power consuming devices like lights and laptops. In addition to that the implementation might be able to generalize well, but has one neural network for all aplliances, unlike kelly CITE. That means that this implementation needs to be trained on the exact devices available

in the household. This implementation is still AS-NIALM however, since it can train on data like REDD. In our implementation we will use a tactic similar to this one. A single neural network to dissagregate into appliance states which is able to generalize over different houses.

IS THIS ACTUALLY THE CASE???

2.4 REDD and python tools

Disaggregation algorithms need to learn a model of how appliances consume energy from existing data. Such algorithms generally require appliance-level data from either the building which will be disaggregated (sometimes referred to as supervised) or buildings other than the one which will be disaggregated (sometimes referred to as unsupervised). A dataset containing both supervised and unsupervised data is the The Reference Energy Disaggregation Data Set (REDD)[4].
REDD is a freely available data set containing detailed power usage information from six residential houses, with the aim of furthering research on energy disaggregation. An example of the data in REDD can be seen in figure 1, excluding unlabeled data like sockets and data with the appliance unknown. REDD is a widely used dataset for energy disaggregation research and makes comparing energy disaggregation algorithms easier. In addition to that REDD provides a general geographic location, namely Massachusetts, US, and timestamps of where and when the measurements where taken. This allows for the inclusion of temperature, wind and weather state data in training and testing NIALM implementations. We will be using this dataset and our own gathered data to train and test the method presented in this paper. Using REDD will make it easier to compare other NIALM implementations with our implementation.

The non-intrusive load monitoring toolkit (nilmtk)[8] 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 usefull. We will be using this toolkit for loading and parsing REDD and plotting certain fractions of the REDD data.

for the creation of our Multi-layer Perception classifier we use the scikit-learn[9] package, so that the architecture of our implementation is similar to that of Gao et al.[10]. 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[11] 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.

Devices in REDD	Max power	on power threshold	min on duration	min off duration
washing machine	2500	20	1800	160
dish washer	2500	10	1800	1800
microwave	3000	200	12	30
fridge	300	50	60	12

2.5 small notes

Gao had an overall accuracy of 91.2 procent on REDD then AS-NIALM will be tested.
talk about what we use
measurments are different because ofcorona WHY WE CHOOSE TO USE THIS IMPLEMENTATION!!! With for example a table
when the choice has been augmented, move on to the explanation.
maybe instead of the mean take the min? spikes are usually positive.
maybe transform the time to some scale? partial_fit() iterates only once over the neural network mains in data are fucked. so need to sum appliances we choose to have a timeperiod between readings of 3 seconds just like REDD database. 3 seconds generates to much Nan. While 10 seconds seems to hold the error at a minimum. the start time of the appliance being used is used for training the neural network we choose to remove the sockets and unkown from the meters in REDD since they do not neccecarily represent appliances.

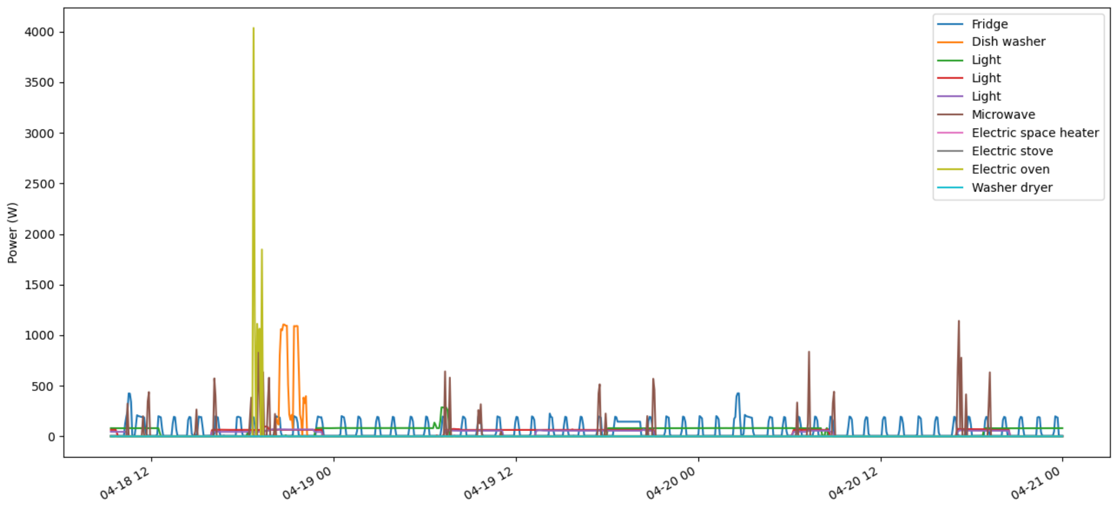


Figure 1: example data from the 1 house of REDD. Not included are unlabeled appliances like sockets and unknowns

To prove that for measuring the data and then summing it up. We hooked a lamp onto a plugwise device in an extention cored which on itself is also connected to a plugwise device. While the plugwise device measuring the light consumption was mostly off, the plugwise device measuring the extentioncord had an occasional small spike of not more than 2.5W, but more plugwise devices seem to show such an occasional spike in their off state. When the light turns on the plugwise device seems to measure the power accuratly, but still shows signs of the occasional power spike. Taking an error margin of 2.5 into account makes it possible to measure devices and dissagregate on their sum. the occasional spike might also be the plugwise devices requiring power. When plugging in two plugwise devices into the extiontion cord, the occasional spike sometimes shows a spike of 4-5W. This shows the the spike is most likely caused by the consumption of the plugwise devices themselves. [INCLUDE A TABLE WITH THIS DATA] We only pick the important things out of redd,not sockets. We choose to sum up the data of REDD because the mains contain a lot of NaN values and we choose to not use the sockets and unkown appliance data, so the main contains power consumption that we do not dissagregate. We showed before that almost no extra power is conumed of the grid in a house [expirement with extentioncord] explain what the labels mean gao uses 1 minute sampeling and addition of active appliances as well We used https://www.timeanddate.com to gather information on the temperature and weather of the houses in REDD maybe use cross validation in approach? why we use low_frequency over high_frequency without the electric oven and electric stove Gao gets an accuracy of 97 procent we propose to introduce a as-nialm technique that dissagregates an energy signal using a list of appliances with the power conumption in watt and the kilowatt hour. This can be read of the label or if no information is specified general data can be used. such as http://energyusecalculator.com. which used a For the majority of electronics shown on this website, we have relied on the P3 International line of products and in particular the P3 P4400 Kill A Watt Electricity Usage Monitor. The use of ‘small’ neural nets on NILM dates back at least to Roos et al. 1994 [10] (neural net article) this is a time series problem. maybe use foward filling to resolve NaN ”We found that synthetic data acts as a regulariser. In other words, training on a mix of synthetic and real aggregate data rather than just real data appears to improve the net’s ability to generalise to unseen houses. For validation and testing we use only real data (not synthetic).” cite neural network article we modify the first step of gao, to fit more general. We saw for example on our data that 0 is often followed by a multi appliance segment, due to the fridge propose to change the third segment, since it will be IMPOSSIBLE to dissagregate a timestep that has multiple appliances. We will take the tactick from kelly where we select appliance on states.

3 APPROACH

3.1 MEASUREMENTS

Devices	average off consumption	average on consumption	on time
washing machine	6	87837	787
phone charger	7	78	5415
desk lamp	545	778	7507
lamp	545	18744	7560
tv	88	788	6344
Fridge	88	788	6344
ipad charger	88	788	6344
water heater	88	788	6344
microwave/oven	88	788	6344

Apart from the REDD dataset we will be using our own dataset containing the power consumption data of 10 devices from a 1 person studio apartment in Groningen with a duration of 3 weeks. This data has been made with the goal of included the most frequenct present devices in a household. This means that, apart from big appliances like washing machines, microwaves and water heaters, this dataset also contains relatively low power consuming devices like lights and phone chargers. For the total list of devices and their average consumption see INCLUDE FIGURE
This data was gathered using a total of 10 plugwise[1] devices and a USB stick called a 'Stick' as shown in figure 3. The set of these Plugwise devices communicated over a wireless ZigBee mesh network around one coordinator, shown in figure 3 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[5] The python-plugwise library[2] was used to create a small script to communicate with the stick and request the data from the plugwise devices. This script gathered power consumption data for each one of the 10 devices for 3 weeks. The aggregate data was calculated by summing up the total appliance consumption and adding a small error value. To evaluate whether or not this method would produce correct aggregate data we did a small experiment:
3 devices were added to a extension cord witch in turn was connected to a plug-wise device. So that plugwise device was gathering the power consumption of the 3 devices. One first sight it is visible that an occasional spike occurs of 2.1-2.2 W, being different from one plugwise to another but stays mostly constant for an individual plugwise device. For instance, if the appliance is turned off, plugwise device 1 may have the occasional spike of 2.14024W exactly, while the other has a spike of 2.1125W exactly. Another thing that should be noted is that the plugwise devices consume power as well, about 1.1W per piece according to V. Deleger et al[5], so their power consumption could show up on the aggregate data, but not on the individual data. The real aggregate data is then plotted against the sum of the consumption of the individual devices. show in figure: INCLUDE FIGURE.

mesh communicates with the 'Stick' which in turn allows for gathering data on a pc where the Stick is plugged in. and further explains the details of the implementation.

Using this implementation the power consumption of a set of 10 devices, which consisted of a microwave oven, a TV, a PlayStation 4, two lamps, an Alienware R17 laptop, a Samsung galaxy A70 phone charger, an iPad charger, a water heater and a washing machine, were individually analysed. Note that we chose to 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[4]. 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[10] is much more difficult to dissagregate. Our own gathered data will thus

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

provide a better overview of the efficiency of our algorithm for more difficult devices.

We use overfitting on the data, so it is not able to generalise well

why our implementation is much harder to dissagregate

3.2 Problem Formulation

The problem of dissagregating a aggregate energy consumption signal can be formulated as follows:

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

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 the amount of appliances. As we have shown in EXPERIMENT $y(t)$ can be represented as the sum of the power consumption of all devices at a certain point in time including a small error value $e(t)$.

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 define the appliance activation $x_n(t)$ as follows, where the value 1 represents the on state and 0 the off state:

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

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 dataset, still consume power while turned off. These cases are also referred to as vampire appliances [12]. $e_n(t)$ accounts for the occasional spike and acts as an error margin.

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 aggragate signal $y(t)$, a set of appliance consumption/states is returned.

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

Note that our implementation produces a slightly modified version of M_x, namely F: where given a set of characteristics S, would return a set of activate labels of appliances N.

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

3.3 Research Questions

We propose to create a method for energy disaggregation capable of recognizing states of known devices from overall power consumption. We would like to investigate to what proportion external factors like temperature, wind and weather data have an effect on appliance dissagregation. In addition to that we would like to test our implementation on generalizability accross housings.

explain why temperature would be a viable research

this will lead to the following research questions:
- how well does a multilayer perceptron classifier work on standard household appliances. - to what extend do external factors have effect on the accuracy of energy dissagregation.

If a satisfying answer is found to a number of these research questions we will have the following results:
- a more accurate algorithm, written and tested in python, to produce more reliable results for appliance dissagregation
- an implementation able to recognize unknown devices using a general dataset

3.4 Methodology

Gao calls the breakpoint identifier the segmentation classifier

The implementation we will be testing and extending is based on Gao's multilayer perceptron classifier[10]. We choose this implementation since it offers opportunity for change in improvements in comparison with other implementations. This is because the multilayer perceptron classifier doesn't take the energy signal of the appliance as its input like Kelly et al.[7] and Luca et al.[13], but just a few characteristics of it as well as time, shape and last active appliance, making it possible to add more paramaters like temperature, wind and other possible factors. This in turn makes it easier for the neural network to learn relations between devices, where Kelly would have a lot of trouble to do so since each neural network only dissagregates one appliance. This implementation uses two neural network classifiers, the segmentation classifier and the segment labeling classifier, to dissagregate an energy signal. First the segmentation classifier breaks the signal up into segments, where a segment is a set of energy consumption where no appliances are turned on or off. This is done by identifying breakpoints, a point on the signal when a device has been turned on or off. Second 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.
For the first step we use a classifier that can be used to label all points in power series as either breakpoints or non-breakpoints almost exactly like gao's, namely a multilayer perceptron classifier trained on two features, the mean power and power difference (delta), at each data point. Gao's breakpoint classifier seems to be working near perfectly for the chosen appliances in it's article. For the second step we try out a range of possible inputs for the segment labeling classifier and choose the most implementation that proofs to be more efficient.

3.5 Breakpoint identifier

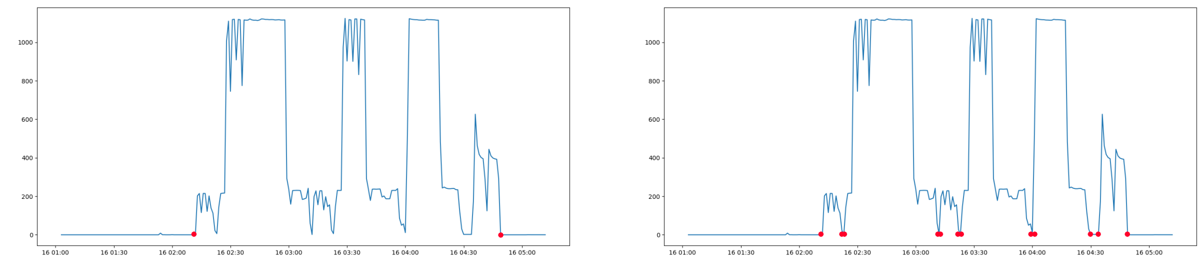
The first multilayer perceptron classifier in Gao's implementation is the breakpoint identifier. This neural network should be able to identify weather an appliance state has change on a point on the signal, in other words if a breakpoint that specific point is a breakpoint. Using these breakpoints we can create segments, where each segment represents different combinations of appliances. For each point in the dataset the classifier will output either a 1, saying that this point is a breakpoint, or a 0, not a breakpoint. The breakpoint identifier is trained based on the two features defined in , that is, the mean power and power difference (delta), at each data point.

create table

These features are extracted from the aggregated energy signal and the true breakpoints that are trained on are extracted from the separate appliance consumption signals. Goa identified these true breakpoints manually, this is however unfeasible for large quantities of appliances and almost impossible to reproduce. Since we will test this implementation on 10 appliances, 6 more than was tested in the original paper, we decide to identify the breakpoints using a naive algorithm. This algorithm would detect when a appliance was turned on for example by checking if the current value is above a certain power threshold and the previous value is below the same power threshold . For most appliances this would produce correct results, since spikes are averaged out by the sample rate of 1 HZ and appliances are only stated as on if they actually consume power, but for pullsating appliances like the dishwasher, which consume power in bursts, the definition of correct becomes a bit vague. Take for example a slice of the data from the dishwasher appliance at figure 2. If one were to manually identify breakpoints, one would probably go for Gao's, since it does signify one big on state, however according to the definition of a breakpoint, or a change in appliance states, at figure 2b, is not wrong. One could see it that the dish washer is turned off for a few minutes, before it consumes power again. Testing both variations proved that the breakpoint identifier had a hard time dissagregating them, mainly due to the increased amount of breakpoints vs non-breakpoints in contrast to the orginal implementation, but proved to increase accuracy for the segment labeler. That is why, apart from the ease of use for larger appliances we chose to implement the naive method shown in figure 2b.

include the table with power threshold data

did i already say this?



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

Figure 2: true breakpoint definitions examples for a part of dish washer

(a) the stick (b) 2 of the 10 plugwise devices. (1 Circle+, 1 Circle)

Figure 3: measuring devices

3.6 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 . We create the training and testing input by extracting this information from the signal and create the target output by getting the state of each device from the same time period as the segment.

include table

3.7 MULTI APPLIANCE DISSAGREGATION

At this step we try to dissagregate the mutli appliance segments. We do this by first subtracting neighbouring single appliance segments as follows: If the nearest left neighbour that is a single appliance is the same appliance as the nearest neighbour on the right, we will presume the multiappliance segment is lifted by the consumtion of that appliance and subtract its average consumption from the multiapplaince consumption. If the two neighbours are not the same, we will presume the multiapplaince segment is lifted by both, and subtract both consumption from the signal. The remaining consumption will be put through the segment labeler again, and the same process will repeat untill no multistate appliances are left.

3.8 TRAINING THE NEURAL NETWORK

The classifiers are trained and created to match the specifications in Gao et al.[10]. We use 10-fold cross validation to train the neural network and measure the accuracy and we apply random oversampling [11] 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[9]. The specifications are listed in the table below . 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.

reference table

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

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

4 Experiment 1: REDD

First we test our implementation on performance and compare it to goa's mutlilayer perceptron. We test on the same dataset as Gao tests, namely REDD. We will be testing our implementation on 4 appliances: a fridge, a microwave, a washer dryer and a dish washer. Note that Gao tests on the following 4 appliances: a fridge, a microwave, a washer dryer and a air conditoner, but due to the fact that house 6 and 5 containt corrupted data[10] and measures power on dates where we do not have weather data on, which will be important for expirement 2, we choose not to include those. This means we do not have data on the air-conditioning. Instead, we decided to dissagregate the dish washer.
Just like Gao, for each appliance, we down-sample the power signal from 1 Hz to one measurement per minute. Finally, we obtain our experimental data by aggregating and adding up the power signals from all chosen appliances in the same house.
The results should be as similar as possible, since we used the same architecture. But the testing is done of slightly different data. Since the washer dryer consumes power in pulses, it could be harder to disaggregate. In addition to that Gao manually identifies breakpoints, while we let an algorithm detect the breakpoints. Our training/ testing data might thus not be completely the same.

4.1 Breakpoint identifier

The first classifier we test is the breakpoint identifier. The classification report of the breakpoint identifier can be seen in depicting the performance of the implementation. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[9] we measured an accuracy of **0.98** with a standard deviation of **0.01**. Gao doesn't specifically note the accuracy of the breakpoint identifier, but states that the breakpoint identifier "performs nearly perfectly in identifying breakpoints and non-breakpoints". Keeping in mind that Gao has a lower ratio of breakpoint vs non-breakpoint than our implementation due to the inclusion of the dishwasher and the fact that we do not manually label the breakpoints, we can conclude that this implementation is sufficiently similar to gao's breakpoint identifier. In the confusion matrix of 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.

include table

include table

confusion matrix	non-breakpoint	breakpoint
non-breakpoint	10611	377
breakpoint	36	496

classification report	precision	recall	f1-score	support
non-breakpoint	1	0.97	0.98	10988
breakpoint	0.57	0.93	0.71	532

accuracy: 0.96

4.2 segment labeler

The second classifier, the segment labeler, is up next. The classification report of the segmnet labeler can be seen in depicting the performance of the implementation. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[9] we measured an accuracy of **0.90** with a standard deviation of **0.03**. Goa's measured an accuracy of **0.945**. This difference in accuracy is mainly due to the absence of

include table

house 6 in the training data, resulting in less datapoints for fridge. The washer dryer also seems to perform bad in this implementation. . The confusion matrix of the segment labeler can be seen in . Considering only the segments labeled as single appliance or empty by the segment labeler, **439** out of **492** appliance or empty instances were labelled correctly, where **35** where mislabeled and **18** labels not labeled at all (due to the segment being labeled as a single appliance while being a multi appliance segment).

explain why

include table

confusion matrix	empty	fridge	microwave	washer dryer	dishwasher	multi
empty	211	2	0	0	0	0
fridge	1	159	1	1	19	6
microwave	0	0	37	0	1	6
washer dryer	0	0	0	2	3	6
dishwasher	0	3	2	0	14	3
multi	0	0	9	1	8	37

classification report	precision	recall	f1-score	support
empty	1	0.99	0.99	213
fridge	0.97	0.85	0.91	187
microwave	0.76	0.84	0.80	44
washer dryer	0.5	0.18	0.27	11
dishwasher	0.31	0.64	0.42	22
multi	0.64	0.67	0.62	55

accuracy: 0.86

4.3 multiappliance labeler

The last step of Gao's multilayer perceptron is dissagregating the multiappliance segments further. We test this step on the **55** multi appliance segments from the testing data. From the total of **112** single appliance labels within the multi appliance segments **73** were correctly. Combing these results by adding up the total and correct predicted labels like Gao we end up with **512** out of **604** correct. It got an accuracy of **84%** Compared to Gao's **91.2%**.

4.4 conclusion

With an accuracy of **0.98** and **0.90** for the breakpoint identifier and the segment labeler respectivly and an overall accuracy of **84%**, we conclude that the implementation is similar and efficient enough to use as a benchmark against our proposed improvements. We do regonize that, although we use the same dataset, we train and test only on 4 houses, instead of 6. We may also split the point between training and testing data differently then Gao. Resulting in less points to test on then Gao.

5 Experiment 2: own data set

Here we propose to test our implementation of the multilayer perceptron classifier of Gao against our own dataset and compare it with the results of experiment 1. Our dataset is created with the intention of accuractly simulating a 1 person studio apartement in present day. Where REDD only includes household appliances like fridge, washing machine and a microwave and some general appliances like lights, other common appliances in households are missing like a laptop, phone charger, tv or gaming console. Our implementation does include these applainces. 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 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-apliance segments will incraese substantially. Thus with this expirement we can also see how well the multistate dissagregation performs.

5.1 conclusion

6 Experiment 3: external features

6.1 conclusion

7 Experiment 4: generalizeability

7.1 conclusion

8 Conclusion

9 Future improvements

10 Contributors

Special thanks to Azka Pratama and Viktoriya Degeler for lending out raspberri pi and the measuring equipment.

Add guy from email

11 REFERENCES

References

[1] D. Chowdhury M. Hasan and Z. R. Khan. Non-intrusive load monitoring using current shapelets. *Applied Sciences*, 9(24), 2019.

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

[4] 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.

[5] 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.

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

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