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 28, 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 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)[1]. 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[2] 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[3]. 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

REDD	max	mean on	mean off	mean on time	activations	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 1: characteristics off the data in house 1 of REDD

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)[5] 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[6] package, so that the architecture of our implementation is similar to that of Gao et al.[7]. 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[8] 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.

2.5 small notes

Gao had an overall accuracy of 91.2 procent on REDD then AS-NIALM will be tested. our missing data elements are removed by foward filling training on all houses

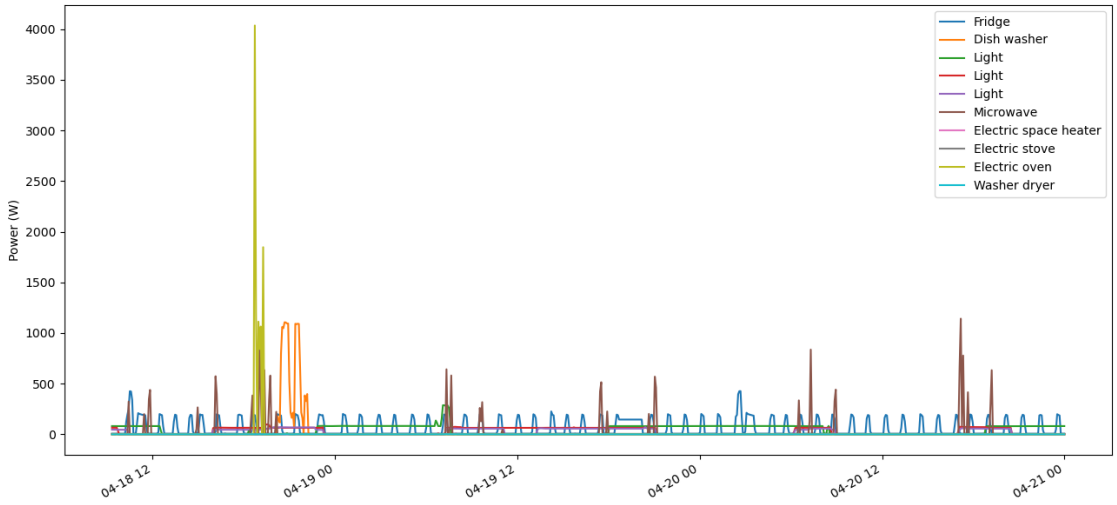


Figure 1: example data from the 1 house of REDD. Not included are unlabeled appliances like sockets and unknowns

studio data	max	mean on	mean off	mean on time	activations	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 2: characteristics of appliance power consumption

3 MEASUREMENTS

3.1 APPROACH

are the namings stick and plugwise all capitals?

explain the columns of the table

Apart from the REDD dataset we will be using our own dataset for training and testing our implementation. This dataset contains the power consumption data of 10 devices from a 1 person studio apartment in Groningen. We measured the power consumption data every 10 seconds for 3 weeks. The data was gathered using a total of 10 Plugwise[1] devices and a USB stick called a 'Stick' as shown in figure 15. The set of these Plugwise devices communicated over a wireless ZigBee mesh network around one coordinator, shown in figure 15 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[9]
The python-plugwise library[2] was used to create a small script to communicate with the stick and request the data from the plugwise devices. This script was run on a dedicated laptop for 3 weeks. A plot of the

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

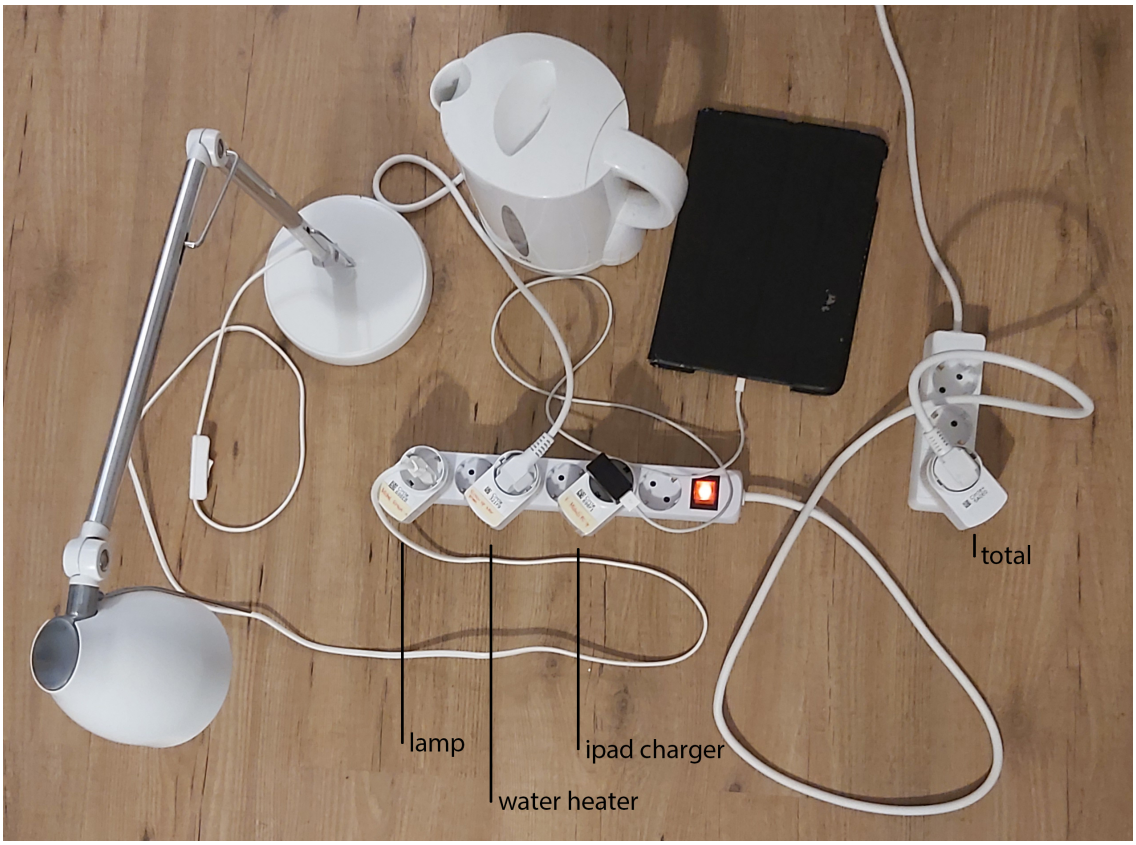


Figure 2: setup sum of individual consumption experiment

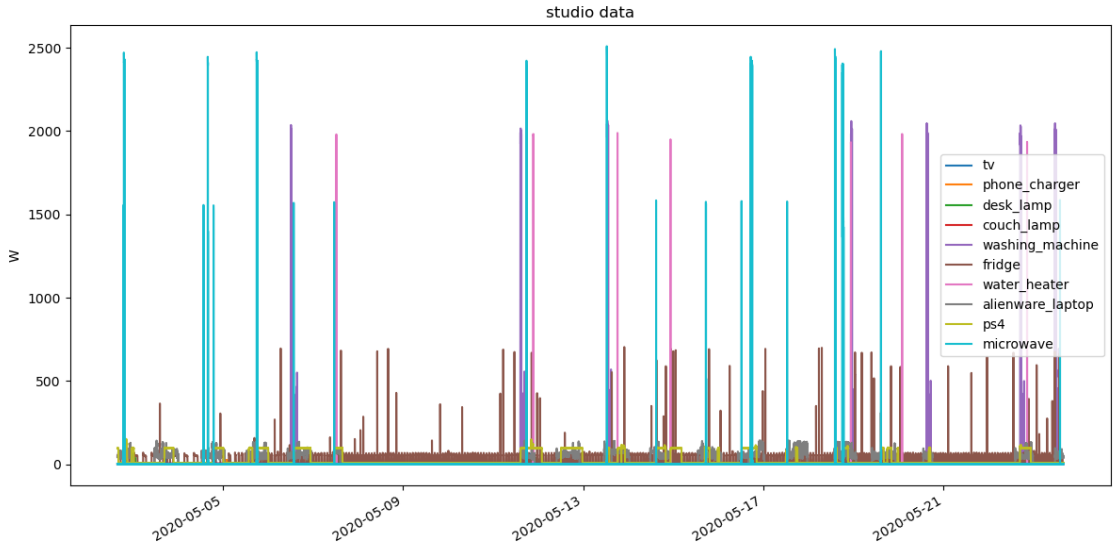


Figure 3: plot of the studio data including all individual appliances

result of the data gathering can be seen in figure 3

3.2 DATASET DESCRIPTION

The choice of included appliances are based on the most frequenctly present devices one could find 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. 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[7] is much more difficult to dissagregate. Our own gathered data will thus provide a better overview of the efficiency of our algorithm for more difficult devices.
For an overview of all appliances and a their consumption characteristics see table 2.
We gathered the consumption data for the following devices:

A TV
or a Philips 40PFL4358H/12 3D TV to be precise, which can be seen in figure 4. 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 2 this predicting is pretty accurate. In this dataset the tv was used mostly only during the day, that includes mornings and evenings and is on for around an hour. Regarding relations to other devices, when the TV is turned on, the ps4 almost always follows, since the ps4 is used for almost all activities with the TV.

talk about the relation between devices

phone charger
or a Samsung galaxy A-70 with a thunderbolt charger to be precise, which can be seen in figure 5. We chose this appliance, because the phone charger is a rather new, but frequently present, device in the current household and because a phone charger is a very low power consuming device. Including a low power consuming device allows us to see how well the implementation does on such devices. This device was mostly plugged in before bedtime, and consumed power until the phone was fully charged.

desk lamp

Figure 4: TV and ps4

Figure 5: phone charger

Figure 6: desk lamp

or a 30W lamp to be precise, which can be seen in figure 6. We chose this appliance because lights are relatively low power consuming, but in large quantities present in households. This lamp, as with all lights, are mainly used in the evening or during cloudy weather.

living room lamp
or a 50W lamp to be precise, which can be seen in figure 7. 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 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.

washing machine
or a BEKO WTV71483CSB with energy label A+++ to be precise, which can be seen in figure 8. 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.

fridge
or the ikea LAGAN fridge/freezer with energy label A++ to be exact, which can be seen in figure 9. 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 7: living room lamp

Figure 8: washing machine

Figure 9: fridge

water heater
or the OK. OWK 103 to be exact, which can be seen in figure 10. According to the label the water heater can consume up to 2200W, as we can see from table 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 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.

laptop
or an Alienware R17 Gaming laptop to be precise, which can be seen in figure 11. We chose this appliance because since the 90's, computers can be seen as standard in households. 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 for example. 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. 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.

microwave
or a KOENIC KMW 4441 microwave/oven to be precise, which can be seen in figure 12. 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 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 at any time of the day, but long activation's where most present during the evening.

say something about the plot

The aggregate data was calculated by summing up the total appliance consumption. To evaluate whether or not this method would produce correct aggregate data we shall execute a small experiment:

Figure 10: water heater

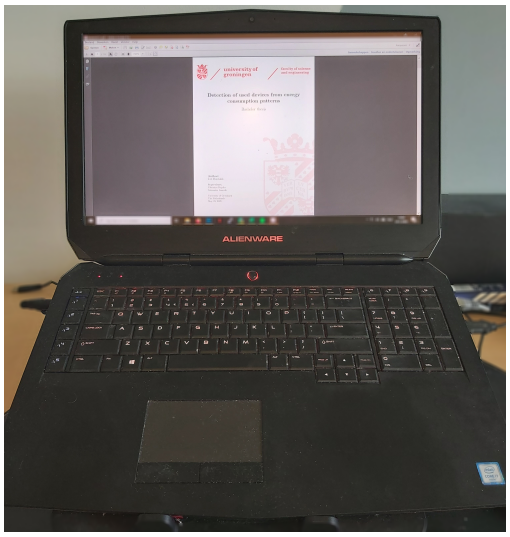


Figure 11: laptop

Figure 12: microwave

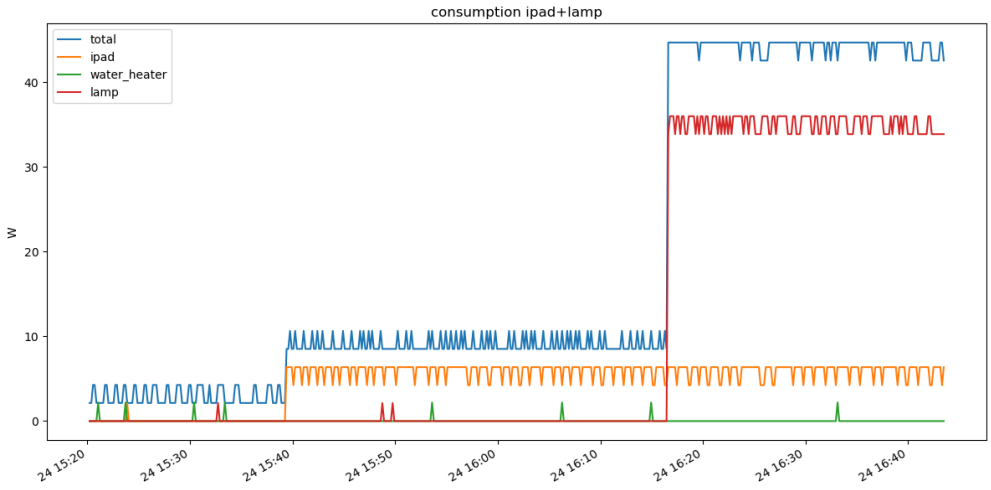
3 devices connected to plugwise devices (Circle's), an iPad charger, a lamp and a water heater, are added to a extension cord which in turn is connected to a plug-wise device, the circle +. So that the plugwise device was gathering the power consumption of the 3 devices. See figure 2 for the actual setup. 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 2. in figure 13a 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 13b and figure 13c it is clear that difference lies between 2W and 4W most of the time. It is also clear that the difference seems to grow larger when a large appliance is turned on. The difference of 2W and 4W can be explained by the power consumption of the plugwise devices. We have 3 plugwise devices connected to an extension cord, one plugwise devices has an average consumption of 1.1W [9], which means that the average consumption on the total would at least be 3.3W. The spikes could however indeed be an sign of the consumption of power due to the resistance in the extension cord.
Since the plugwise devices should not be part of the equation in a real household and the spike only represents 0.5% of the total power consumption, we will use the summed appliances for our aggregate power consumption data. Since implementations like Gao et al.[7] and Kelly et al.[3] use the same approach for their aggregate data, we will be able to implement a implementation of the algorithms closer to the originals.

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.

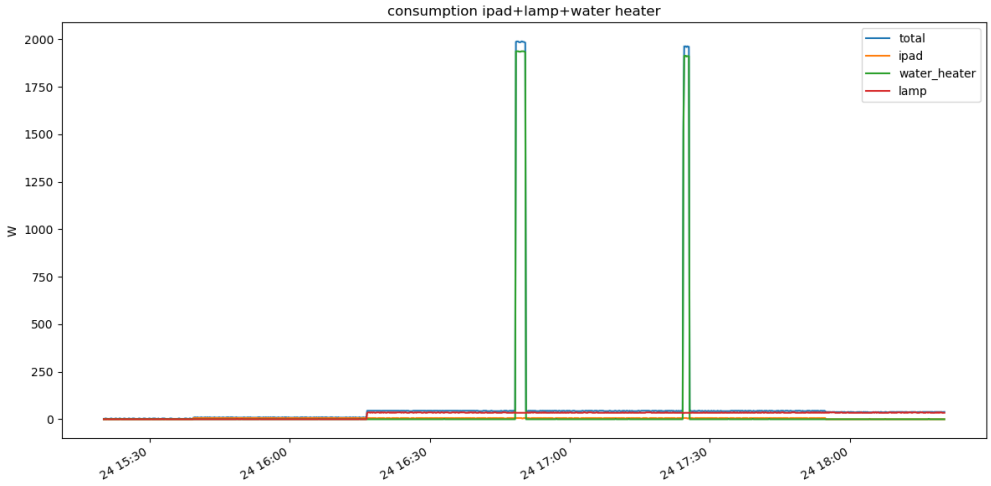
Using this implementation the power consumption of a set of 10 devices, which consisted of a microwave oven, a TV, a PlayStation 4, two lamps, an Alienware R17 laptop, a Samsung galaxy A70 phone charger, an iPad charger, a water heater and a washing machine, were individually analysed.

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

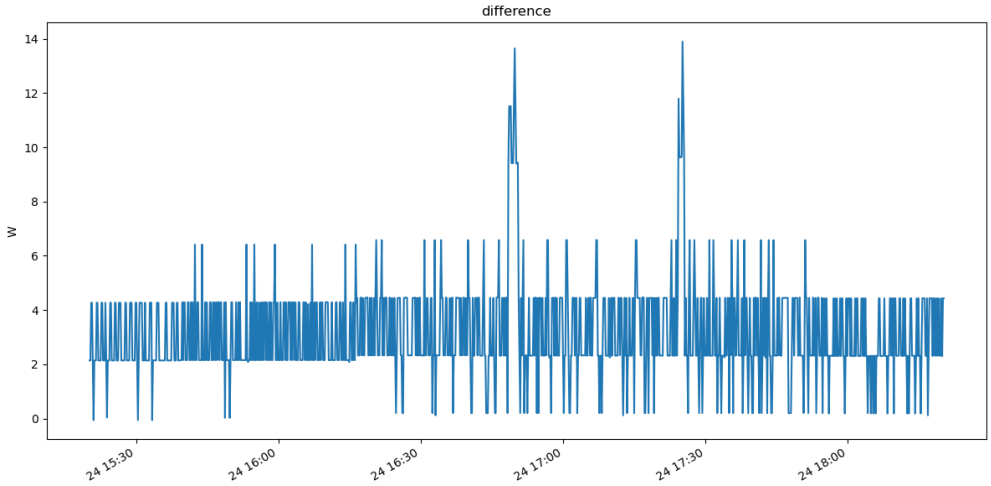
why our implementation is much harder to dissagregate



(a) consumption iPad+lamp



(b) consumption ipad+lamp+waterheater



(c) difference actual vs summed appliance consumption

Figure 13: actual total consumption vs summed appliance consumption outcomes

4 PROPOSAL

4.1 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 [10]. $e_n(t)$ 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.

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

4.2 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.

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

4.3 Methodology

The implementation we will be testing and extending is based on Gao's multilayer perceptron classifier[7]. 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.[3] and Luca et al.[11], 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.

Gao calls the breakpoint identifier the segmentation classifier

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.

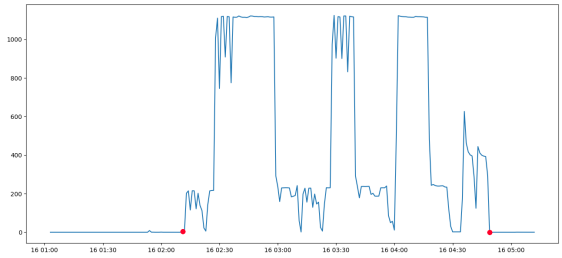
create table

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 14. 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 14b, 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 14b.

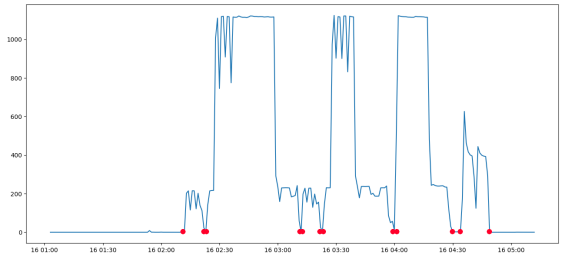
include the table with power threshold data

did i already say this?

segment labeler



(a) Gao's true breakpoints



(b) our true breakpoints

Figure 14: 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 15: measuring devices

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

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.

training the neural network

The classifiers are trained and created to match the specifications in Gao et al.[7].
We use 10-fold cross validation to train the neural network and measure the accuracy and we apply random oversampling [8] 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[6]. 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

Table 3: parameter values for scikit-learns Multilayer perceptron classifier

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

Table 4: confusion matrix of the breakpoints identifier on REDD

5 Experiments

5.1 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[7] 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.

Breakpoint identifier
The first classifier we test is the breakpoint identifier. The classification report of the breakpoint identifier can be seen in table 5 depicting the performance of the implementation. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[6] we measured an accuracy of **0.98** with a standard deviation of **0.03**. Gao doesn't specifically note the accuracy of the breakpoint identifier, but states that the breakpoint identifier "performs nearly perfectly in identifying breakpoints and non-breakpoints". Keeping in mind that Gao has a lower ratio of breakpoint vs non-breakpoint than our implementation due to the inclusion of the dishwasher and the fact that we do not manually label the breakpoints, we can conclude that this implementation is sufficiently similar to gao's breakpoint identifier.
In the confusion matrix of table 4 we see that most breakpoints are correctly identified, but 377 non-breakpoints are labeled as breakpoints. However the segment labeler might be able to solve the incorrect marked breakpoints. We will analyse the combined accuracy later.

segment labeler
The second classifier, the segment labeler, is up next. The classification report of the segmnet labeler can

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

Table 5: classification report of the breakpoints identifier on REDD with an accuracy of 0.96

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

Table 6: confusion matrix of the segment labeler on REDD

be seen in 7 depicting the performance of the implementation. The confusion matrix can be seen in table 6. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[6] we measured an accuracy of **0.88** with a standard deviation of **0.06**. In the article of Gao et al. an accuracy of **0.945** was measured. This difference in accuracy is mainly due to the absence of house 6 in the training data, resulting in less 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

multiappliance labeler

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 **164** were correctly identified. Combing these results by adding up the total and correct predicted labels like Gao we end up with **603** out of **710** correct. It got an accuracy of **85%** Compared to Gao's **91.2%**.

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

talk about the multi-state segments that are not able to be dis-sagregated

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

Table 7: classification report of the segment labeler on REDD with an accuracy of 0.86

confusion matrix	empty	fridge	microwave	washer dryer	dishwasher	multi
empty	226	120	1	0	0	0
fridge	1	284	12	2	3	13
microwave	0	1	48	0	1	23
washer dryer	0	2	0	2	3	6
dishwasher	0	14	4	1	7	38
multi	0	0	13	2	6	41

Table 8: confusion matrix of combined nets on REDD

classification report	precision	recall	f1-score	support
empty	1	0.65	0.79	347
fridge	0.67	0.90	0.77	315
microwave	0.62	0.66	0.64	73
washer dryer	0.29	0.15	0.20	13
dishwasher	0.35	0.11	0.17	64
multi	0.34	0.66	0.45	62

Table 9: classification report of the combined nets on REDD with an accuracy of 0.70

identifier. This test is important, since it is the accuracy as a whole that counts when the neural nets are implemented in real households.
To test the implementation as a whole, we first gave the input to the breakpoint indentifier as we did before. With the output from the breakpoint indentifier we look up the states of each of the given breakpoints and use that as our target for the segment labeler. Using the breakpoint from the breakpoint identifier and the aggregate signal, the input is created and tested on the segment labeler. the results can be seen in table 8 and table 9. As you can see, the accuracy of 70% for the combined test is considerably lower than the standalone implementation. It is also visible that combined implementation performs very poorly on identifying multi-label appliances, indicating that the multi-appliance disaggregation will be nearly impossible to do accurately.
To test the multi appliance disaggregation we apply the same original tactic of multi appliance disaggregation, but with the breakpoints given by the breakpoint identifier and the labels given by the segment labeler. From the **121** segments labeled as multi aplliance by the segment labeler only **303** out of **484** of the labels were found correctly. This is an accuracy of **60%**.

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

5.2 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

confusion matrix	non-breakpoint	breakpoint
non-breakpoint	42661	1838
breakpoint	180	681

Table 10: confusion matrix of the breakpoints identifier on the studio data

classification report	precision	recall	f1-score	support
non-breakpoint	1	0.96	0.98	44499
breakpoint	0.27	0.79	0.40	861

Table 11: classification report of the breakpoints identifier on the studio data with an accuracy of 0.96

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.

Breakpoint identifier

The first classifier we test on our own datais the breakpoint identifier. The classification report of the breakpoint identifier can be seen in table 11 depicting the performance of the implementation. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[6] we measured an accuracy of with a standard deviation of . Gao doesn't specifically note the accuracy of the breakpoint identifier, but states that the breakpoint identifier "performs nearly perfectly in identifying breakpoints and non-breakpoints". Keeping in mind that Gao has a lower ratio of breakpoint vs non-breakpoint than our implementation due to the inclusion of the dishwasher and the fact that we do not manually label the breakpoints, we can conclude that this implementation is sufficiently similar to gao's breakpoint identifier.
In the confusion matrix of table 10 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.

do k-fold cross validation

segment labeler

The second classifier, the segment labeler, is up next. The classification report of the segmnet labeler can be seen in 14 depicting the performance of the implementation. The confusion matrix can be seen in table 12. Using 10-fold cross validation and help of the cross_val_score() method of scikit-learn[6] we measured an accuracy of with a standard deviation of . One can see that although the accuracy remains the same, most of the appliances have very little to no data point. This is because due to the large quantity of devices in the dataset, a lot of the segments in this dataset are multi-appliance segments. This is why with the studio data as training data the implementation seems to be very good at mutli state dissagregation. Since a lot of single states are missing from the training data as well, the segment labeler will most likely have a lot of trouble dissagregating the single states that are present, even more so. 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 training data. This does not look good for the multi appliance disaggregation.

i know how to solve this problem, mention it here already?

conf. matrix	empty	tv	phone	d. lamp	c. lamp	washer	fridge	w. heater	laptop	ps4	multi
empty	108	0	0	0	0	0	0	0	1	0	0
tv	0	0	0	0	1	0	0	0	0	0	0
phone	0	0	50	0	0	0	0	0	0	0	0
d. lamp	0	0	0	0	0	0	3	0	2	0	0
c. lamp	0	0	0	0	0	0	0	0	0	0	0
washer	0	0	0	0	0	0	0	0	0	0	1
fridge	0	0	0	0	0	0	73	0	1	1	1
w. heater	0	0	0	0	0	0	0	0	0	0	1
laptop	0	0	0	1	0	0	18	0	69	23	12
ps4	0	0	0	0	0	0	0	0	0	0	0
multi	0	0	0	0	0	0	5	0	13	3	474

Table 12: confusion matrix of the segment labeler on the studio data

classification report	precision	recall	f1-score	support
empty	1	0.99	1	109
tv	0	0	0	1
phone charger	1	1	1	50
desk lamp	0	0	0	5
couch lamp	0	0	0	0
washing machine	0	0	0	1
fridge	0.74	0.96	0.83	76
water heater	0	0	0	1
laptop	0.8	0.56	0.66	123
PlayStation 4	0	0	0	0
multi	0.97	0.96	0.96	495

Table 13: classification report of the segment labeler on the studio data with an accuracy of 0.90

classification report	precision	recall	f1-score	support
empty	0.96	0.72	0.83	105
tv	0	0	0	1
phone charger	0.18	0.50	0.26	6
desk lamp	0	0	0	6
couch lamp	0	0	0	0
washing machine	0	0	0	1
fridge	0.56	0.78	0.65	68
water heater	0	0	0	1
laptop	0.85	0.34	0.48	1000
PlayStation 4	0	0	0	0
multi	0.72	0.94	0.81	1332

Table 14: classification report of the combined method on the studio data with an accuracy of 0.68

multiappliance labeler

The last step is disaggregating the multi-appliance segments further. We test this step on the **495** multi appliance segments from the testing data. From the total of **4950** single appliance labels within the multi appliance segments **3527**. so it achieves an accuracy of **71%**

combined implementation

We saw that the segment labeler expierences 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 equally bad as with the REDD dataset, as can be seen in table 13. This test on the combination of the two neural networks also gives us insight into the breakpoint identifier. For example we can see that the breakpoint identifier produces far less breakpoints for the phone, and way to much for the laptop. This is because the laptop has a very varying power consumption, while the phone consumes almost no power. . To test the multi appliance disaggregation we apply the same original tactic of multi appliance disaggregation, but with the breakpoints given by the breakpoint identifier and the labels given by the segment labeler. From the **1739** segments labeled as multi aplliance by the segment labeler only **12845** out of **17390** of the labels were found correctly. This is an accuracy of **73%**. Not considerably worse than the stand alone implementation, but still not sufficiently accurate.

talk about the multi-state segments that are not able to be dis-sagregated. are those counted?

maybe talk about this in breakpoints section

conclusion

Although the neural nets performed well on REDD, it struggles to dissagregate more appliances due to the increase of multi appliance segments. The breakpoint identifier also seems to have trouble with the laptop and the phone charger.

include k-fold prediction

6 Experiment 3: external features

6.1 conclusion

7 Experiment 4: generalizeability

7.1 conclusion

8 Conclusion

9 Future improvements

syntetic data generation might be beneficial to multi appliance dissagregation.

10 Contributors

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

Add guy from email and add alex

REFERENCES

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

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

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

[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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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