University of Essex

Department of Mathematical Sciences

MA981-7-SP: Dissertation

Non-intrusive Load Monitoring: State-of-the-art Deep Learning and Transfer Learning

Registration number: 1802655

Ryan O'Missenden

Date of submission: (14 January 2022)

Word count: 10,370

Contents

Abstract **1**

1 Introduction **3**

2 Background/Literature Review **4**

2.1 Energy disaggregation 4

2.2 Appliance states 6

2.3 Learning 7

2.4 Deep learning 7

2.5 Neural Networks 8

2.6 Batch size 8

2.7 Epoch 9

2.8 State of the art 9

2.9 Issue of state of the art 12

2.10 NILMTK 12

2.11 Transfer learning 13

3 Methodology **14**

3.1 General model structure 15

3.1.1 Artificial data 15

3.1.2 Fine-tuning 15

3.1.3 Sample rate 15

3.1.4 Epoch and Batch selection 16

3.2 Data sets 16

3.2.1 UKDALE 16

3.2.2 REDD 16

3.2.3 IAWE 17

3.2.4 REFIT 17

3.2.5 DRED 17

3.2.6 SMART . . . 17
3.2.7 Dataport . . . 17
3.3 Algorithms . . . 18
3.3.1 Mean . . . 18
3.3.2 Edge detection . . . 18
3.3.3 Combinatorial optimisation . . . 19
3.3.4 ExactFHMM, AFHMM and AFHMM_SAC . . . 19
3.3.5 Denoising Autoencoder . . . 20
3.3.6 Seq2Seq . . . 20
3.3.7 Seq2Point . . . 21
3.3.8 Recurrent Neural Networks . . . 21
3.3.9 WindowGRU . . . 22
3.4 Evaluation . . . 22
3.4.1 Mean Average Error . . . 23
3.5 Experiments: Stage 1 . . . 23
3.5.1 Trial 1 . . . 23
3.5.2 Trial 2 . . . 24
3.5.3 Trial 3 . . . 24
3.5.4 Trial 4 . . . 24
3.5.5 Trial 5 . . . 25
3.5.6 Trial 6 . . . 25
3.5.7 Trial 7 . . . 25
3.6 Experiments: Stage 2 . . . 25
3.6.1 Trial 1 . . . 25
3.6.2 Trial 2 . . . 26
3.7 Experiments: Stage 3 . . . 26
3.7.1 Trial 1 . . . 26
3.7.2 Trial 2 . . . 26

4 Results . . . 27

5 Discussion **35**

6 Conclusion **37**

A Stage 1: Appliance graphs **42**

A.1 UKDALE - Broadband 42

A.2 UKDALE - Computer 43

A.3 UKDALE - Dishwasher 44

A.4 UKDALE - External hard disk 45

A.5 UKDALE - Fridge 46

A.6 UKDALE - Kettle 47

A.7 UKDALE - Laptop 48

A.8 UKDALE - Microwave 49

A.9 UKDALE - Toaster 50

A.10 REFIT - Kettle 52

A.11 REFIT - Dish washer 53

A.12 REFIT - Cooker 54

A.13 DRED - Cooker 55

A.14 DRED - Electric heating element 56

A.15 DRED - Fan 57

A.16 DRED - Fridge 58

A.17 DRED - Laptop 59

A.18 DRED - Microwave 60

A.19 DRED - Oven 61

A.20 DRED - Sockets 62

A.21 DRED - Television 63

A.22 DRED - Toaster 64

A.23 DRED - Washing machine 65

A.24 REDD - Dish washer 66

A.25 REDD - Fridge 67

A.26 REDD - Light 68

A.27 REDD - Microwave 69
A.28 REDD - Sockets 70
A.29 REDD - Washer dryer 71
A.30 SMART - Fridge 72
A.31 SMART - Dish washer 73
A.32 SMART - Washing machine 74
A.33 SMART - Sockets 75
A.34 SMART - Boiler 76

B Stage 2: Appliance graphs 77
B.1 Train on REDD, test on SMART - Microwave 77
B.2 Train on REDD, test on SMART - Fridge 78
B.3 Train on REDD, test on SMART - Dish washer 79
B.4 Train on UKDALE, test on REFIT - Microwave 80
B.5 Train on UKDALE, test on REFIT - Fridge 81
B.6 Train on UKDALE, test on REFIT - Dish washer 82

C Stage 3: Appliance graphs 83
C.1 Train on UKDALE and REDD, test on REFIT - Microwave 83
C.2 Train on UKDALE and REDD, test on REFIT - Fridge 84
C.3 Train on UKDALE and REDD, test on REFIT - Dish washer 85
C.4 Train on UKDALE and REDD, test on SMART - Microwave 86
C.5 Train on UKDALE and REDD, test on SMART - Fridge 87
C.6 Train on UKDALE and REDD, test on SMART - Dish washer 88

D Python code 88

List of Tables

1 MAE: training on house 1 and testing on house 2 from UKDALE 27
2 MAE: training on house 1 before testing on house 2 from UKDALE 28
3 MAE: results from training on houses 3, 5 and 9 before testing on house 2 in REFIT 29
4 MAE: Type 1 results from training on the same house in DRED 29
5 MAE: Type 2 results from training on the same house in DRED 30
6 MAE: Type 3 and 4 results from training on the same house in DRED 30
7 MAE: training on Houses 1, 3 and 5 before testing on House 2 from REDD 31
8 MAE: training on house 1 and testing on house 3 from SMART 32
9 MAE: training on REDD and testing on SMART 32
10 MAE: training on UKDALE and testing on REFIT 33
11 MAE: training on UKDALE and REDD and testing on REFIT 34
12 MAE: training on UKDALE and REDD and testing on SMART 34

Abstract

The recent interest and concern of climate change has brought attention to how countries can reduce their impact on the climate in various ways, such as through the efficient use of energy.

Non-intrusive load monitoring (NILM) is a research area that is focused on disaggregating appliance level energy consumption from a single energy monitoring device such as a smart meter. In recent years variations of deep learning algorithms have emerged and obtained state-of-the-art results for NILM. This raised issues of comparability of methods, evaluation metrics and processing which led to the creation of the Non-Intrusive Load Monitoring Toolkit (NILMTK) as a way of standardising NILM research.

Most research into NILM has been focused on training and evaluating models on houses within a single data set which has led researchers to start expanding this further by investigating Cross-domain transfer learning which aims to leverage the power of transfer learning through training on multiple data sets or appliances in an effort to make models more robust to changes in energy usage patterns, social behaviours or appliances.

In this study deep learning methods are reviewed and evaluated against more traditional methods of energy disaggregation such as Hidden Markov Models (HMM). The study makes use of six data sets and the NILMTK package to develop a broad understanding that is standardised and can be replicated. Performance is also tested and evaluated for data sets within the same region in order to test the models generalisation ability across data sets with similar appliances and behavioural patterns. Finally cross-domain transfer learning is explored by combining a UK based and USA based data set for training before evaluating them against data sets from the UK and USA.

The results found that deep learning outperformed more traditional methods on almost every appliance and experiment. They also generalised well when tested on different data sets from the same region. Cross-domain transfer learning contained mixed results from both the neural networks and traditional methods but showed promise for this research area.

1 Introduction

The devastating effects of the change in the climate has brought about worldwide focus on how to counteract the human activities contributing towards it; with many countries in the European Union committing to reduce greenhouse gases by 80-95% by 2050 [1]. This has led researchers [2] to consider how machine learning can be leveraged to support the work on climate change, with one highlighted area being the optimisation of energy usage for buildings. Perez-Lombard L, Ortiz J, and Pout C [3] identified that one third of energy consumption across the world can be attributed to buildings; highlighting the need for a way to assist consumers in reducing their energy usage.

In Europe there has been a large scale roll out of smart meters for household buildings that monitor the total energy consumption whilst displaying it back to consumers and energy companies. This allows consumers to track their overall usage but does not provide any detailed breakdown of appliance specific consumption which could be used to manage how often appliances are used, identify any that are broken and generally give consumers more control over efficient use of their devices.

There are two methods of identifying appliance energy signatures, also known as energy disaggregation, termed Intrusive load monitoring (ILM) and Non-intrusive load monitoring (NILM). Intrusive load monitoring relies on metering devices that record the energy consumption of each appliance in the household but has to be situated nearby. This is known as intrusive because it consists of installing several meters within the building for each appliance. The results from ILM are very accurate but come at a high cost in regards to installing meters and wiring per household. Non-intrusive load monitoring (NILM) on the other hand takes information gathered from a single energy monitoring device within a household or commercial building, often a smart meter, to identify total energy consumption. This method is considered to be non-intrusive as it relies on one device, which in the case of smart meters are already installed within households to help inform customers of their overall energy usage, compared to ILM which requires extensive work on individual households. Because of the practical nature of NILM compared to costlier ILM, research has primarily considered how to disaggregate appliances from a single mains meter.

Previous approaches for NILM have included methods such as Combinatorial Optimatisation, Sup-

port Vector Machines, Graph Signal Processing, Decision Trees and Hidden Markov Models. Recently deep learning has emerged into the NILM research area with variations of Neural Networks achieving state-of-the-art performance levels.

Whilst Supervised learning methods such as neural networks have been found to achieve significant results with NILM, they require large amounts of information to provide accurate predictions as they often initialise with badly randomised weight parameters that require a lot of tuning before they converge to a solution that is optimal. This has identified the need and subsequent collection of various energy data sets, for example UKDALE [4] and REDD [5], which contain household appliance and mains level data collected via intrusive load monitoring devices.

Another emerging area within NILM is transfer learning, which utilises information gained from one problem and applies it to another related problem, often with significant improvements in performance. It has often been used in deep learning problems such as image classification where large data sets of images are used to help train models to learn important features such as edges and lines which can then be applied to a different problem.

Deep learning for NILM will be considered in this paper, with experiments conducted on the various neural networks to identify how well they generalise to unseen houses. Cross-domain transfer learning will also be considered to see how well the models perform when exposed to various different regional data sets containing a mixture of appliances and environmental factors.

2 Background/Literature Review

2.1 Energy disaggregation

George W.Hart first proposed the idea of analysing information collected from NILM devices in his research into utility load monitoring [6] where he also identified the disaggregation problem and proposed the initial edge detection model as a method to solve it.

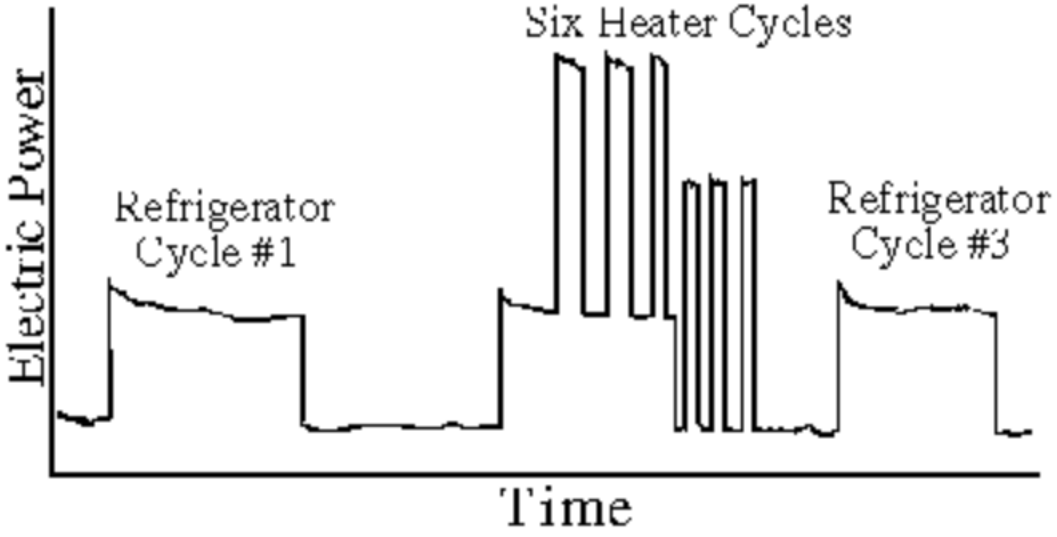


Figure 1: Example of energy disaggregation [6]

As displayed in figure 1, Energy disaggregation involves identifying individual appliance usage from the total energy consumption that is recorded because NILM devices such as smart meters aggregate the electricity usage of all appliances within the system and the signatures of the appliances become overlaid.

Energy disaggregation can essentially be defined by the following equation at any time step:

$$Y_t = \sum_{i=1}^{I} x_{it} + \epsilon_t$$

Y represents the aggregate mains power consumption observed and can be written as $Y = (y_1, y_2, \cdots, y_T)$ with y_t being the value of y at each timestep. The aggregate power consumption for each appliance can be shown as $X_i = (x_{i1}, x_{i2}, \cdots, x_{iT})$ where x_{it} is the value of X_i at each timestep. ϵ_t is the error term which is the difference between the aggregate power consumption and individual appliance consumption at each timestep.

2.2 Appliance states

Domestic appliances can be categorised into four types based on how they operate and consume energy. Appliances that switch between binary states (ON/OFF) at any point in time are considered Type 1 or known as two state appliances, where they only consume energy during the ON state. Examples of type 1 appliances are kettles, light bulbs and toasters. Type 2 appliances such as dishwashers and washing machines have a fixed number of states which the appliance repeatedly transitions between. The transitions through the states can be identified by examining the edges of the changes in power consumption. Type 3 appliances tend to be quite difficult to disaggregate as they are continuously variable devices where the appliances don't consume energy as a constant and makes states difficult to identify. Examples include light dimmer switches. The final type (Type 4) are appliances that are permanently in the ON state, continuously operating over large periods of time and have almost identical reactive and active power consumption. TV set top boxes, landline telephone sets and smoke detectors are all examples of this. Figure 1 provides an example of typical differences in load signatures from Type 1 - Type 3 appliances.

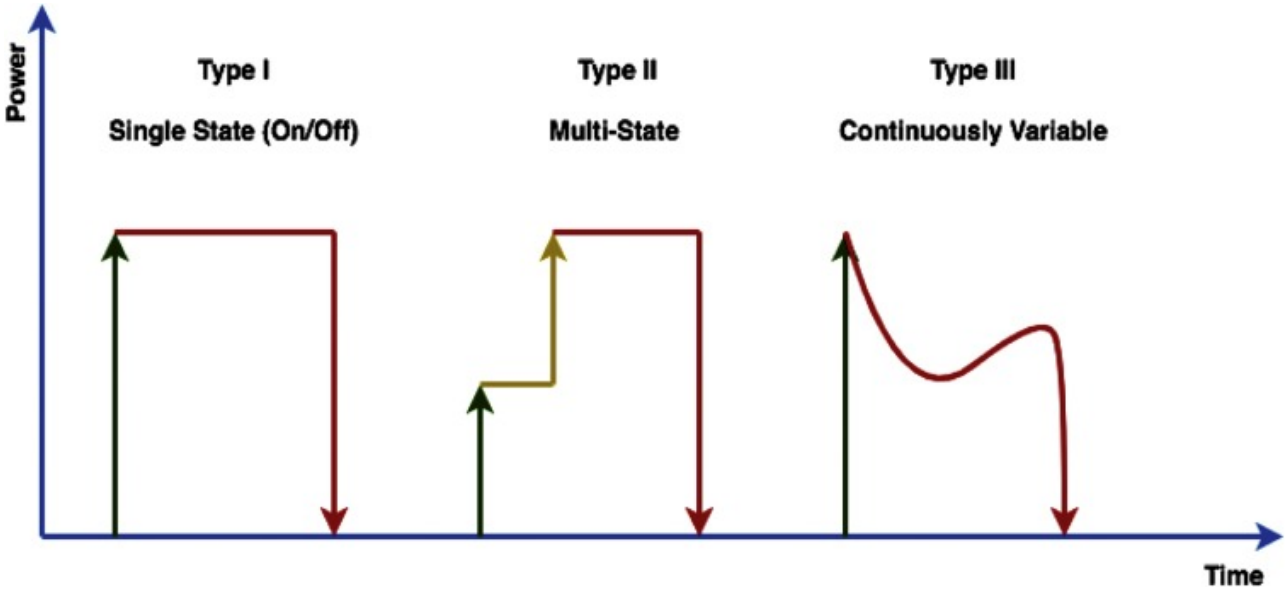


Figure 2: Differences in appliance type load [7]

2.3 Learning

The various approaches employed to tackle the problem of energy disaggregation and NILM can be grouped into two categories of learning – Supervised and unsupervised. Supervised learning relies on having a data set of observations where the outcomes are labelled, with the chosen algorithm learning the best mapping function of the inputs to outputs. In NILM data sets, intrusive load monitors are installed to collect energy consumption data from each appliance as well as the aggregated consumption from the mains. The ground truth data from the appliances and mains is then used to train models to disaggregate the appliance signatures from the mains. This method of learning allows for the model's performance to be evaluated and tends to be more accurate, however, the requirement of intrusive load monitors restricts the availability of data as well as incurring a high installation cost. Unsupervised learning is where there is a data set of observations but no labelled outcomes/ground truth. Unsupervised algorithms use similarity indexes to find similarities within the data, such as how clustering uses the distance between data points to group those close together. In relation to NILM, unsupervised learning algorithms only have access to the aggregated energy consumption where techniques such as edge detection can be performed. Unsupervised algorithms are likely to be less accurate in their predictions compared to supervised learning but are often faster and easier to scale up due to the lack of training time required.

2.4 Deep learning

Deep learning is a branch of machine learning comprised of algorithms based on Artificial Neural Networks (ANN) such as convolutional neural networks and recurrent neural networks where the word deep refers to the depth of the network's layers. The structure and concept of ANN's were inspired in the 1990's by human biology, specifically how neurons in the human brain operate, but did not gain a lot of traction until around 2012 when computing power and availability of large amounts of labelled data significantly increased.

2.5 Neural Networks

ANN's can be defined as directed acyclical graphs with units or artificial neurons that are connected, normally in layers and allow information to travel from one neuron to the next. There are generally three types of layers with ANN's; an input layer which receives the data, an output layer that produces the output and hidden layers which are located in between. As information is passed through the layers of the network, often called the forward pass, each neuron has a weight calculated for the sum of its input, adds a bias and flows through an activation function to produce a final output at that node before being passed to the next node. Backpropagation or a backwards pass is then conducted by computing the error between the prediction and the ground truth before using a method such as gradient descent to update the weights in the hidden layers in a direction that reduces the error.

2.6 Batch size

Batch sizes are a hyperparameter of neural networks that define how many samples the model processes before updating the weight parameters. The error from the predictions is calculated after each batch with the algorithm improving the performance by moving along the error gradient. Because there can be many batch divisions of the data set, the type of gradient descent is generally defined by three groups. Batch Gradient Descent is where the batch size is equal to the size of the training set, essentially creating one batch with all the samples. Stochastic Gradient Descent is where the batch size is equal to one sample so that the number of batches will be the same number of samples in the training set. Mini-Batch Gradient Descent is in the middle of Batch and Stochastic Gradient Descent where the batch size is more than one but less than the total number of samples in the data set.

2.7 Epoch

Another hyperparameter of neural networks are epochs which refer to the number of training cycles that are to be made through the entire data set, with a forward pass and backwards pass counting as one cycle. A single epoch would mean all the data is passed forward and backwards only once through the network. The optimal number of epochs varies as it relies on factors such as the size of the data set and the characteristics of the algorithm being deployed.

2.8 State of the art

Building on from the work of Hart there have been several areas explored to solve the disaggregation problem for NILM. The current state-of-the-art methods tend to focus around using variations of Hidden Markov Models (HMM) and artificial neural networks which have achieved significant results.

A variation of HMM was used by Goncalves H, Ocneanu A, Berges M, and Fan R [8] who focused on developing an unsupervised algorithm that aimed to determine the power consumption, number and state of appliances without any prior knowledge. This was achieved by creating clusters to identify event changes before reconstructing the power signal using a matching pursuit algorithm. They were successful in identifying large appliances but found that smaller appliances were difficult to separate as they tended to cluster together.

Kolter JZ and Jaakkola T [9], investigated energy disaggregation by using an Additive Factorial Hidden Markov Model (AFHMM) which was an improvement on traditional FHMM's where accurately inferring from the model is difficult with approximation algorithms prone to local minimas. They tackled the inference problem by constraining the allowed posteriors so that at any given time only one HMM changed state achieving a good performance.

Parson O, Ghosh S, Weal M, and Rogers A [10] also considered HMM's by proposing two unsupervised methods, the first to create a general behaviour model of appliances by modelling each appliance instance as a HMM and the second which tunes instances of specific appliances by using

the general behaviour model to generate appliance signals from the aggregated mains consumption. They found that the tuning model outperformed the state-of-the-art at the time which was a factorial HMM.

Methods other than HMM's have also achieved a good performance in dissagregation such as Liao J, Elafoudi G, Stankovic L, and Stankovic V [11] who proposed the use of Dynamic Time Warping (DTW) to classify appliances, which is unsupervised time-series based method that measures the similarity between two sequences which may vary in length. The method is computationally expensive as it requires creating a library that stores appliance signatures before pattern matching between the window and the library to classify appliances. They found that the method was adept at identifying multi-state appliances such as a washing machine but struggled with appliances that have low consumption levels such as a boiler. In their experiments they found that the model had a high success rate outperforming the state-of-the-art HMM used as a baseline.

Cominola A, Giuliani M, Piga D, Castelletti A, and Rizzoli AE [12] also used DTW to extract the consumption of appliances after the operating status had been clustered using fuzzy C-means. They found their method to be less computationally expensive compared to HMM's and effective in disaggregation, especially in situations where appliances are operated at the same time.

Another method explored by He K, Stankovic L, Liao J, and Stankovic V [13] is Graph Signal Processing (GSP) which is a technique that aims to graph the signal structures through the correlation of the data points. Their method aims to initially minimise the variation in the total graph before using simulated annealing to improve it further. A HMM and Decision tree model were also used as a state-of-the-art baseline and it was found that their method outperformed both models.

Zhao B, Stankovic L, and Stankovic V [14] also investigated GSP but with an unsupervised method that doesn't require any prior knowledge or training period. It works by detecting event windows, extracting features through clustering and then classifying appliances based on classes that have already been defined. Their method when tested against two HMM's and a supervised GSP baseline was found to be on par with the GSP model and outperform the two HMM's. Whilst this approach is unsupervised it does rely on each appliance signature being saved in a database and manually

labelled.

Altrabalsi H, Stankovic V, Liao J, and Stankovic L [15] approached the NILM problem by using a mixture of Support Vector Machines (SVM) and K-Means clustering. SVM's have been found to perform well in classifying appliances as well as being able to scale as the number of appliances grow which is a downside to HMM's. They do require a large training data set however, which makes them computationally expensive and not suited to providing real time disaggregation. The aim of the study was to use K-Means clustering, a low computation method, to extract subsets of the data and reduce the amount of training samples as well as the computational expense for the SVM model. The overall performance was good but the main benefit was the reduction in processing time.

Kelly J and Knottenbelt W [16] took the initial step of investigating the ability of deep learning techniques on energy disaggregation, specifically using three neural networks including a network that regresses the average power, start and end time to create an appliance activation window that they call 'rectangles', a denoising autoencoder (dAE) and a Recurrent Neural Network (RNN) with long short-term memory (LSTM) units. They benchmarked their work against the Combinatorial Optimisation and FHMM algorithms featured in the NILMTK package. Generally, across all appliances they found that their dAE and Rectangles model outperformed the two benchmark algorithms in the majority of the evaluation metrics such as the F1 score and Mean Absolute Error (MAE). Whilst the RNN model did outperform the benchmark on the two state appliances such as kettles, it struggled when disaggregating multi-state appliances such as the washing machine. Kelly J and Knottenbelt W [16] did highlight however that their benchmark algorithms wouldn't be considered state-of-the-art as standard models were chosen with little to no fine tuning. Linh NV and Arboleya P [17] also considered RNN for NILM but trained and tested on the REDD data set [5] compared to the UKDALE data set [4] in Kelly J and Knottenbelt W [16]. They found similar results in that the RNN model generally performed well but struggled with multi-state appliances.

Bonfigli R, Felicetti A, Principi E, Fagiani M, Squartini S, and Piazza F [18] attempted to improve upon the performance of the dAE model initially proposed in Kelly J and Knottenbelt W.

They added early stopping and variable step size to the training phase as well as a median filter in the disaggregation phase to recombine the overlapping output windows. When compared to the AFAMAP model proposed in Kolter JZ and Jaakkola T [9] which they considered to be the best variation of HMM's, they found that their improved dAE model outperformed the average of all appliances.

2.9 Issue of state of the art

The wealth of studies into NILM such as those described above, suffer from an issue of comparability which makes identifying a state-of-the-art modelling technique or method quite difficult. Studies often train and test on different houses within a single data set [16], at different sampling rates and using specific appliances or break into new areas such as cross domain training/testing [19]. There are also major differences in similar algorithms that are implemented, such as the varying structures of neural networks created for disaggregation. There are also various studies considering evaluation metrics used for benchmarking the performance of models such as Pereira L and Nunes N [20] as well as studies that suggest new metrics for NILM like in Klemenjak C, Faustine A, Makonin S, and Elmenreich W [21]. All of these disparate experimental frameworks mean algorithms, methods and results cannot be compared to assess the efficacy of the project or built upon to improve the current state-of-the-art research.

2.10 NILMTK

Batra et al [22] identified that research into NILM had only focused on singular data sets in their experiments which did not allow for evaluation against different households to test how the models generalised. Further, there were various algorithms created and tested by researchers which did not allow for comparisons against a benchmark meaning older models could not be compared to the new state of the art models being created. Batra et al developed the Non-Intrusive Load Monitoring

Toolkit (NILMTK) which is an open-source Python library designed to reduce the entry barriers of researching NILM and to address the issues they found. The toolkit works to load various household energy data sets, allowing users to conduct exploratory analysis of the data, implement a selection of disaggregation algorithms, evaluate their results with a basket of metrics and offers an experimental API which acts as a complete pipeline for energy disaggregation.

NILMTK was enhanced in 2019 with the addition of NILMTK-Contrib [23] which introduced current state of the art deep neural network algorithms that have been found to produce improved disaggregation results.

2.11 Transfer learning

With the complex nature of energy disaggregation and various regional data sets that exist, transfer learning can introduce disaggregation models to a wide variety of appliance energy signatures and different usage patterns across regions such as Europe and USA.

Only a few studies consider Transfer learning in their research into energy disaggregation which means it can be performed in several ways. One of the methods considered is appliance transfer learning which aims to train models on one appliance, learning as many patterns in the signal as possible before transferring this knowledge on to other unseen appliances. D'Incecco M, Squartini S, and Zhong M [19] considered this method of transfer learning in an attempt to create an 'oracle' NILM that can be applied worldwide without the need for large or appliance specific training data. They trained layers in a Convolutional Neural Network (CNN) from a single appliance (washing machine) which can then be used as a base on which to train other appliances.

Another method is cross domain transfer learning which focuses on training models on consumption data from houses within different data sets to enable them to generalise better when exposed to new information. The models are then tested on 'unseen' houses within different data sets to evaluate how well they perform. Batra N, Jia Y, Wang H, and Whitehouse K [24] investigated the concept of cross domain transfer learning for two cities within the same data set. They used the

Dataport data set for their study which is information collected from the USA, specifically focusing on Austin, Texas and San Diego, California. An accuracy score similar to that of the state-of-the-art was achieved despite the differences in the cities energy consumption which is attributed to the differences in weather conditions as Austin is warmer than San Diego.

Humala B, Nambi ASU, and Prasad VR [25] took this area further by investigating the effect of different regions on the generalisation ability of the model, specifically the USA and UK regions through the use of the REDD and UKDALE data sets. Even though the energy usage patterns likely differed due to the habits and appliances in USA and Europe Humala B, Nambi ASU, and Prasad VR [25] managed to achieve a similar state-of-the-art accuracy score.

Three data sets – REDD, REFIT and UKDALE were considered in Murray D, Stankovic L, Stankovic V, Lulic S, and Sladojevic S [26] who investigated cross-domain transfer learning on the same region (Europe) and different regions (USA and Europe) using a CNN and GRU network. Both networks were found to generalise well when compared to the baseline performance from training and testing on the same data set.

3 Methodology

The research aim of this dissertation is to explore and evaluate the various state-of-the-art deep learning models for NILM whilst also considering the performance of cross domain transfer learning. The framework for experimentation will be formed of three stages: the first stage will consider the models performance on houses within the same data set with training being conducted on one or more houses before being tested on another. The second stage will review the performance of the models across data sets within the same region (USA and Europe), with training occurring on a house in one data set before being tested on another unseen house and data set. The final stages will consider model performance on cross domain transfer learning where training will be conducted on houses within two data sets from different regions (one USA and one European) before being tested on another two data sets from both regions.

For all experimental stages only models and data sets that work with the NILMTK package will be used. This is to provide a standardised approach to how models are being implemented as well as

the evaluation metrics used to measure performance. Because the NILMTK API takes a dictionary of parameters, it requires the same appliances are featured in the training data set as well as the test set to work. This impacts on the appliances selected in each experiment with the majority being evaluated in stage one as houses in the same data set often share similar appliances whereas stage two and three will feature less as the regions only share a select number.

3.1 General model structure

3.1.1 Artificial data

Artificially aggregated data has not been used in this project, mainly to test how the models perform on real observed data that contains noise due to this being what they will experience when deployed as a practical application.

3.1.2 Fine-tuning

Fine tuning of the algorithms to increase their performance on specific appliances or data sets will not be used in this study as the aim is to review how well the models perform when exposed to unseen houses and appliances.

3.1.3 Sample rate

The sample rate parameter determines the rate in seconds of how the data is sampled and the selection can adjust how well the model performs. A sample rate too high can lead to underfitting due to loss of features as there are bigger gaps between the data points whereas a rate too low can provide granularity of the data points but in doing so captures the noise present. A sample rate of 60 is recommended and chosen for all models in this study as it helps to smooth the noise whilst maintaining features.

3.1.4 Epoch and Batch selection

Due to the lengthy periods of time selected for training the models, there are a large number of data points in the training sets and so makes the processing computationally expensive. To balance the computational expense whilst maintaining performance of the training period, epochs of 50 have been chosen for all models and 30 selected for WindowGRU with batch sizes of 1024.

3.2 Data sets

This dissertation makes use of 7 data sets: UKDALE, REDD, DRED, Dataport, IAWE, SMART and REFIT. A selection of data sets from different countries were selected to test not only the generalisation ability of the models but also explore the effects of cross-domain transfer learning.

3.2.1 UKDALE

The UK Domestic Appliance-Level Electricity (UK-DALE) [4] is an open-source energy data set collected from mains and sub-meters installed within 5 houses in the United Kingdom. In 2015, UK-DALE was re-released with further recordings for house number 1 which increased its coverage from 1 to 2.5 years. The information covers the period between 2013 – 2015 where data was sampled at a rate of 1 and 6 seconds, with the 1 second sample rates being recorded for 3 of the houses.

3.2.2 REDD

The Reference Energy Disaggregation (REDD) [5] released in 2011 was the first publicly available household energy data set and has been used in a number of energy disaggregation research projects. It measures the mains and sub-meter energy usage of 6 households in the USA, sampling at a rate of 1 second over the course of several weeks.

3.2.3 IAWE

The Indian data set for Ambient Water and Energy (IAWE) [27] data set contains information on a single household in India over a period of 73 days, where 33 sensors monitoring the aggregated and sub-metered electricity is sampled every 1 second.

3.2.4 REFIT

The REFIT data set [28] is comprised of energy consumption data sampled every 8 seconds for mains and appliances collected from 20 homes in the UK between 2013 and 2015. Initial exploration into the data set highlighted that the information for house 13 and 21 included photovoltaic energy generation and were excluded from this study.

3.2.5 DRED

The Dutch Residential Energy data set (DRED)[29] is an openly available data set from The Netherlands. Aggregated and appliance level usage is captured over the space of 6 months for 11 appliances in a single household.

3.2.6 SMART

SMART [30] is a project that aims to support the optimisation of household energy consumption by providing a range of information including energy data for 114 apartments and 7 homes as well as information on photovoltaic generation and the weather. In this study we consider the data set for the 7 homes, selecting only the first 3 as they have a converter script available for the NILMTK package. The 3 homes only have a small number of appliances available but contain 3 years worth of data for them.

3.2.7 Dataport

Dataport [31] is the largest data set worldwide for household energy consumption, collected from 1,000 homes across four states within the USA, specifically Colorado, California, Texas and New York. The data set is too large to download the full amount of information and requires an account to be authorised to access the information stored in an SQL database.

3.3 Algorithms

The models selected for review are the NILMTK neural networks (RNN, DAE, WindowGRU, Seq2Seq and Seq2Point) and the baseline non-neural network (Mean, CO, Hart85, AFHMM, AFHMM_SAC and ExactFHMM). Discriminative Sparse Coding was also initially explored as a non-neural network algorithm but had significant errors when deployed on multiple data sets and consequently dropped from this study.

3.3.1 Mean

The Mean algorithm is a fairly simplistic model that has been found to perform well in comparison to more complex disaggregation methods and because of this has been used as a benchmark in various papers [23]. It works by calculating the average power state of appliances whilst training and storing them individually. Every time an appliance is encountered again the mean power state for that appliance is updated. When predicting, the algorithm returns the mean value for all appliances where it assumes appliances are always ON.

3.3.2 Edge detection

The edge detection algorithm in NILMTK is based on the unsupervised model initially suggested by George Hart in his pioneering work in 1985. Hart suggested detecting the edges of the energy signal to identify stepwise changes in the power, splitting the information into steady and transitory states. The stepwise changes are commonly appliances switching state such as a kettle being turned ON or OFF which can then be used for disaggregation. Since this model is unsupervised, a best-case mapping exercise is undertaken in the toolkit where the output of the algorithm is linked to categories of appliances that obtain the maximum accuracy.

3.3.3 Combinatorial optimisation

Combinatorial optimisation assumes appliances can be in 1 of K number of states with each state having a level of energy consumption. The algorithm allocates appliances to states with the aim to minimise the difference between the total energy usage of the household and the total energy usage of the various appliances. Combinatorial optimisation has been shown to perform relatively well in energy disaggregation and is often used as a benchmark; however, it does not scale well as the number of appliances increase due to the complexity of the calculations.

3.3.4 ExactFHMM, AFHMM and AFHMM_SAC

The factorial variation of a hidden markov model (FHMM) considers each appliance to be a hidden markov model (HMM) that has Ki states. The NILMTK package [23] uses a binary vector to represent the state of the appliance at each timestep, giving a value of 1 when the appliance is at state k and 0 when it is not. Batra et al [23] implement an exact method for their FHMM using the Kronecker product to combine the HMMs and allow for the inference of each appliances hidden states over time.

The Approximate Factorial Hidden Markov Model (AFHMM) is an variation of a HMM that aims to resolve common issues such as getting stuck at local minimum points as well as the computational expense. This is done by relaxing the values of states into [0, 1] and converting the optimisation problem into that of a convex one.

AFHMM with signal aggregate constraint (SAC) is an enhancement of the FHMM through the addition of SACs that provide an expected value for the appliance aggregate over a set time period. When these two methods are combined they allow for the optimisation problem to be converted into a relaxed convex one[23].

Similar to combinatorial optimisation, the calculations for the variations of HMMs grow exponentially in complexity as the number of appliances increase and this makes them quite computation-

ally expensive, AFHMM and AFHMM_SAC in particular.

3.3.5 Denoising Autoencoder

Autoencoders are an unsupervised deep neural network that aims to reconstruct input data as outputs by learning the representations of the inputs. It is formed of two parts; an encoder which downsamples the data, reducing the dimensionality and a decoder which reconstructs the input from the latent space. The latent space representation learnt by the model allows for important features to be extracted, however, autoencoders can suffer from overfitting and risk just copying the input. Denoising Autoencoders (dAE) are a variation of autoencoders that corrupt the input through the addition of noise before training the model to predict the original uncorrupted input with any differences from the original being punished by a loss function. This allows the model to learn to remove noise and extract useful features from the data.

dAE are often used in applications such image pre-processing to clean old images of grain or improve image quality for an optical character recognition algorithm. It was proposed by Kelly et al that dAE's could be used for energy disaggregation by considering appliance signatures as noisy, such that the appliance xjt and noise nt are represented by the mains signature yt.

The dAE denoises each appliance individually using a sliding time window of the mains consumption and outputs the predicted consumption of the appliances for that window. The denoising of each appliance means groups of appliances require multiple models to be trained.

3.3.6 Seq2Seq

Sequence to sequence (Seq2Seq) is a deep learning model that has primarily had major success in language processing, such as translation, where sentences from different languages have varying lengths. The model works by transforming one sequence to another through the mapping of inputs and outputs of different fixed lengths. Each element of the input is forward propagated through an encoder (usually an RNN) that condenses the information into a vector and acts as an initial state for the decoder which produces an output prediction for each timestep. In the case of the model

constructed in NILMTK for energy disaggregation, the input is the mains power reading which is mapped by a non-linear regression to the output sequence of an individual appliance at each timestep. The architecture follows that proposed by Zhang C, Zhong M, Wang Z, Goddard N, and Sutton C [32] as shown in figure 3.

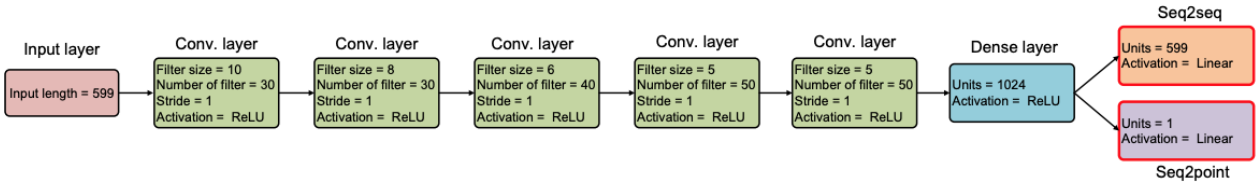


Figure 3: Sequence to Sequence and Sequence to Point architectures used in NILMTK [32]

3.3.7 Seq2Point

Zhang C, Zhong M, Wang Z, Goddard N, and Sutton C [32] proposed a new architecture (Figure 3) based on Seq2Seq called Sequence to point (Seq2Point) that aims to predict the midpoint of an appliance consumption time window rather than predicting the whole window. Zhang C, Zhong M, Wang Z, Goddard N, and Sutton C theorised that the midpoint of the appliance window should relate to the information before and after that point. The model works by adjusting the input of the neural network to that of a time window for the mains consumption and maps it to the midpoint of the corresponding output window.

3.3.8 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a variation of artificial neural networks that are regularly used for sequential data such as time series or language processing. The model works by feeding the inputs through one at a time, saving each output and connecting it back to the next input layer for future predictions. By considering the previous output alongside each input, RNN's are able to learn the long-term dependencies in sequential data. Whilst retaining all this information in memory makes RNN's quite powerful, it does become very complex with a large computational expense as well as potentially suffering from the issue of vanishing or exploding gradients. The issues with

gradients occur during the back-propagation when the weights of the network are updated with an error gradient either too small or large. An update too small can prevent weights from changing value and result in the learning rate of the model slowing or even stopping, whereas an error gradient too large can accumulate, causing huge parameter changes and creating an unstable model that cannot learn. To overcome this issue, RNNs use Long Short-Term Memory (LSTM) units with an in-built memory cell that stores information, using gates to regulate what information is retained and output at a certain time step. RNN's ability to handle sequential data makes it a suitable algorithm for the information gathered in NILM such as mains power readings that are sampled every few seconds. Kelly J and Knottenbelt W proposed an RNN architecture in the NILMTK which takes a single sample of the mains power consumption as the input and predicts a single sample for the appliance as an output at each timestep.

3.3.9 WindowGRU

In the NILMTK package, WindowGRU is an algorithm based on research from Krystalakos O, Nalmpantis C, and Vrakas D [33] who took the architecture of the RNN in the toolkit and reduced the computational cost by replacing the LSTM units with Gated Recurrent Units (GRU). As well as adding the GRUs, the sizes of the recurrent layers were optimised, reducing the parameters of model during training by 60% in comparison to the RNN. The model calculates the consumption of an individual appliance for the last time point by taking a window of mains consumption as the input.

3.4 Evaluation

To evaluate the performance of the models predictions in the various experiments the Mean Average Error (MAE) will be used. F1 score, Root Mean Square Error (RMSE), relative error, R^2, Normalised Disaggregation Error (NDE) and Normalised Error in Assigned Power (NEP) are available

in the NILMTK but are not widely used across studies and so are not considered within this study. Accuracy, precision and recall are currently not supported within NILMTK and consequently not used within this study either.

3.4.1 Mean Average Error

MAE measures the absolute difference between the prediction and the ground truth at each timestep before the errors are aggregated and averaged to give an average value per timestep. It can be defined with the following equation:

$$\text{MAE } = \frac{1}{T}\sum_{t=1}^{T}|y_t - y_t| \tag{3.1}$$

MAE is robust to outliers and is a popular metric used within many NILM studies

3.5 Experiments: Stage 1

This stage revolves around training and testing models on houses in the same data set. It is to see how the models perform when faced with disaggregating houses with a similar makeup.

3.5.1 Trial 1

The first trial in this stage trains on house 1 in UKDALE (Europe) because it has the longest time period and largest number of appliances, before testing on house 2. A selection of appliance types has been chosen (Type 1: Kettle, Microwave and Toaster, Type 2: Fridge and Dishwasher, Type 3: Computer and Laptop, Type 4: Broadband router and External hard disk) to test the model's ability to learn different types of signatures. The training time period selected for house 1 was a year (26/04/2016 - 26/04/2017) and test time period was almost 8 months, the full length for house 2 (17/02/2013 - 10/10/2013).

3.5.2 Trial 2

The second trial trains on house 3, 5 and 9 in REFIT (Europe) and tests on house 2. A selection of appliance types shared by the houses has been chosen (Type 1: Microwave and Kettle, Type 2: Dishwasher, Type 3: Television, Type 4: None). More appliances were considered but errors within the NILMTK package would not allow training for the Fridge or Washing machine as it found similar appliance names within the data set e.g. Fridge freezer. The training time period selected for houses 3, 5 and 9 was roughly 1.5 years (17/12/2013 - 02/06/2015) and test time period was just over 1.5 years for house 2 (17/09/2013 - 28/05/2015).

3.5.3 Trial 3

The third trial trains and tests on DRED (Europe) which only contains data for 1 house. A selection of appliance types has been chosen (Type 1: Cooker, Microwave, Oven and Toaster, Type 2: Fan, Electric heating element, Washing machine and Fridge, Type 3: Laptop and Television, Type 4: Sockets). The training time period selected was 3 months (05/07/2015 - 05/10/2015) and test time period was 2 months (06/10/2015 - 05/12/2015).

3.5.4 Trial 4

The fourth trial trains on houses 1, 3 and 5 in REDD (USA) and tests on house 2. A selection of appliance types has been chosen (Type 1: Light and Microwave, Type 2: Fridge, Dishwasher and Washer Dryer, Type 3: None, Type 4: Sockets). The training time period selected for house 1, 3 and 5 was roughly a month (18/04/2011 - 22/05/2011) and test time period was the same for house 2.

3.5.5 Trial 5

The fifth trial trains on house 1 in SMART (USA) and tests on house 3. There were only a few appliances within this data set so the following were selected (Type 1: None, Type 2: Fridge, Dishwasher and Washing machine, Type 3: None, Type 4: Sockets and Boiler). The training time period selected for house 1 was 3 years (31/12/2013 - 31/12/2016) and test time period for house 3 was 2 years (31/12/2013 - 31/12/2015).

3.5.6 Trial 6

The sixth trial aimed to train and test on the only house within IAWE (Asia). Unfortunately during the experimentation stage none of the models would produce results with this data set, throwing errors with the NILMTK package. This was likely due to its small window of time for training and testing as the error revolved around not enough data being available when it came to predictions.

3.5.7 Trial 7

The seventh trial aimed to train and test on the three houses within Dataport (USA) across a multitude of appliances for roughly a year. Unfortunately this was not possible due to access not being granted to Dataport in time for experimentation.

3.6 Experiments: Stage 2

The second stage investigates the cross-domain transfer learning performance for data sets within same region. It is split into 2 trials, focused on training models using REDD and UKDALE before testing on data sets within their respective regions. This is to provide a cross-domain baseline of model performance before UKDALE and REDD training is combined in stage 3.

3.6.1 Trial 1

The first trial of the second stage considers USA data sets, training on house 1 in REDD and testing on house 2 in SMART. the appliances for the trial were selected based on whether they were shared across UKDALE, REDD, REFIT and SMART to provide a comparison to results obtained from stage 3. A selection of appliance types has been chosen (Type 1: Microwave , Type 2: Fridge and Dish washer, Type 3: None, Type 4: None). The training time period selected for house 1 in REDD

was just over a month (18/04/2011 - 24/05/2011) and test time period was three years for house 2 in SMART (31/12/2013 - 31/12/2016).

3.6.2 Trial 2

The second trial of the second stage focuses on European data sets, training on house 2 from UKDALE before testing on house 2 from REFIT. As with the first trial, the appliances were selected based on being shared by the four data sets used in stage 3. A selection of appliance types has been chosen (Type 1: Microwave , Type 2: Fridge and Dish washer, Type 3: None, Type 4: None). The training time period selected for UKDALE house 2 was 6 months (16/04/2013 - 10/10/2013) and test time period for REFIT house 2 just under 2 years (17/09/2013 - 28/05/2015).

3.7 Experiments: Stage 3

The third and final stage of experiments combines two different regional data sets for the training phase, specifically UKDALE and REDD. It is comprised of two trials where the generalisation performance of the models after combined training is evaluated against a European and USA data set, specifically REFIT and SMART.

3.7.1 Trial 1

The first trial of the third stage trains house 2 from UKDALE and house 1 from REDD before testing on house 2 from REFIT. As with stage 2 three appliances have been chosen based on being shared by the four data sets in question (Type 1: Microwave , Type 2: Fridge and Dish washer, Type 3: None, Type 4: None). The training time period selected for UKDALE house 2 was a year (16/04/2013 - 10/10/2013) and just over a month for REDD house 2 (18/04/2011 - 24/05/2011). The test time period was just under 2 years for REFIT house 2 (17/09/2013 - 28/05/2015)

3.7.2 Trial 2

The second trial of the third stage trains house 2 from UKDALE and house 1 from REDD before testing on house 2 from SMART. A selection of appliance types has been chosen (Type 1: Microwave , Type 2: Fridge and Dish washer, Type 3: None, Type 4: None). The training time period selected for UKDALE house 2 was a year (16/04/2013 - 10/10/2013) and just over a month for REDD house 2 (18/04/2011 - 24/05/2011). The test time period was 3 years for SMART house

2 (31/12/2013 - 31/12/2016).

4 Results

Trial one trained/tested on houses within UKDALE with the MAE results for type 1 and 2 appliances shown in table 1 and the power usage of the ground truth and model predictions visualised in appendix A.1 - A.9. The results show that the neural networks (NN) were generally quite successful at disaggregating the energy consumption of type 1 (Two state) and type 2 (Multi-state) appliances, achieving a low MAE in comparison to the non-neural networks (Non-NN). Seq2Point and Seq2Seq had the best performance generally across all appliances in table 1 compared to the NN and Non-NN algorithms, except the Dish washer where WindowGRU obtained the lowest error. For the non-NN algorithms the mean performed surprisingly well, even outperforming some of the state-of-the-art NNs on appliances such as the computer and toaster. The AFHMM_SAC algorithm also had a consistently good performance but took over 5 days to run on a GPU making it very computationally expensive.

	Type 1			Type 2	
	Kettle	Microwave	Toaster	Fridge	Dish washer
Mean	48.054268	12.102802	8.988946	37.769161	75.941223
CO	39.111725	31.29039	35.175869	78.651237	118.499252
Hart85	55.819084	39.521473	34.422749	49.529594	72.349953
FHMMExact	42.551697	108.425499	31.962141	53.191868	35.871281
AFHMM	94.324638	39.732384	44.410667	38.301189	88.118752
AFHMM_SAC	35.40107	11.679463	9.260456	31.234304	55.162373
RNN	10.51579	10.105571	3.992678	29.091677	39.969711
Seq2Point	**7.966928**	**9.100262**	3.709853	26.433096	39.615021
Seq2Seq	10.532505	9.86004	**3.500914**	**22.626245**	38.082794
DAE	19.663301	10.351177	10.435557	26.140749	40.004227
WindowGRU	14.534949	10.473172	10.273918	32.706059	**27.361202**

Table 1: MAE: training on house 1 and testing on house 2 from UKDALE

Table 2 looks at the MAE results for type 3 and 4 appliances. NNs perform well on the two type 3 appliances, obtaining very similar error values with RNN being lowest error for the computer and Seq2Point for the Laptop. Type 4 is the traditionally the hardest appliance type to disaggregate but the general performance of both NNs and Non-NNs are relatively good apart from Hart85 which

has a much higher MAE. Combinatorial Optimisation (CO) does well on the external hard disk where it outperforms the other algorithms.

	Type 3		Type 4	
	Computer	Laptop	Broadband router	External hard disk
Mean	0.705697	12.027143	4.259223	12.919944
CO	6.702976	15.789051	5.21025	**8.471948**
Hart85	39.874702	34.712463	36.623878	34.409603
FHMMExact	1.038303	14.908295	5.385968	13.763855
AFHMM	0.734902	13.853347	**1.6183**	13.214809
AFHMM_SAC	0.760197	11.311239	3.818552	12.906783
RNN	**0.704412**	7.575277	4.303452	12.903152
Seq2Point	0.710925	**7.336428**	4.392914	12.909381
Seq2Seq	0.718942	7.907234	4.293953	12.928708
DAE	0.707231	7.404043	4.317731	12.908535
WindowGRU	0.706593	8.374202	4.252302	12.908824

Table 2: MAE: training on house 1 before testing on house 2 from UKDALE

The second trial focused on training and testing on houses in REFIT with Table 3 showing the MAE results. It can be seen from the table and graphs in appendix A.10 - A.12 that the NNs outperformed the more traditional NILM algorithms across all appliance types apart from the microwave where Hart85 obtained a lower MAE result. RNN, Seq2Point and Seq2Seq obtained the smallest MAE for the kettle, dish washer and television respectively where the graphical representation of the predictions in the appendix are almost identical to the ground truth. The FHMM algorithm on the other hand appeared to struggle with disaggregating every appliance when considering the MAE but when visualising the power usage it seems this has been skewed by a number of predictions that are extremely high when there was very little usage by the appliance.

	Type 1		Type 2	Type 3
	Microwave	Kettle	Dish washer	Television
Mean	14.025064	45.364479	105.111977	27.770399
CO	91.896347	161.391434	145.508804	54.863342
Hart85	**3.232651**	23.044636	97.621613	43.298279
FHMMExact	87.562332	118.122269	113.838814	86.968971
RNN	6.660788	**9.915269**	34.334118	17.499918
Seq2Point	5.681811	11.462317	**31.676817**	17.641842
Seq2Seq	5.319926	13.654952	40.82888	**17.040442**
DAE	6.457726	18.261751	48.72316	22.614422
WindowGRU	5.77724	21.115688	42.37888	17.858078

Table 3: MAE: results from training on houses 3, 5 and 9 before testing on house 2 in REFIT

The third trial trained and tested on different time periods for the only house in the DRED data set. Table 4 shows the MAE results for the type 1 appliances and appendix items A.13 - A.23 show the power usage ground truth and predictions for all appliances. Similar to trial 1, the NNs generally outperform the non-NNs with Seq2Point achieving the best result for two of the four appliances. Hart85 had a significantly higher error rate across all type 1 appliances compared to all the algorithms used.

	Type 1			
	Cooker	Microwave	Oven	Toaster
Mean	15.56296	6.200979	3.80784	1.561132
CO	5.668115	9.897218	3.050872	2.38699
Hart85	31.595068	30.031107	29.170872	28.439838
FHMMExact	5.024982	2.840436	2.916052	1.846226
RNN	3.95166	2.580661	**1.725092**	0.704581
Seq2Point	5.31597	**2.128341**	1.96467	**0.641495**
Seq2Seq	4.266376	2.356529	1.864267	0.689489
DAE	4.755477	4.612803	3.201541	1.416595
WindowGRU	**3.921916**	2.359537	2.325821	1.237159

Table 4: MAE: Type 1 results from training on the same house in DRED

The MAE for type 2 appliances are displayed in table 5. Seq2Point outperforms all models across every appliance of this type but the performance of the other NNs holds up well, with DAE having a slightly higher error on three of the four appliances. For a simplistic measure, the mean outperforms the other non-NNs for three of the four appliances as well as achieving a low error

for the fan similar to that of the NNs and only struggling with the fridge. Again Hart85 has a relatively high error value for each appliance apart from the fridge where it stays in line with the other non-NNs.

	Type 2			
	Fan	Electric heating element	Washing machine	Fridge
Mean	2.117838	18.053839	6.114801	41.90239
CO	16.107422	20.517744	9.18088	31.183979
Hart85	28.267057	36.111233	30.546299	31.743103
FHMMExact	15.359023	19.506992	6.220903	29.175024
RNN	2.348713	12.528011	3.525215	27.176342
Seq2Point	**1.372757**	**12.171072**	**2.827319**	**26.974781**
Seq2Seq	2.349674	12.678114	3.16864	27.865934
DAE	2.087653	15.227066	4.328003	28.016762
WindowGRU	1.728683	13.045388	4.221283	27.281672

Table 5: MAE: Type 2 results from training on the same house in DRED

Type 3 and 4 appliance results for trial 3 are shown in Table 6. WindowGRU achieves the best result for the two type 3 appliances and Seq2Seq for the type 4 appliance. As with the other appliance types, the NNs are generally more successful than the non-NNs at disaggregation. Hart85 on the other hand has struggled across all appliance types, having a significantly higher error than the other algorithms.

	Type 3		Type 4
	Laptop	Television	Sockets
Mean	9.850993	5.802611	1.385589
CO	12.85496	14.315316	7.133221
Hart85	29.724546	28.119009	28.798344
FHMMExact	11.074613	12.255075	5.289587
RNN	6.494445	2.541904	1.281272
Seq2Point	9.146392	2.898412	1.185074
Seq2Seq	6.731286	2.429127	**1.071001**
DAE	6.491034	2.311795	1.162264
WindowGRU	**4.95261**	**2.198578**	1.406141

Table 6: MAE: Type 3 and 4 results from training on the same house in DRED

Table 7 considers all the appliance MAE results for trial 4 which focuses on training and testing on houses within REDD. It can also be seen by the power usage predictions and ground truth for the

appliances in appendix items A.24 - A.29 that the NNs do well across all appliances with Seq2Point having the lowest error on the Microwave and Dish washer, WindowGRU on the light and Seq2Seq on the Washer dryer. Seq2Point also had the second lowest error on the light and the fridge as well as the third lowest for the washer dryer showing a good performance all round. The FHMM did well on disaggregating the fridge and light consumption but struggled on the other appliances. Hart85's performance is poor across all appliances apart from on the Fridge where it keeps in line with other methods, even outperforming DAE and WindowGRU.

	Type 1		Type 2			Type 4
	Light	Microwave	Fridge	Dish washer	Washer dryer	Sockets
Mean	29.255951	19.673189	76.965294	24.270401	46.072075	**11.351704**
CO	53.950687	50.568249	48.66782	33.82177	19.802473	21.937899
Hart85	74.894432	79.643578	30.742414	78.480011	73.199211	75.013527
FHMMExact	26.281292	23.511032	**27.518848**	34.246582	22.661018	55.166756
RNN	26.602736	12.772479	32.674732	15.040881	2.271451	12.432703
Seq2Point	24.916204	**9.862741**	27.799969	**6.423064**	2.444851	17.190519
Seq2Seq	25.643454	12.066082	28.91538	11.687813	**1.678894**	16.098833
DAE	26.92551	22.59136	44.3811	12.912072	17.042856	17.789526
WindowGRU	**20.614273**	10.241246	31.034838	12.909094	4.277078	12.151745

Table 7: MAE: training on Houses 1, 3 and 5 before testing on House 2 from REDD

The results for trial 5, which involved training and testing on houses within SMART, are shown in table 8. Hart85 is not included in the table as it did not produce any results because it could not find steady states within the SMART data set when using the NILMTK package. Surprisingly, CO and FHMM obtain exactly the same MAE for all appliances and obtain the lowest MAE for all type 2 and 4 appliances. Seq2Seq had the lowest MAE for the type 1 appliance, the fridge but it is worth noting that all the algorithms achieved a relatively low MAE when evaluated against data contained in house 3. When plotting the ground truth and the predictions for appliances in SMART which are shown in appendix A.30 - A.34 it can be observed that the power usage of the appliances is quite low in comparison to other data sets. When looking at the model predictions across each time step it can be seen that they are quite similar, providing a constant line across the graph.

	Type 1	Type 2		Type 4	
	Fridge	Dish washer	Washing machine	Sockets	Boiler
Mean	0.050432	0.03643	0.004996	0.079309	0.12655
CO	0.061556	**0.02868**	**0.002599**	**0.033402**	**0.100836**
FHMMExact	0.061556	**0.02868**	**0.002599**	**0.033402**	**0.100836**
RNN	0.050531	0.03644	0.004959	0.078727	0.127416
Seq2Point	0.050429	0.036802	0.005068	0.07933	0.126766
Seq2Seq	**0.050228**	0.036568	0.004965	0.079095	0.126537
DAE	0.050436	0.036113	0.00491	0.079459	0.126447
WindowGRU	0.05032	0.067639	0.002757	0.069678	0.130001

Table 8: MAE: training on house 1 and testing on house 3 from SMART

Table 9 looks at the MAE results and appendix items B.1 - B.3 contain the predictions and ground truth for the first trial within stage 2 which experiments with training and testing on two different data sets within the same region. Specifically, training on REDD and testing on SMART, both USA data sets. Combinatorial Optimisation (CO) performs extremely well on this experiment obtaining an extremely low MAE for all three appliances, way below that of other methods apart from on the Dish washer where it achieves a joint MAE value with the FHMM, RNN and Seq2Point models. Notably WindowGRU has an extremely large MAE value for the dish washer almost ten times more than the Mean algorithm. When compared to the results obtained from training and testing on REDD shown in table 7 the results seem to have improved, potentially indicating that disaggregating appliances within SMART is easier and may be due to the low power usage of appliances.

	Type 1	Type 2	
	Microwave	Fridge	Dish washer
Mean	22.300686	55.000107	25.801783
CO	**0.011684**	**0.072929**	**0.015357**
FHMMExact	3.988316	6.927072	**0.015357**
RNN	6.511631	10.085876	**0.015357**
Seq2Point	4.081916	5.847533	**0.015357**
Seq2Seq	4.381122	11.101905	0.39441
DAE	1.815905	6.939919	0.408097
WindowGRU	3.970791	8.851661	204.599152

Table 9: MAE: training on REDD and testing on SMART

The second trial in stage 2 looks at training on UKDALE before testing on REFIT with the MAE results shown in table 9 and algorithm predictions and ground truth power usage in appendices B.4 - B.6. For the microwave the algorithms had an average performance with Hart85 achieving the lowest MAE compared to its poor performance across previous trials. All of the algorithms struggled with the dish washer, especially CO, DAE and FHMM with RNN obtaining the lowest MAE value of 52.85 which is much higher than the best results from any of the previous trials. FHMM and CO also have relatively high MAE values for the fridge, 122.08 and 128.05 respectively, compared to the other algorithms which generally performed quite well such as Seq2Point which achieved the lowest MAE of 8.24.

	Type 1	Type 2	
	Microwave	Fridge	Dish washer
Mean	43.307487	9.503291	95.715828
CO	35.162971	128.049286	162.584076
Hart85	**28.736259**	37.883297	90.16024
FHMMExact	53.54866	122.077873	111.474365
RNN	30.344385	9.644649	**52.854412**
Seq2Point	36.441536	11.444447	73.616371
Seq2Seq	36.339844	**8.240096**	91.699226
DAE	31.890347	18.533953	132.21077
WindowGRU	30.110352	14.396706	60.865955

Table 10: MAE: training on UKDALE and testing on REFIT

Appendices C.1 - C.3 contain the appliance power usage predictions and ground truth whilst table 10 shows the MAE results from trial 1 in stage 3 where transfer learning is considering by training UKDALE and REDD together before testing on REFIT. The NN algorithms all perform well on the microwave compared to CO and FHMM which have large MAE values but it is seen that simply using the mean obtains a lower error rate. Of the type 2 appliances the dish washer again is difficult for the algorithms to correctly disaggregate, especially for the non-NNs which across the board have large MAE values such as CO with an MAE of 193.1 in comparison to the NNs where Seq2Point obtained the lowest MAE value of 47.7. Performance for the fridge was average with the NNs again achieving a lower MAE apart from Hart85 which surprisingly obtains the lowest error rate.

	Type 1	Type 2	
	Microwave	Fridge	Dish washer
Mean	**11.758831**	43.593956	93.781693
CO	110.312622	59.145321	193.099945
Hart85	37.883297	**28.736259**	90.16024
FHMMExact	126.686775	59.996429	107.018822
RNN	16.164988	33.739834	60.148235
Seq2Point	19.068787	34.822243	**47.744091**
Seq2Seq	14.470087	33.666904	50.893326
DAE	26.908604	32.109875	63.166573
WindowGRU	15.662368	35.771244	54.360443

Table 11: MAE: training on UKDALE and REDD and testing on REFIT

Table 11 is the second trial in stage 3 and the final experiment of the study, testing the joint training performance of UKDALE and REDD on SMART. Similar to Table 7 where REDD is trained and tested on SMART, CO performs extremely well and achieves the lowest MAE across all the appliances. It can also be seen from the predictions and ground truth in appendix items C.4 - C.6 that FHMM seems to achieve a relatively good performance across appliances being third lowest for MAE on the fridge and dish washer and only just fourth for the microwave. For the NNs there is a mixed bag, Seq2Point and Seq2Seq did particularly well with the microwave and dishwasher but struggled on the fridge whereas RNN obtained low MAE values for the dish washer and fridge but struggled when disaggregating the microwave.

	Type 1	Type 2	
	Microwave	Fridge	Dish washer
Mean	8.671385	46.550751	39.241394
CO	**0.011684**	**0.077824**	**0.015357**
FHMMExact	0.988922	9.927069	0.984649
RNN	3.847263	7.762519	0.602117
Seq2Point	0.9838	23.71557	1.146265
Seq2Seq	0.676076	34.022991	1.011367
DAE	2.594135	12.988195	3.921383
WindowGRU	3.88848	11.943854	2.64832

Table 12: MAE: training on UKDALE and REDD and testing on SMART

5 Discussion

The aim of this study was to compare the performance of the variations of neural networks that are currently state-of-the-art for NILM using traditional non-NN methods such as FHMMs as a performance baseline. This aim was extended further to that of cross-domain transfer learning to see whether training on data sets from different regions would improve the generalisation performance of the models compared to the baseline performance of training and testing on data sets within the same region. The results from training and testing on houses within the same data set (stage 1) found that NNs achieved a much better performance than non-NNs across every trial consistently obtaining low MAE values. Seq2Point and Seq2Seq were notable algorithms across each trial generalising extremely well on unseen houses and across most appliance types.

The results from the SMART data set were markedly different from the rest of the trials due to how well the models performed and the very low MAE. After considering the plots of power usage in appendix A.30 - A.34 it was noted that the usage of appliances in SMART tend to be quite low and this may be overstating the performance of the algorithms.

When evaluating the generalisation ability of the algorithms on different data sets within the same region in stage 2, there was a mixed performance on the USA region with non-NNs such as CO and FHMM outperforming most or all of the NNs however the MAE for every model apart from WindowGRU and the Mean were relatively low. This potentially could have been caused by the fact that REDD only contained 28 days worth of data to train the neural networks affecting their performance.

The results from the UK region had higher MAE across all appliances but the NNs generally outperformed the non-NNs apart from DAE which had difficulty with the dish washer. For the cross-domain transfer learning in stage 3 where both REDD and UKDALE were used for training the models, mixed results were found when tested on SMART in comparison to the regional performance of REDD on SMART in stage 2. It seemed the transfer learning significantly improved the performance of all the models apart from DAE when disaggregating the microwave but significantly decreased the performance for all but RNN and the Mean for the fridge and all but WindowGRU

for the dish washer. In fact, the joint training improved the WindowGRU MAE remarkably from 204.6 to 2.65.

When testing the transfer learning model on REFIT and comparing against the regional performance in stage 2 similar results were found for the microwave with lower error values for all NNs however the non-NNs, except the mean suffered a performance drop. The opposite happened when disaggregating the fridge as all the NNs and the mean saw a performance increase whilst the rest of the non-NNs saw a drop. In contrast to the transfer learning models poor results when disaggregating the dish washer on REDD, it was found to have a lower error value for all algorithms, bar CO and RNN, when disaggregating using REFIT.

Computational power was a limitation of this study as neural networks and variations on HMMMs can be extremely complex and require large amounts of data making it difficult for traditional Computer Processing Units (CPUs) to handle. Even with the use of Graphics Processing Units (GPUs) the neural networks took almost a day to process and certain models such as AFHMM and AFHMM_SAC took 5 or more days often crashing the GPU before completion which is why results for those two models are only available for UKDALE. These two algorithms were shown to provide a good performance when training and testing on houses within UKDALE which means they could possibly rival NNs in performance had the model processing completed. Future work could explore these models further if they had access to multiple GPUs that were able to handle computational expense, however, if these models were to be used in a practical setting the computation time would be a major factor in determining their success.

Whilst this study was successful at measuring the performance of different state-of-the-art algorithms it could be expanded in various ways.

Only certain houses were selected due to limitations on shared appliances, the computation times from large amounts of data in NILM data sets and the algorithms being tested, for example, REFIT had 19 houses available but 3 were used in this study for training which took more than a day to process.

A large number of data sets were explored in this study where the ones selected were based on

download accessibility, whether it could be converted into a h5 file type required for the NILMTK package and could be trained on with no package errors. Data sets such as Dataport would have been extremely useful had access been granted upon request and improvements to NILMTK may have allowed IAWE to be used for training and testing without error. Also, as smart meters continue to be rolled out and NILM continues to grow as a research area there may be more investment into gathering a wide range of data to be used.

Furthermore, only two data sets were combined for training the transfer learning model where further work could be conducted to measure whether using three or more data sets would improve the generalisation ability by exposing the model to different patterns of appliance signals.

Because this study only used NILMTK as a way of standardising model selection and evaluation methods it suffered from only being able to select appliances that were in both the training and test data sets. Further work to NILMTK could allow for investigation into using a wider variety of appliances for training that may improve the generalisation of the model especially considering cross-domain transfer learning where appliances differ from Europe and USA. This would also allow for type 3 and 4 appliances to be considered in the transfer learning model as they aren't commonly measured or shared across data sets which meant they had to be excluded in this study.

Artificial aggregates weren't included in this study so that the algorithms could be trained and evaluated based on noisy data representative of real life. Further work could investigate how the models compare to those trained on the artificial aggregates, similar to work by Batra et al [23] but expanded to see whether this combined with transfer learning enhances the models generalisation capability

6 Conclusion

In conclusion, this study considered the performance of neural networks in comparison to more traditional methods using various data sets and appliances for non-intrusive load monitoring. The results of the experiments consistently found that NNs achieve a state-of-the-art performance and much lower mean average errors to that of traditional methods when applied to NILM. Over the various experiments, Seq2Point and Seq2Seq, both variations of NNs, stood out as quite successful

algorithms, often achieving the lowest error value or a similar performance across different data sets and appliance types. The generalisation ability of the algorithms across similar regions (USA to USA and UK to UK) was also evaluated. The results showed a mixed picture with most NNs in the USA region achieving a low mean average error but being outperformed by the traditional methods. The UK region was also a mixed picture where the NNs mainly outperformed but the overall mean average error was relatively high in comparison to the USA region.

Cross-domain transfer learning was also considered in this study, aiming to see whether there would be a marked improvement to the performance of algorithms if they were trained on two data sets; one from the UK and one from the USA. The results again were mixed with NNs performing well on SMART but not as well as non-NNs and the opposite happening when testing on REFIT but it shows further promise for the area of cross-domain transfer learning on NILM. Further research areas were also identified such as training on more than two data sets, utilising a wider range of appliances and data sets from different regions.

References

[1] Energy roadmap 2050 Energy. 10.2833/10759. Available from: `http://europa.eu`.

[2] Rolnick D, Donti PL, Kaack LH, Kochanski K, Lacoste A, Sankaran K, Ross AS, Milojevic-Dupont N, Jaques N, Waldman-Brown A, and others . Tackling climate change with machine learning. *arXiv preprint arXiv:1906.05433*. 2019.

[3] Pérez-Lombard L, Ortiz J, and Pout C. A review on buildings energy consumption information. *Energy and buildings*. 2008, 40(3):394–398.

[4] Kelly J and Knottenbelt W. The uk-dale dataset, domestic appliance-level electricity demand and whole-house demand from five uk homes. *Scientific data*. 2015, 2(1):1–14.

[5] Kolter JZ and Johnson MJ. Redd: A public data set for energy disaggregation research. In: *Workshop on data mining applications in sustainability (SIGKDD), San Diego, CA*, volume 25, p. 59–62, 2011.

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

[7] Gopinath R, Kumar M, Joshua CPC, and Srinivas K. Energy management using non-intrusive load monitoring techniques-state-of-the-art and future research directions. *Sustainable Cities and Society*. 2020, p. 102411.

[8] Gonçalves H, Ocneanu A, Bergés M, and Fan R. Unsupervised disaggregation of appliances using aggregated consumption data. In: *The 1st KDD workshop on data mining applications in sustainability (SustKDD)*, 2011.

[9] Kolter JZ and Jaakkola T. Approximate inference in additive factorial hmms with application to energy disaggregation. In: *Artificial intelligence and statistics*, p. 1472–1482. PMLR, 2012.

[10] Parson O, Ghosh S, Weal M, and Rogers A. An unsupervised training method for non-intrusive appliance load monitoring. *Artificial Intelligence*. 2014, 217:1–19.

[11] Liao J, Elafoudi G, Stankovic L, and Stankovic V. Non-intrusive appliance load monitoring using low-resolution smart meter data. In: *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, p. 535–540. IEEE, 2014.

[12] Cominola A, Giuliani M, Piga D, Castelletti A, and Rizzoli AE. A hybrid signature-based iterative disaggregation algorithm for non-intrusive load monitoring. *Applied energy*. 2017, 185:331–344.

[13] He K, Stankovic L, Liao J, and Stankovic V. Non-intrusive load disaggregation using graph signal processing. *IEEE Transactions on Smart Grid*. 2016, 9(3):1739–1747.

[14] Zhao B, Stankovic L, and Stankovic V. On a training-less solution for non-intrusive appliance load monitoring using graph signal processing. *IEEE Access*. 2016, 4:1784–1799.

[15] Altrabalsi H, Stankovic V, Liao J, and Stankovic L. Low-complexity energy disaggregation using appliance load modelling. *Aims Energy*. 2016, 4(1):884–905.

[16] Kelly J and Knottenbelt W. Neural nilm: Deep neural networks applied to energy disaggregation. In: *Proceedings of the 2nd ACM international conference on embedded systems for energy-efficient built environments*, p. 55–64, 2015.

[17] Linh NV and Arboleya P. Deep learning application to non-intrusive load monitoring. In: *2019 IEEE Milan PowerTech*, p. 1–5. IEEE, 2019.

[18] Bonfigli R, Felicetti A, Principi E, Fagiani M, Squartini S, and Piazza F. Denoising autoencoders for non-intrusive load monitoring: improvements and comparative evaluation. *Energy and Buildings*. 2018, 158:1461–1474.

[19] D'Incecco M, Squartini S, and Zhong M. Transfer learning for non-intrusive load monitoring. *IEEE Transactions on Smart Grid*. 2019, 11(2):1419–1429.

[20] Pereira L and Nunes N. Performance evaluation in non-intrusive load monitoring: Datasets, metrics, and tools—a review. *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*. 2018, 8(6):e1265.

[21] Klemenjak C, Faustine A, Makonin S, and Elmenreich W. On metrics to assess the transferability of machine learning models in non-intrusive load monitoring. *arXiv preprint arXiv:1912.06200*. 2019.

[22] Batra N, Kelly J, Parson O, Dutta H, Knottenbelt W, Rogers A, Singh A, and Srivastava M. Nilmtk: An open source toolkit for non-intrusive load monitoring. In: *Proceedings of the 5th international conference on Future energy systems*, p. 265–276, 2014.

[23] Batra N, Kukunuri R, Pandey A, Malakar R, Kumar R, Krystalakos O, Zhong M, Meira P, and Parson O. Towards reproducible state-of-the-art energy disaggregation. In: *Proceedings of the 6th ACM international conference on systems for energy-efficient buildings, cities, and transportation*, p. 193–202, 2019.

[24] Batra N, Jia Y, Wang H, and Whitehouse K. Transferring decomposed tensors for scalable energy breakdown across regions. In: *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[25] Humala B, Nambi ASU, and Prasad VR. Universalnilm: A semi-supervised energy disaggregation framework using general appliance models. In: *Proceedings of the Ninth International Conference on Future Energy Systems*, p. 223–229, 2018.

[26] Murray D, Stankovic L, Stankovic V, Lulic S, and Sladojevic S. Transferability of neural network approaches for low-rate energy disaggregation. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 8330–8334. IEEE, 2019.

[27] IAWE. Available from: `https://iawe.github.io/`.

[28] Murray D, Stankovic L, and Stankovic V. An electrical load measurements dataset of united kingdom households from a two-year longitudinal study. *Scientific data*. 2017, 4(1):1–12.

[29] Uttama Nambi AS, Reyes Lua A, and Prasad VR. Loced: Location-aware energy disaggregation framework. In: *Proceedings of the 2nd acm international conference on embedded systems for energy-efficient built environments*, p. 45–54, 2015.

[30] Barker S, Mishra A, Irwin D, Cecchet E, Shenoy P, Albrecht J, and others . Smart*: An open data set and tools for enabling research in sustainable homes. *SustKDD, August*. 2012, 111 (112):108.

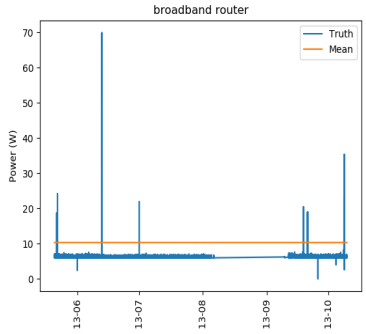
[31] Dataport. Available from: `https://www.pecanstreet.org/dataport/`.

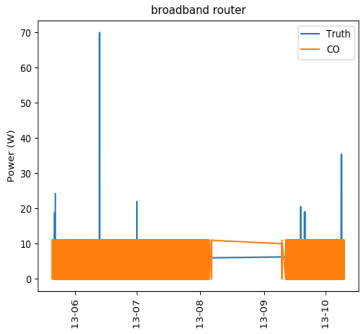
[32] Zhang C, Zhong M, Wang Z, Goddard N, and Sutton C. Sequence-to-point learning with neural networks for non-intrusive load monitoring. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

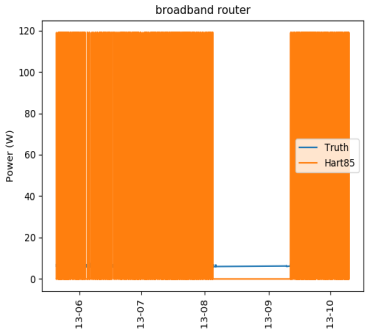
[33] Krystalakos O, Nalmpantis C, and Vrakas D. Sliding window approach for online energy disaggregation using artificial neural networks. In: *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, p. 1–6, 2018.

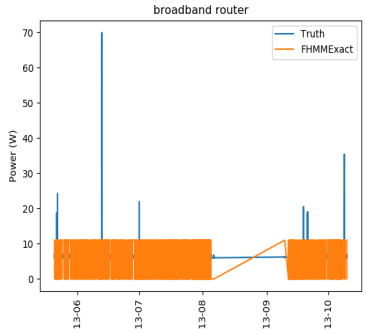
A Stage 1: Appliance graphs

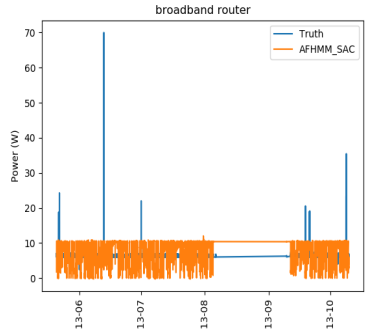
A.1 UKDALE - Broadband

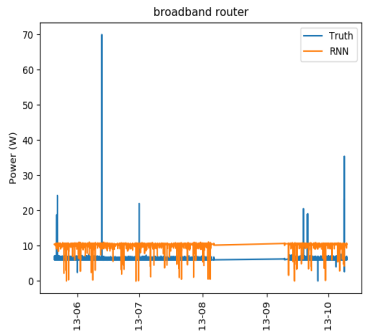


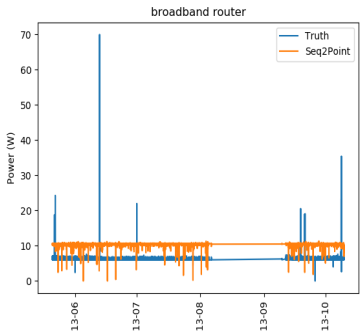


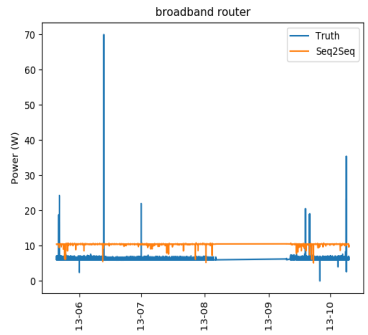


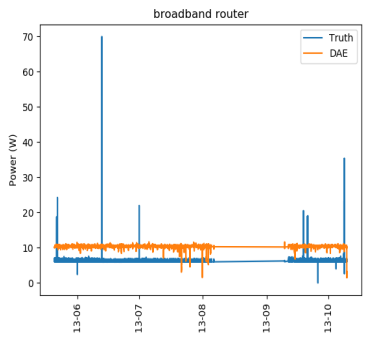


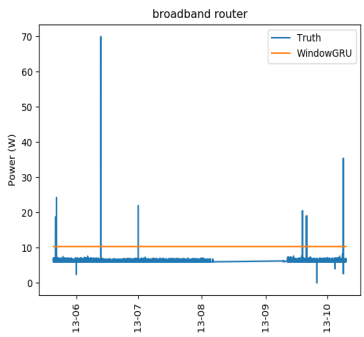




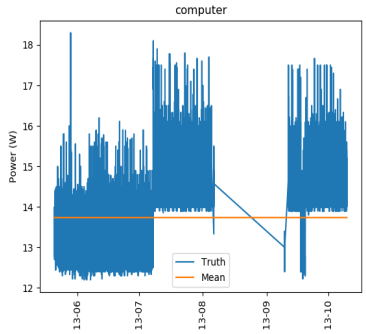


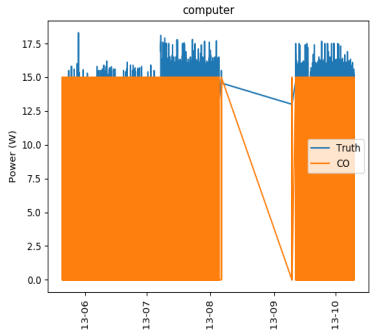


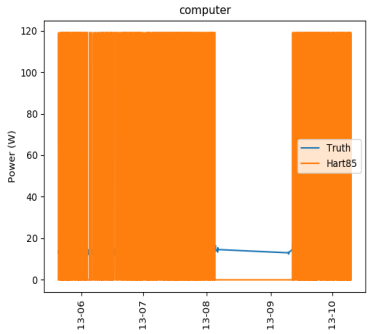


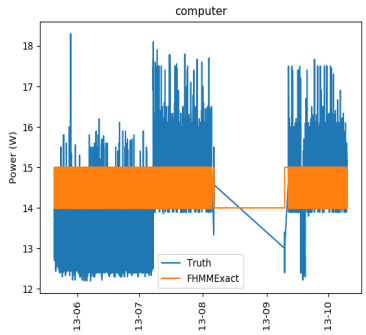


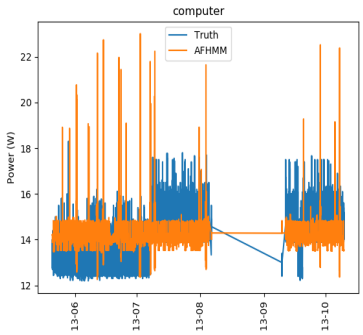
A.2 UKDALE - Computer

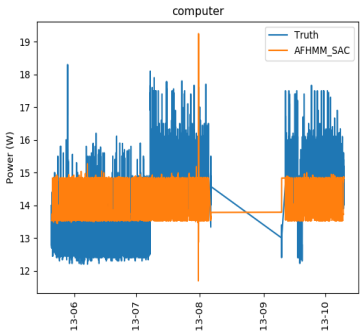


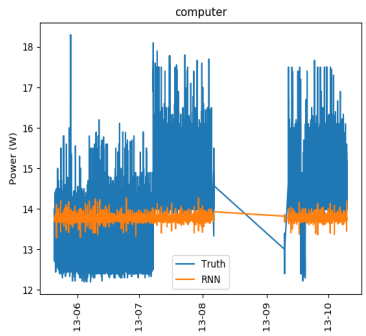


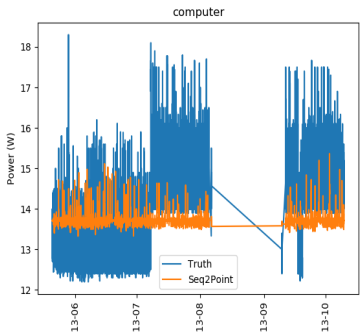


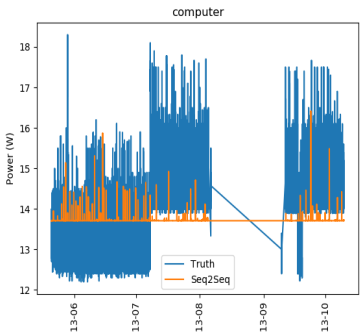


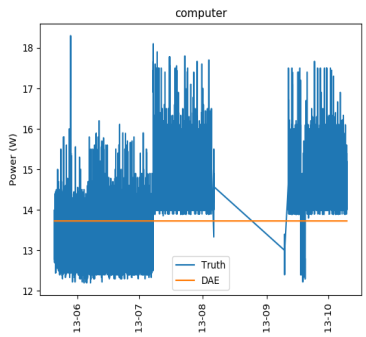


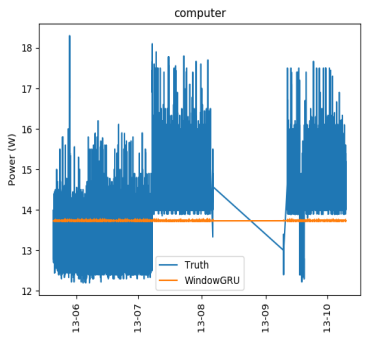




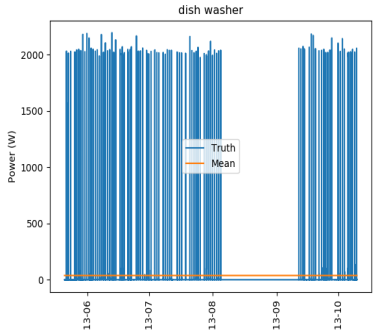


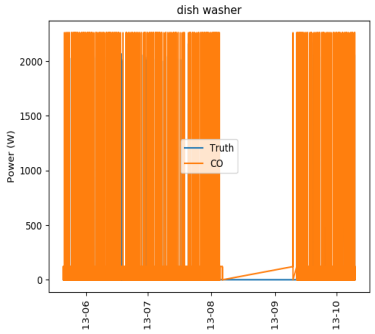


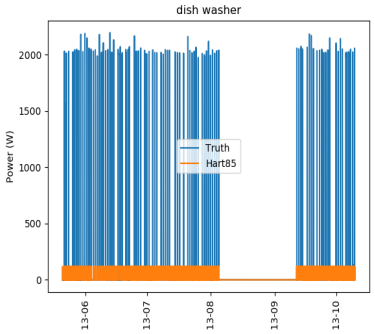


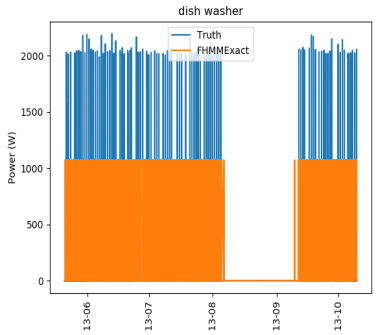


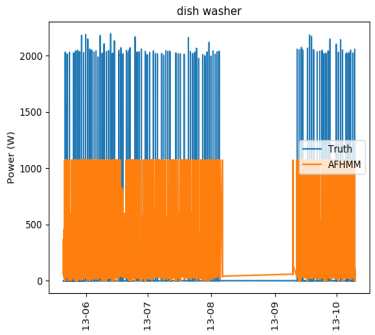
A.3 UKDALE - Dishwasher

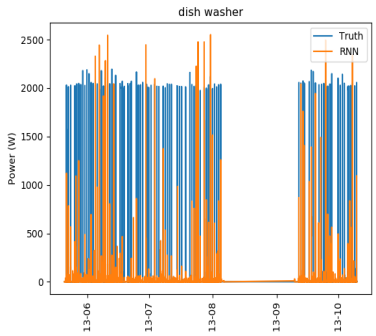


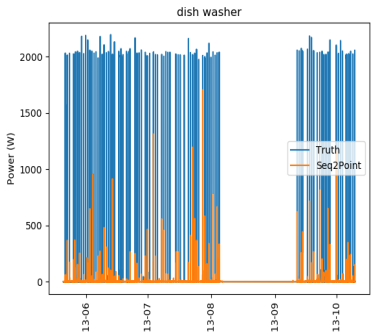


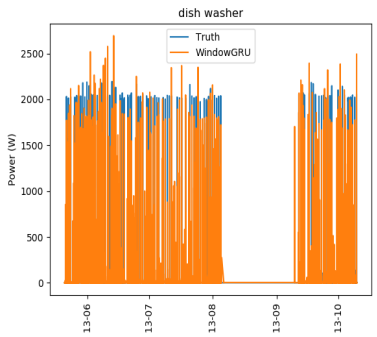




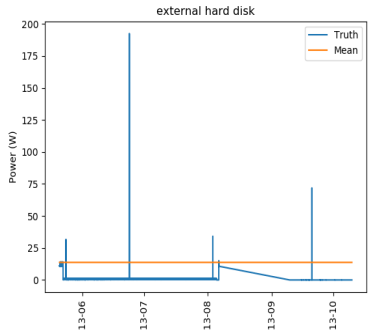


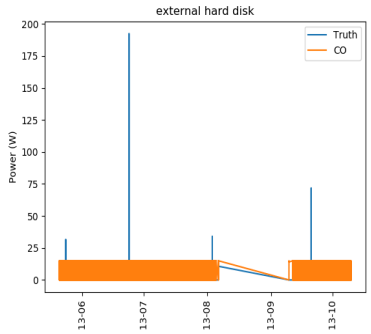


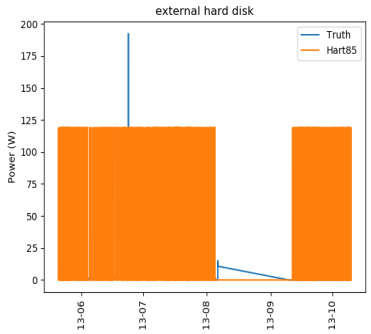


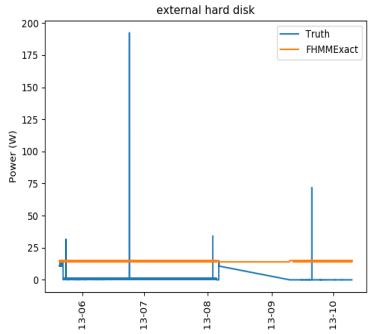


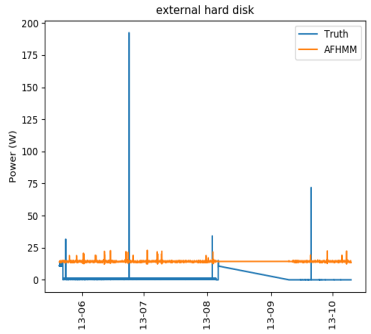
A.4 UKDALE - External hard disk

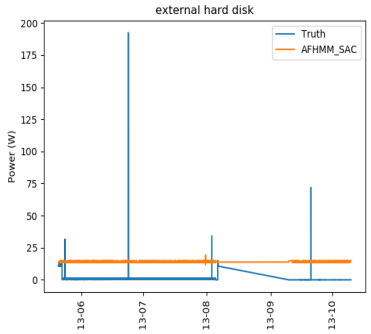


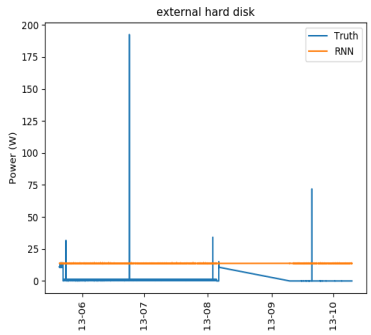


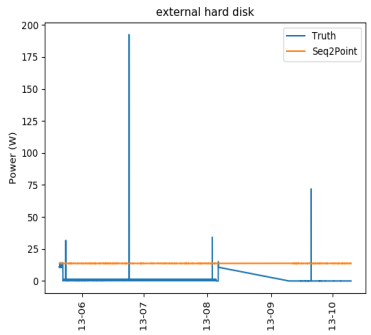


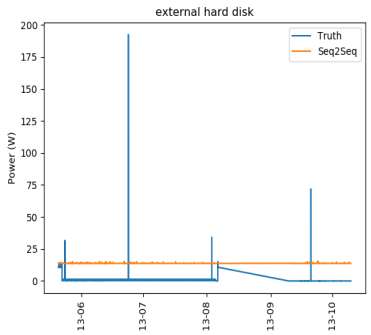


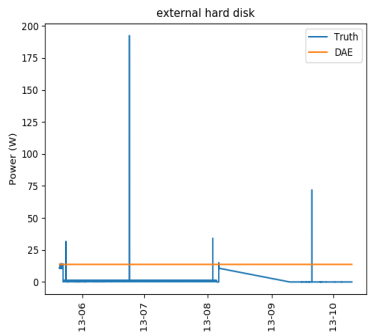


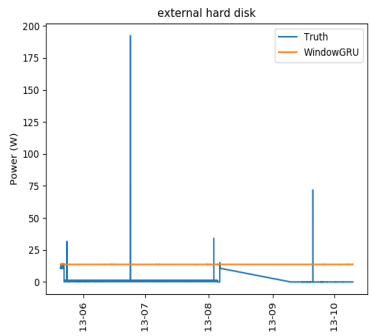




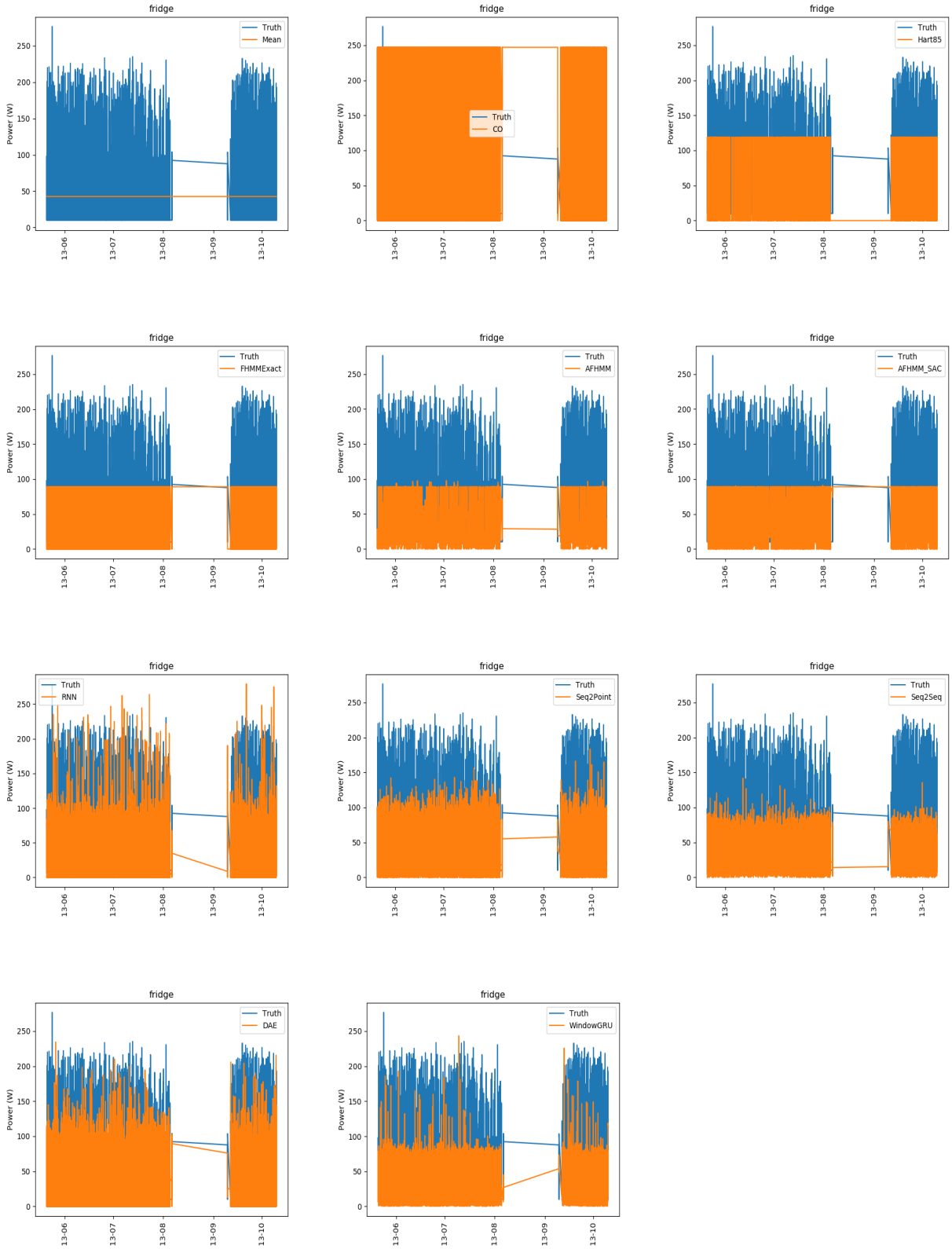




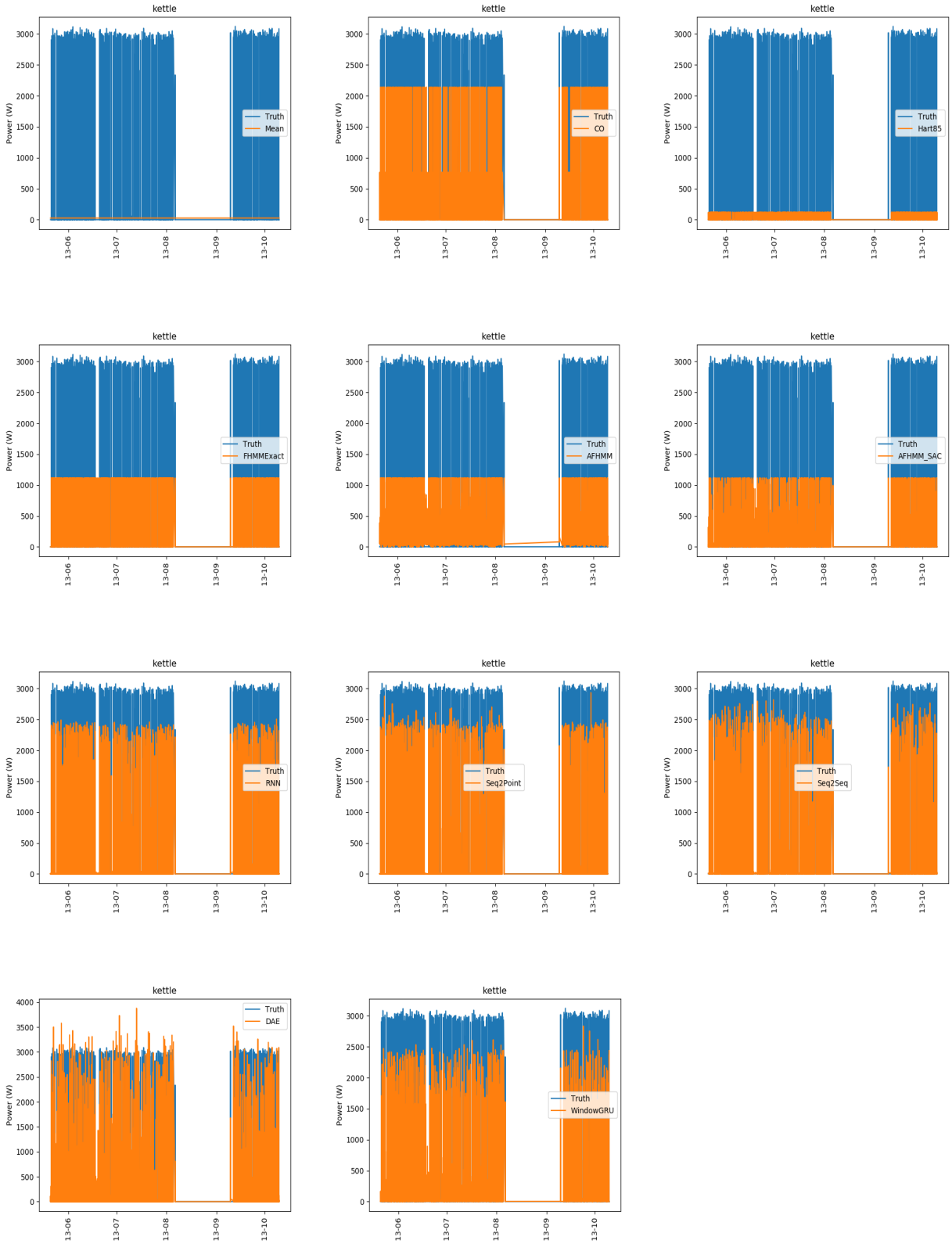




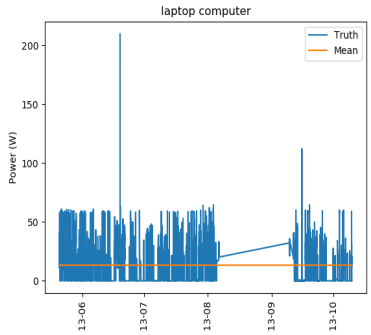
A.5 UKDALE - Fridge

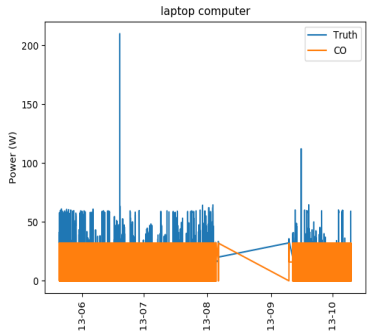


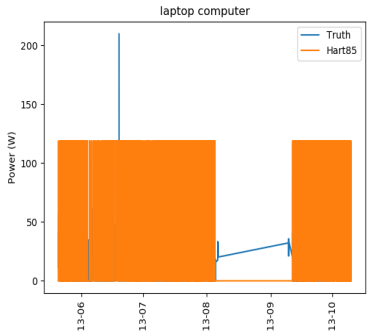
A.6 UKDALE - Kettle

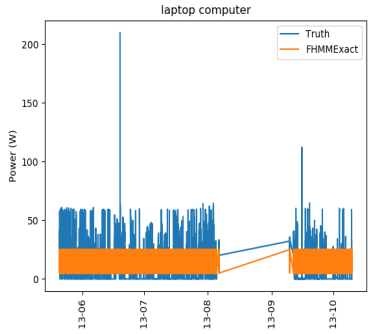


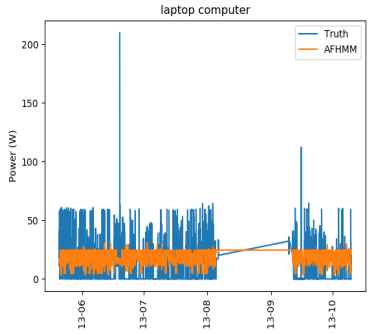
A.7 UKDALE - Laptop

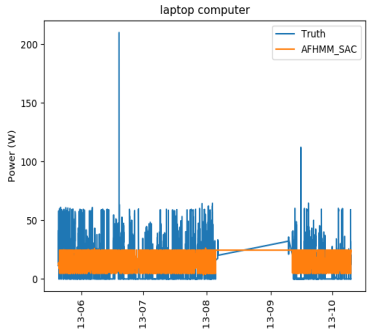


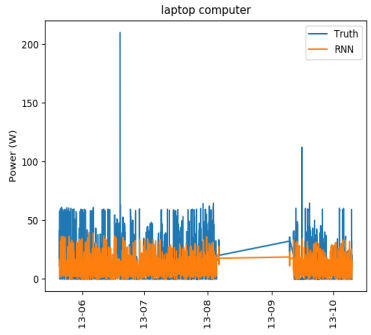


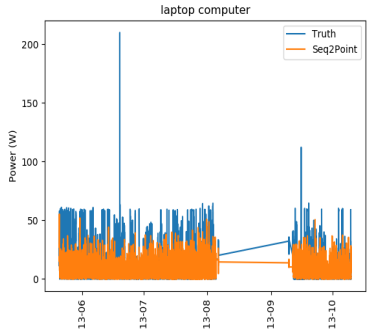


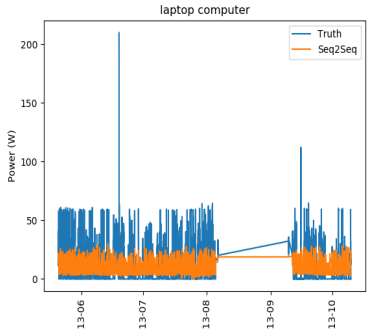


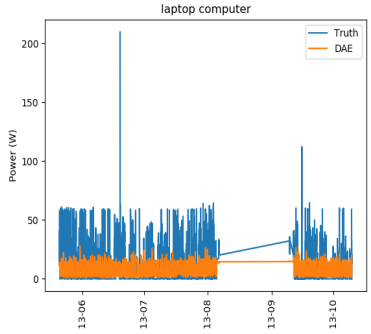


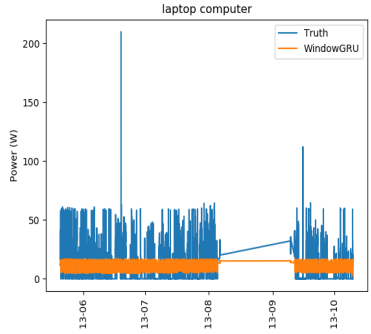




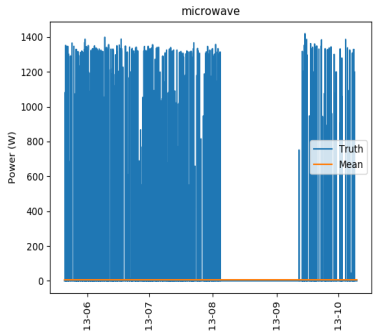


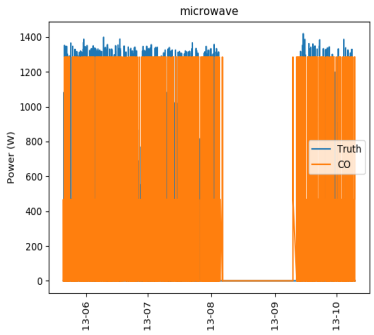


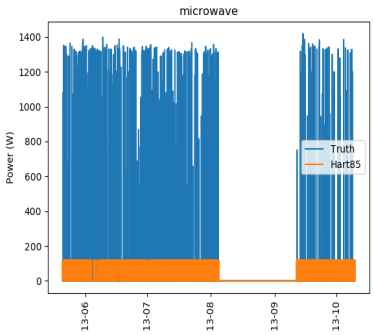


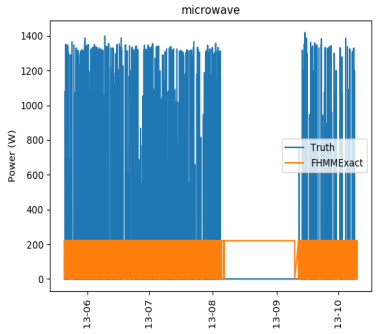


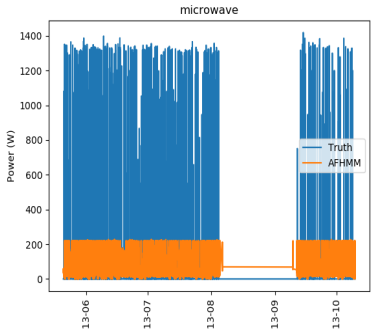
A.8 UKDALE - Microwave

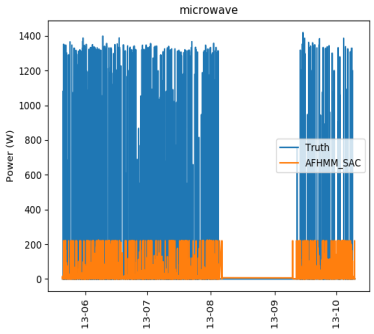


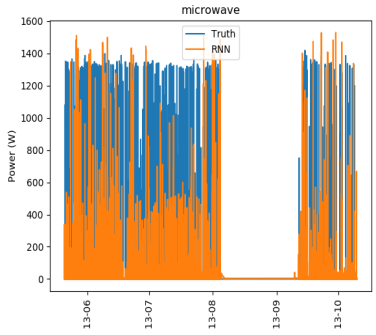


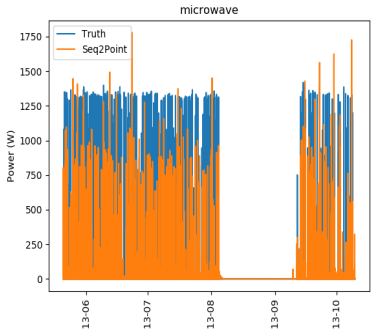


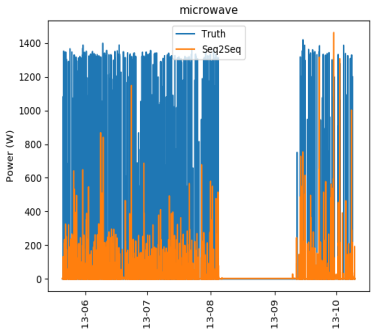


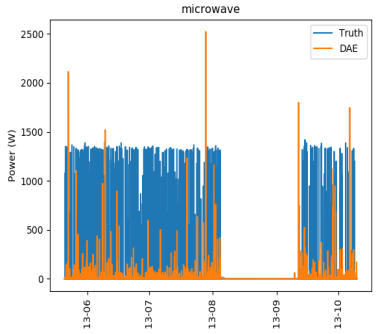


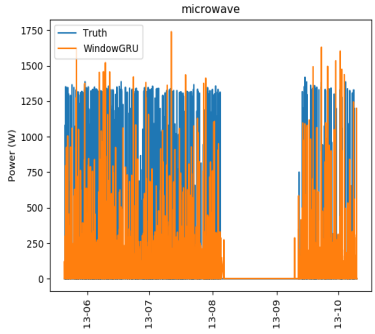




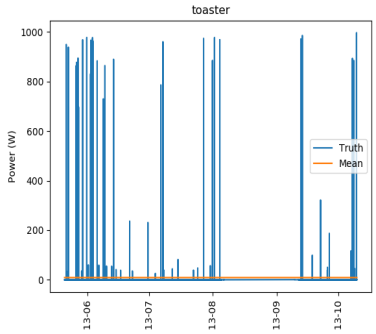


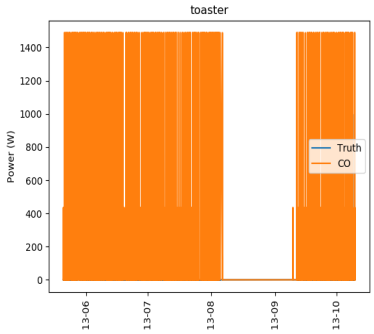


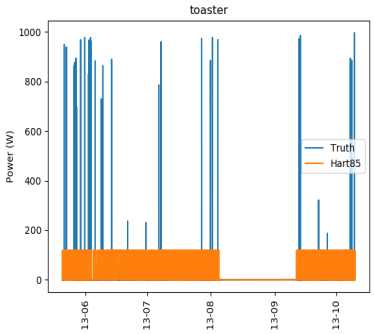


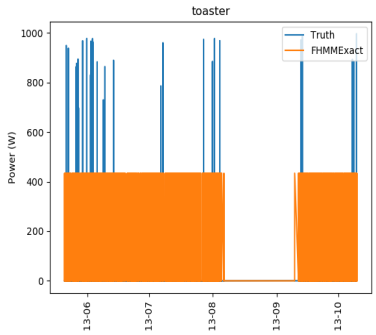


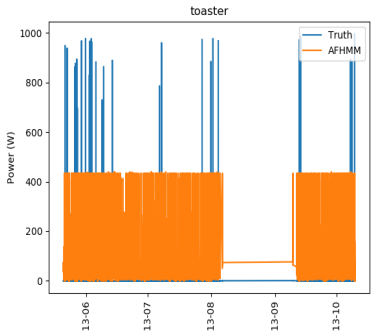
A.9 UKDALE - Toaster

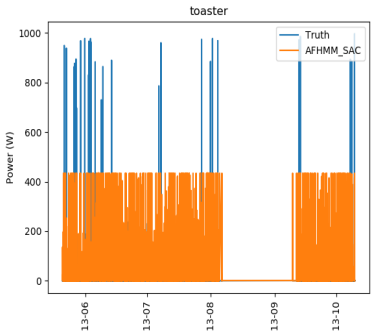


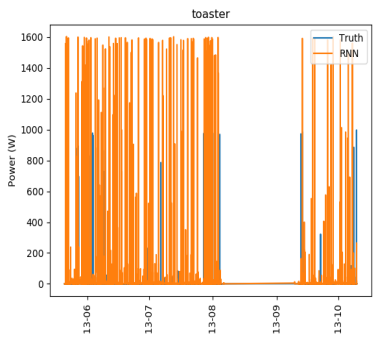


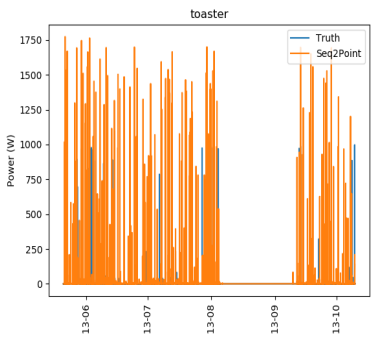


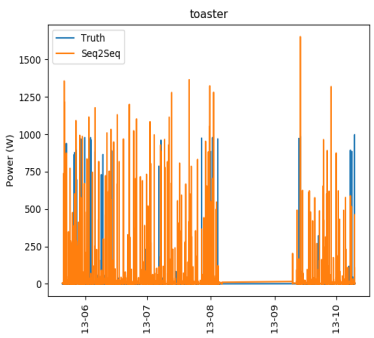


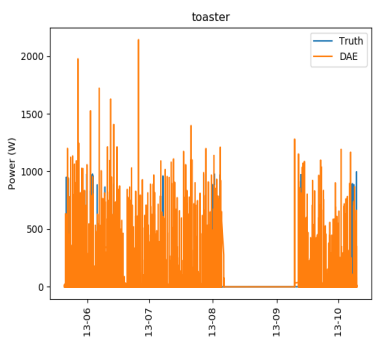


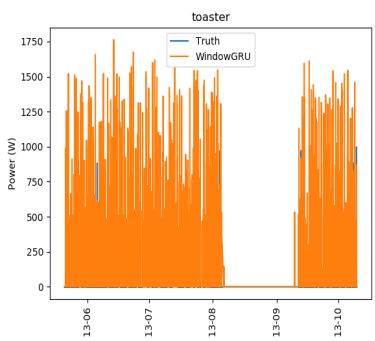


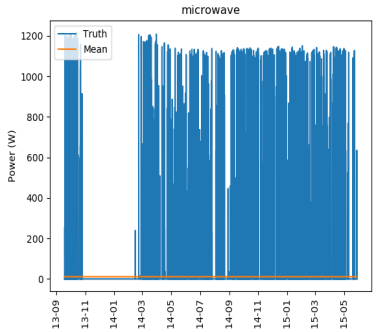


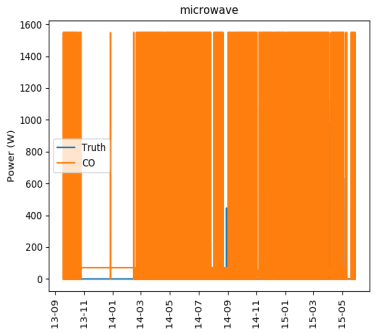


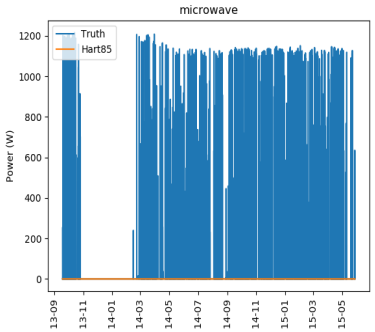


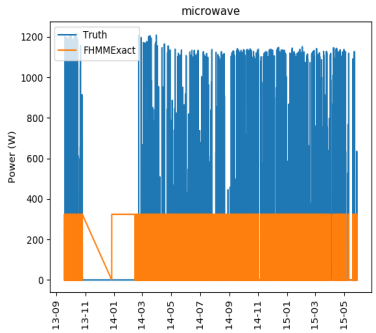


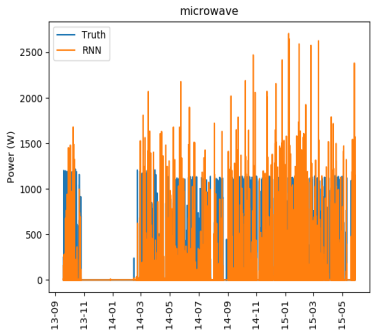


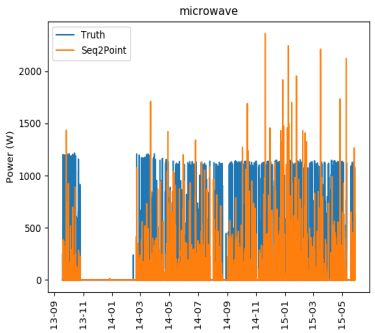

microwave
Truth
Mean
Power (W)
0
200
400
600
800
1000
1200
13-09
13-11
14-01
14-03
14-05
14-07
14-09
14-11
15-01
15-03
15-05

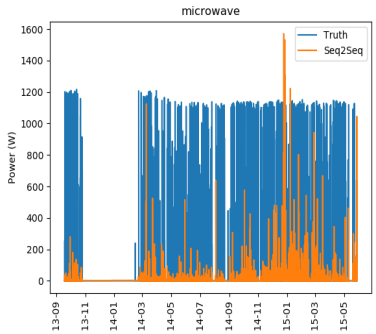

microwave
Truth
CO
Power (W)
0
200
400
600
800
1000
1200
1400
1600
13-09
13-11
14-01
14-03
14-05
14-07
14-09
14-11
15-01
15-03
15-05

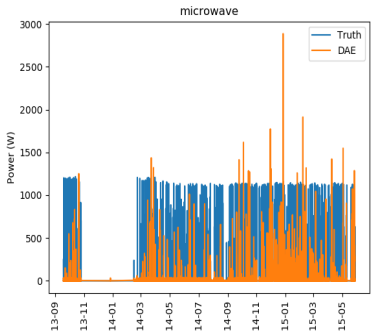

microwave
Truth
Hart85
Power (W)
0
200
400
600
800
1000
1200
13-09
13-11
14-01
14-03
14-05
14-07
14-09
14-11
15-01
15-03
15-05

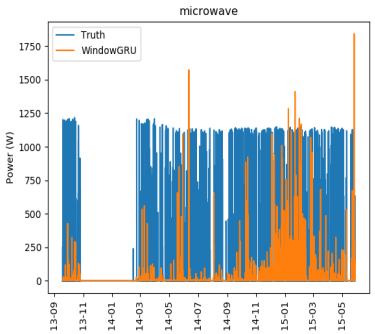

microwave
Truth
FHMMExact
Power (W)
0
200
400
600
800
1000
1200
13-09
13-11
14-01
14-03
14-05
14-07
14-09
14-11
15-01
15-03
15-05


microwave
Truth
RNN
Power (W)
0
500
1000
1500
2000
2500
13-09
13-11
14-01
14-03
14-05
14-07
14-09
14-11
15-01
15-03
15-05

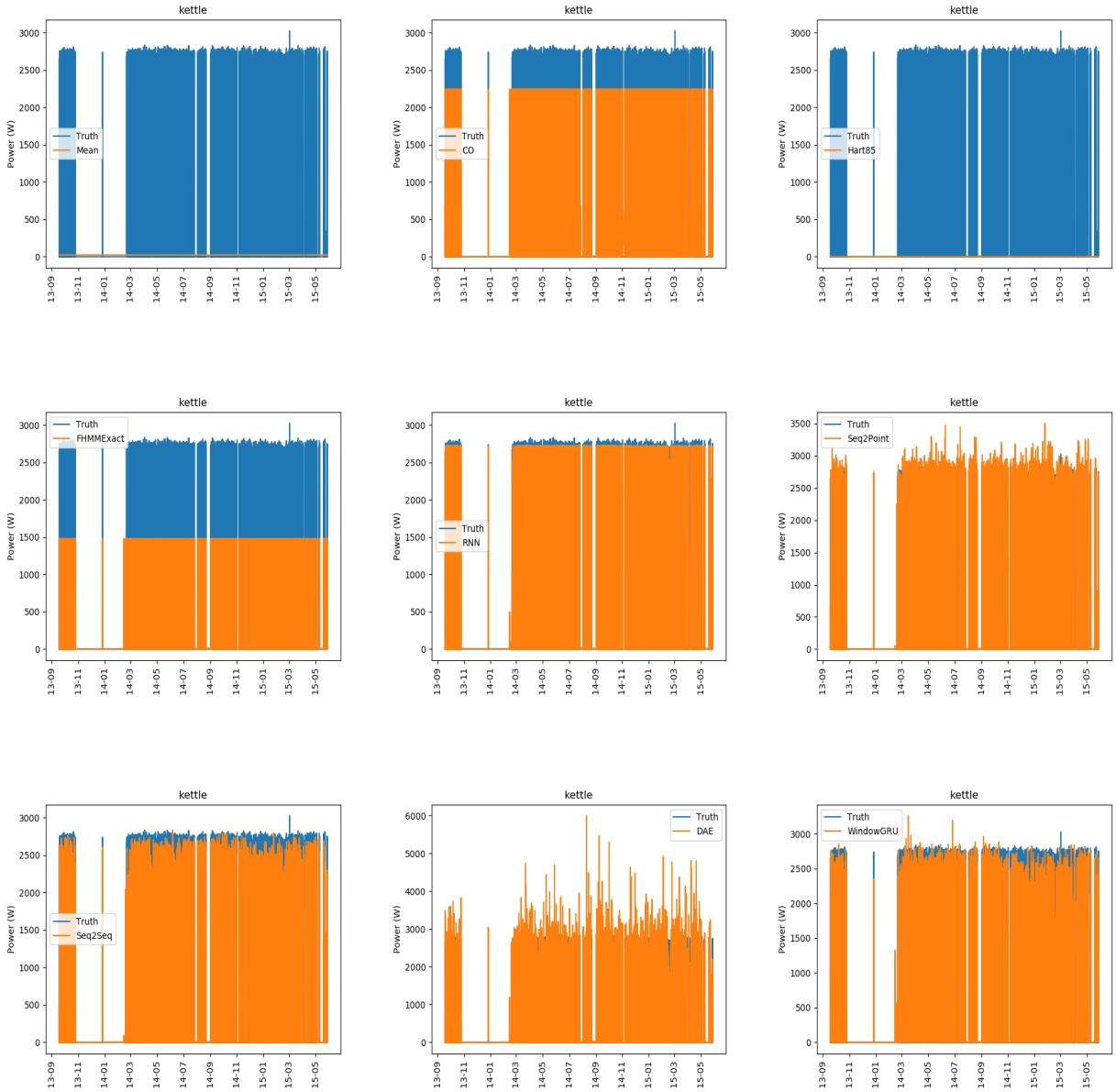

microwave
Truth
Seq2Point
Power (W)
0
500
1000
1500
2000
13-09
13-11
14-01
14-03
14-05
14-07
14-09
14-11
15-01
15-03
15-05


microwave
Truth
Seq2Seq
Power (W)
0
200
400
600
800
1000
1200
1400
1600
13-09
13-11
14-01
14-03
14-05
14-07
14-09
14-11
15-01
15-03
15-05

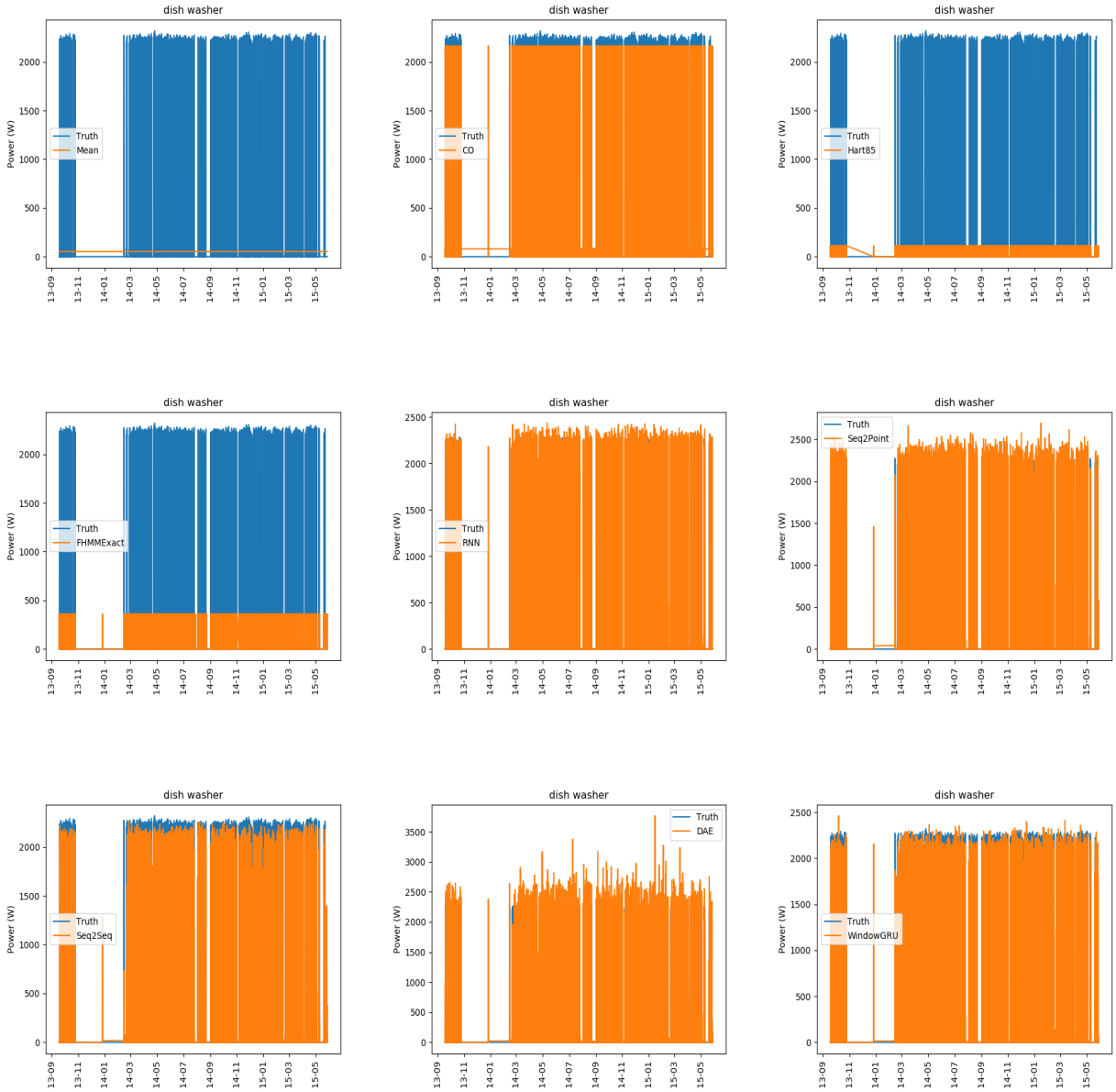

microwave
Truth
DAE
Power (W)
0
500
1000
1500
2000
2500
3000
13-09
13-11
14-01
14-03
14-05
14-07
14-09
14-11
15-01
15-03
15-05


microwave
Truth
WindowGRU
Power (W)
0
250
500
750
1000
1250
1500
1750
13-09
13-11
14-01
14-03
14-05
14-07
14-09
14-11
15-01
15-03
15-05

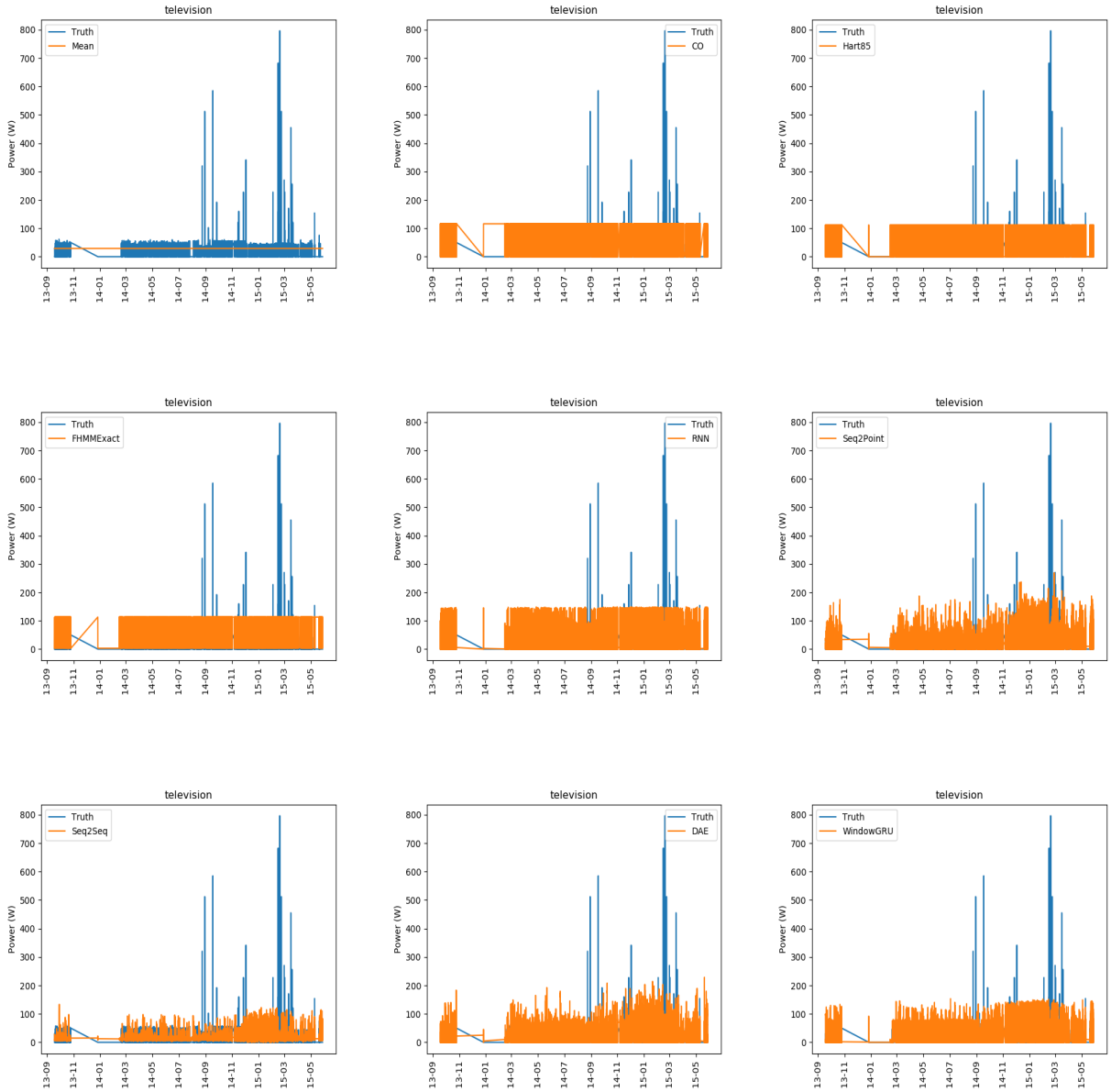
A.10 REFIT - Kettle



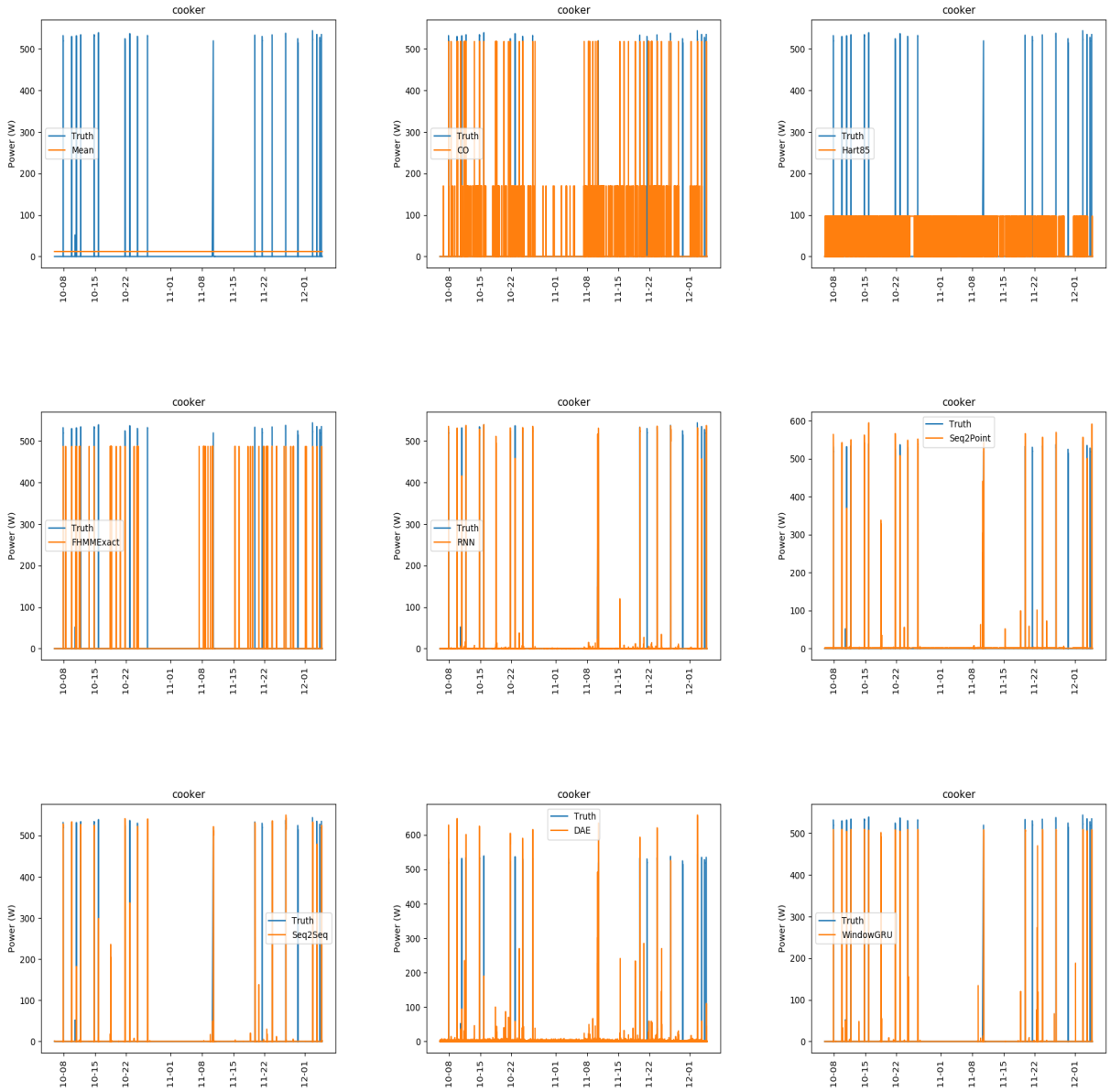
A.11 REFIT - Dish washer



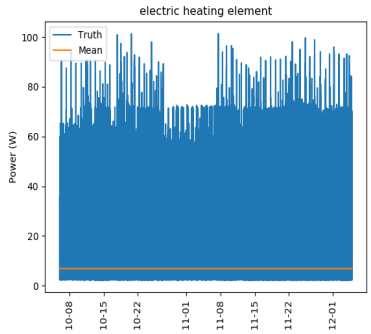
A.12 REFIT - Cooker

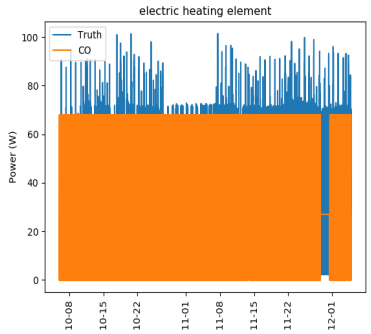


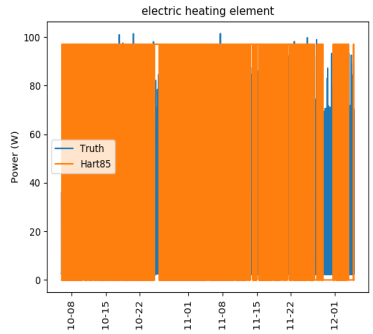
A.13 DRED - Cooker

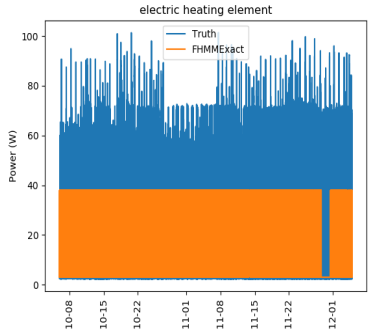


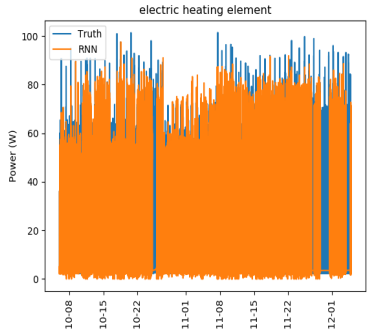
A.14 DRED - Electric heating element

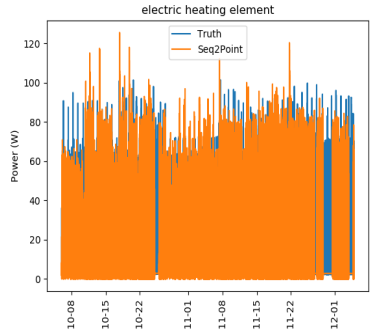


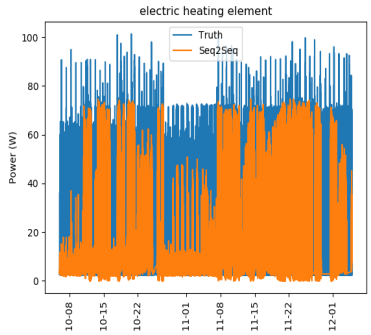


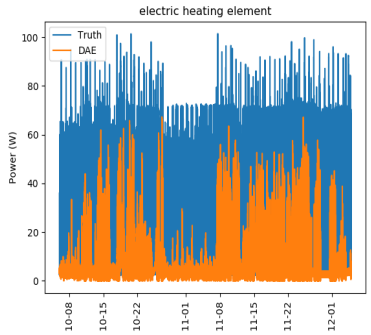


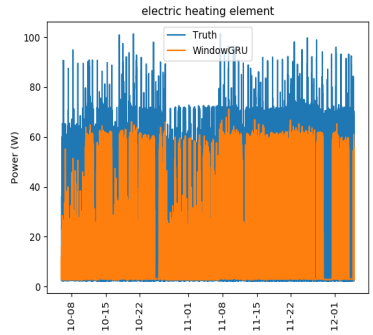




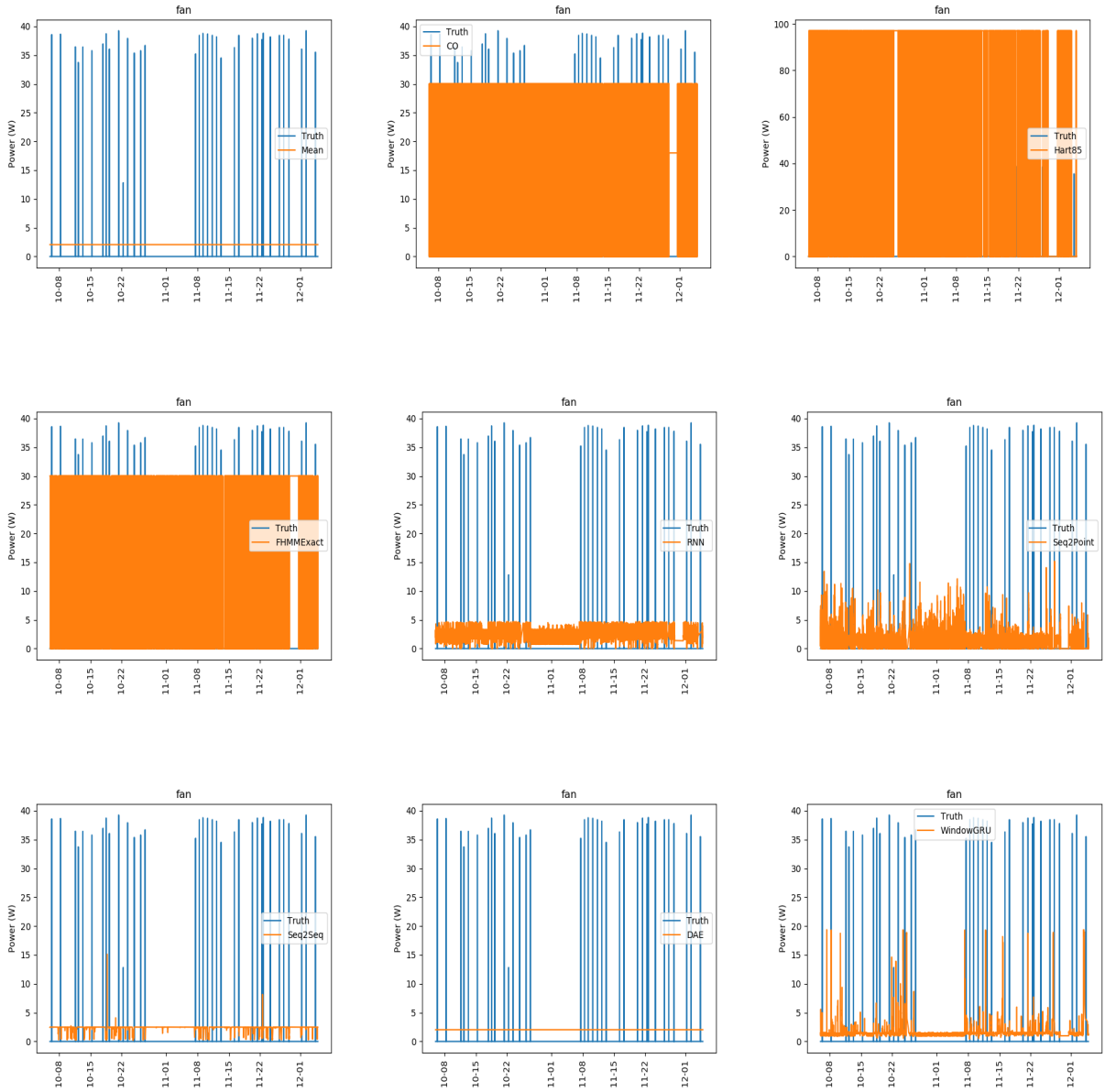




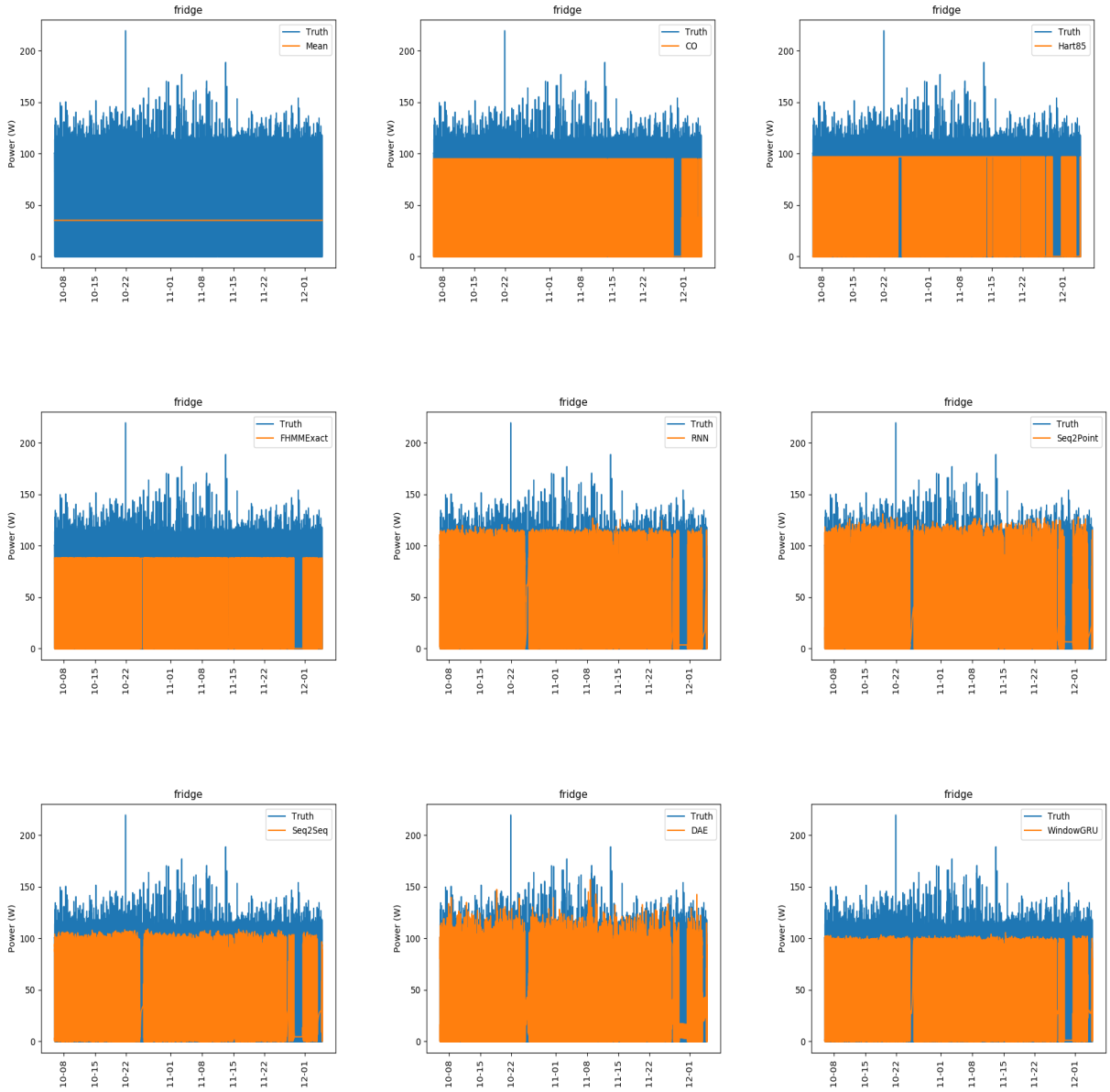




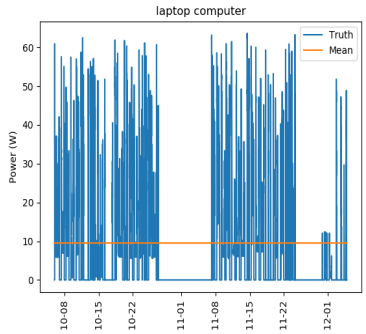
A.15 DRED - Fan

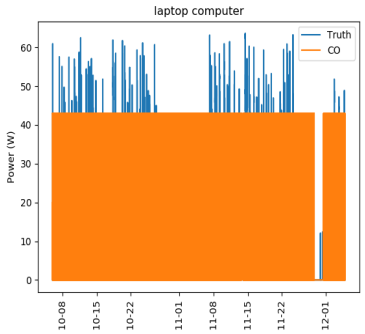

fan
Power (W)
Truth
Mean
CO
Hart85
FHMMExact
RNN
Seq2Point
Seq2Seq
DAE
WindowGRU
10-08
10-15
10-22
11-01
11-08
11-15
11-22
12-01

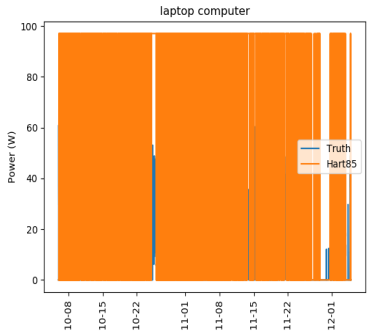
A.16 DRED - Fridge

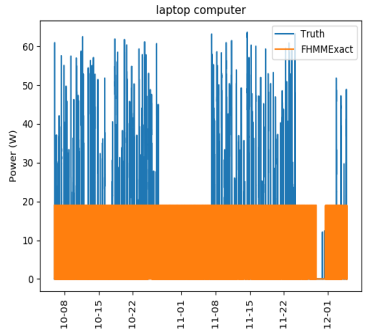


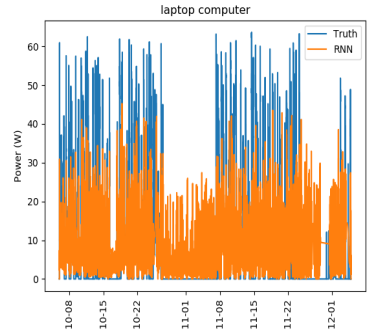
A.17 DRED - Laptop

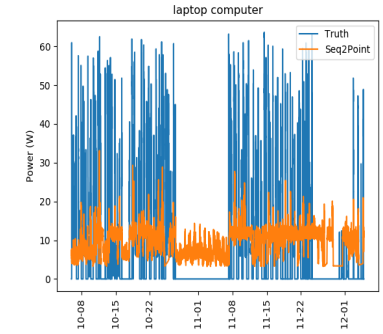


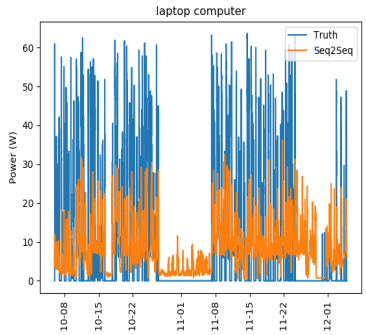


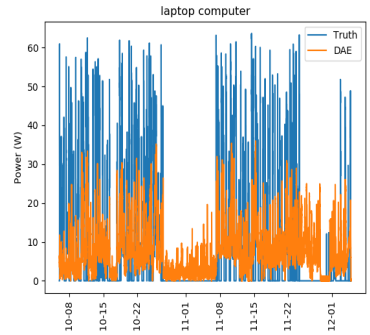


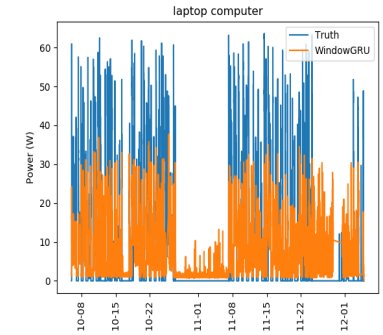




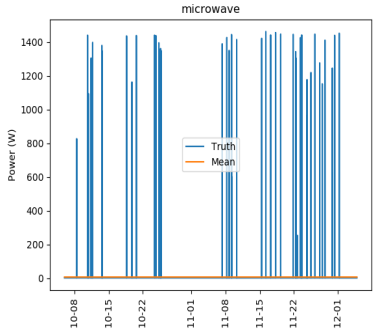


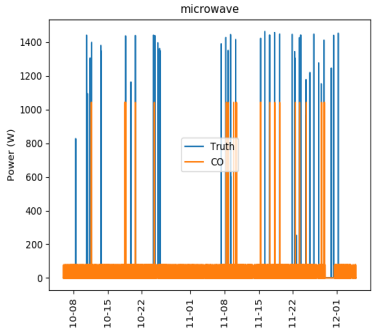


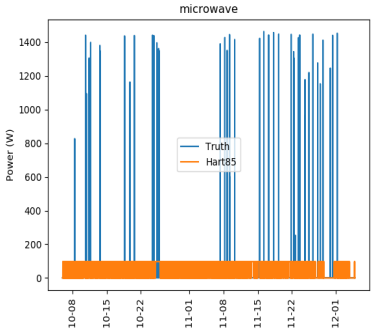


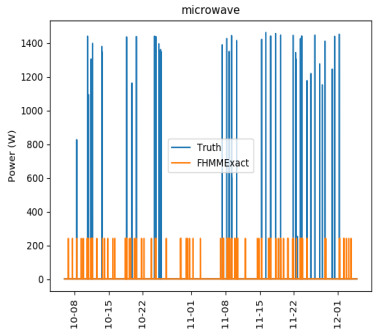


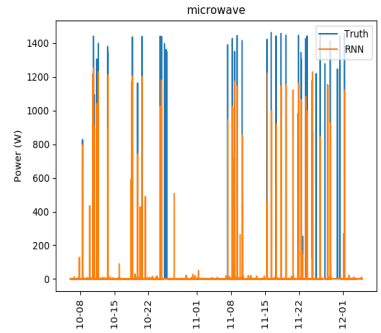
A.18 DRED - Microwave

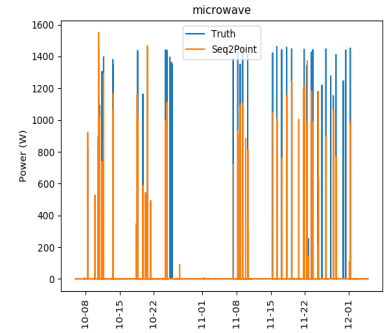


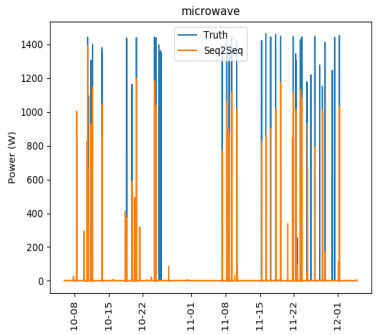


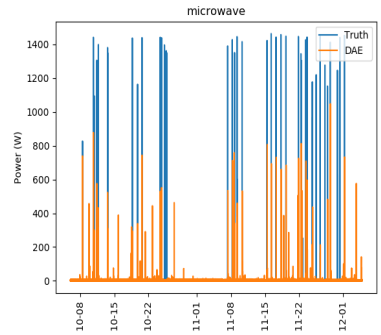


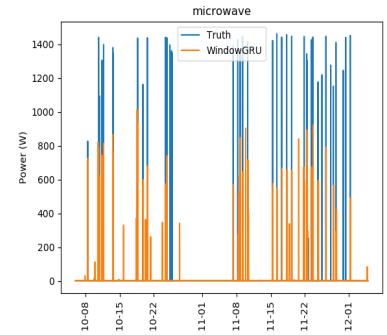




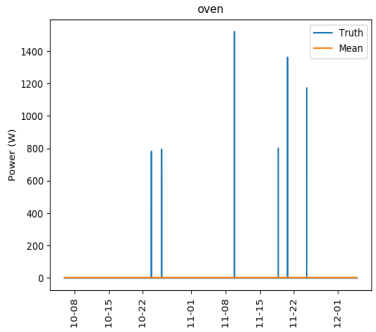


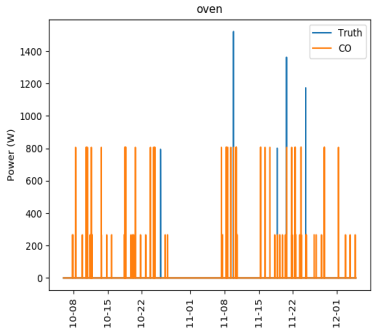


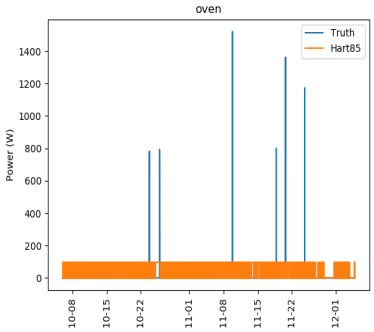


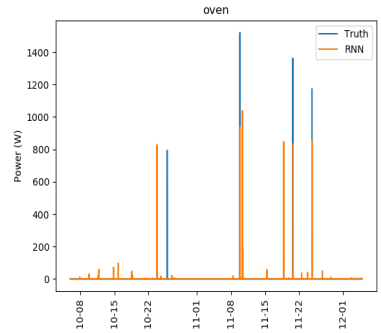


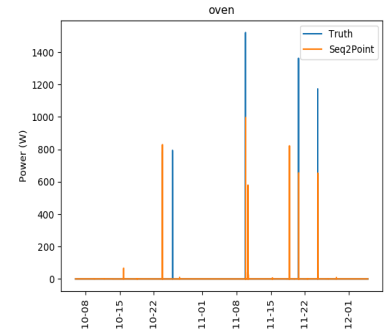
A.19 DRED - Oven

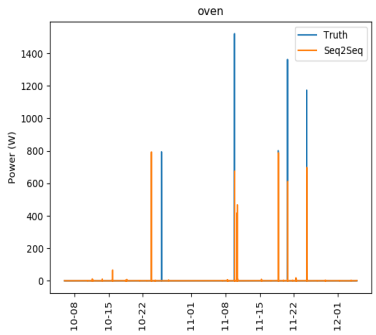


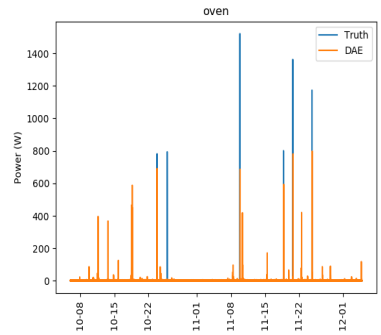


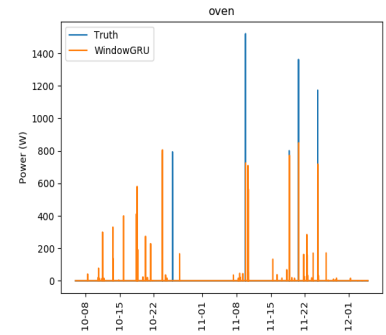




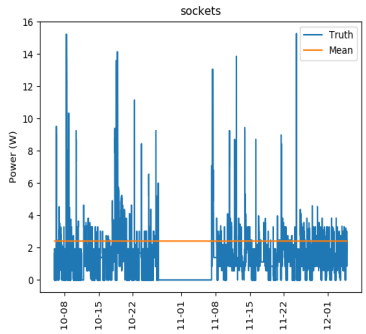


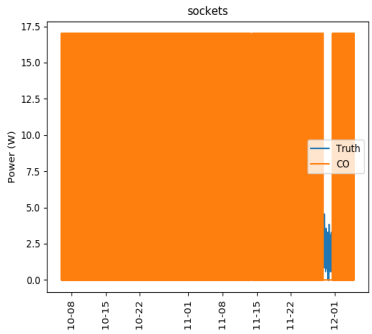


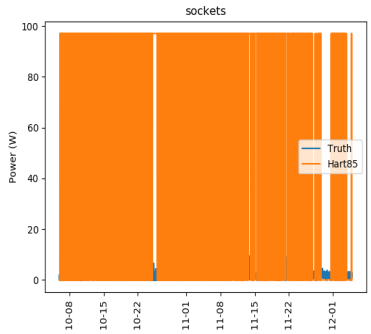


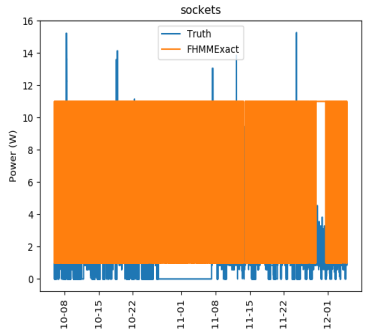


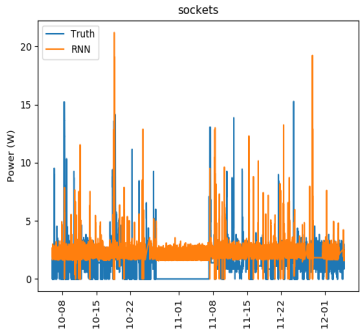
A.20 DRED - Sockets

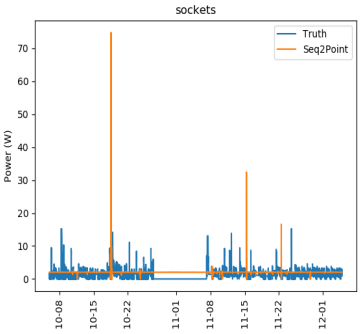


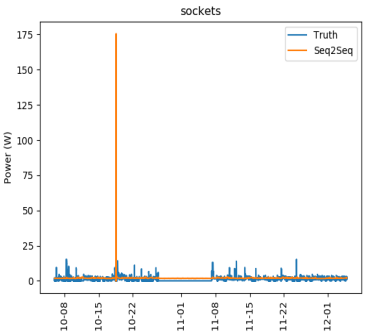


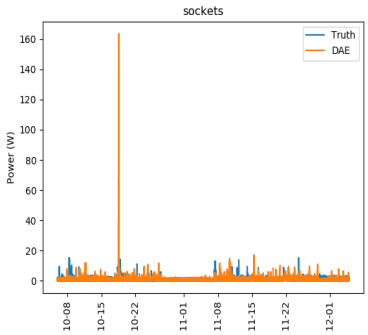


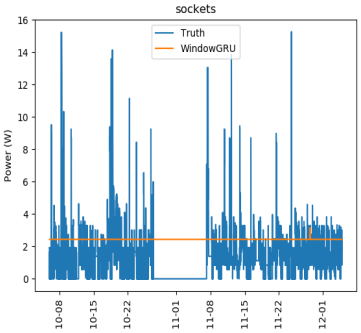




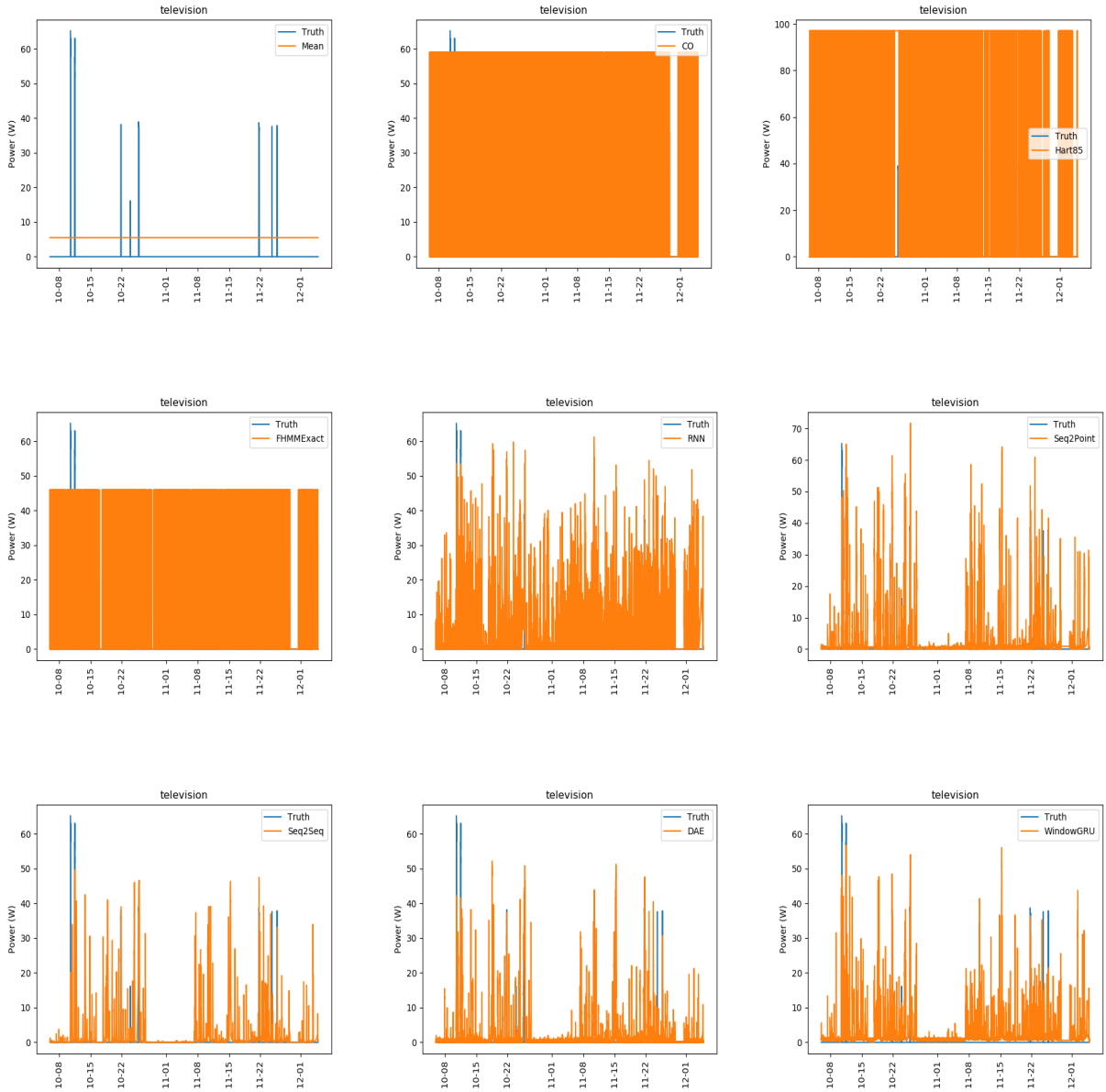




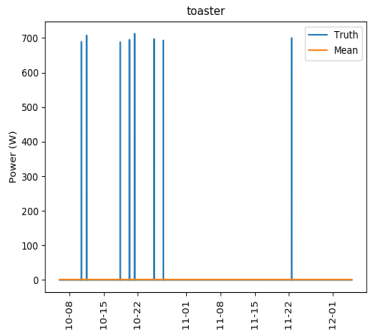


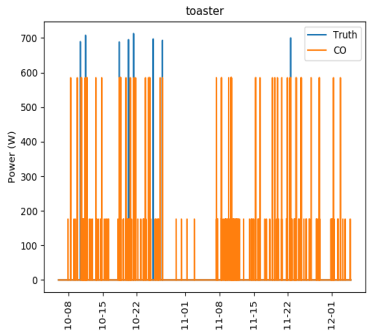


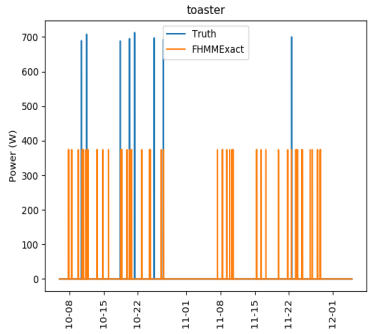
A.21 DRED - Television

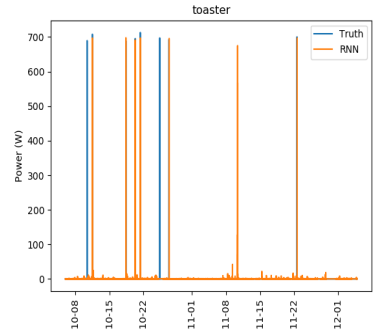


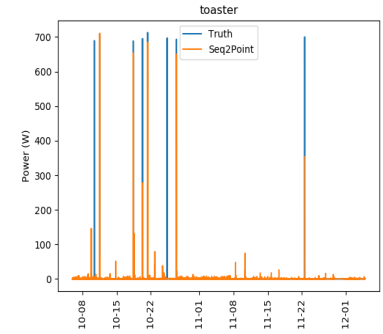
A.22 DRED - Toaster

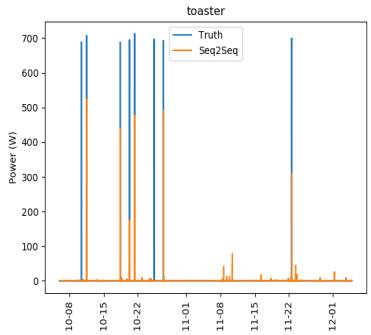


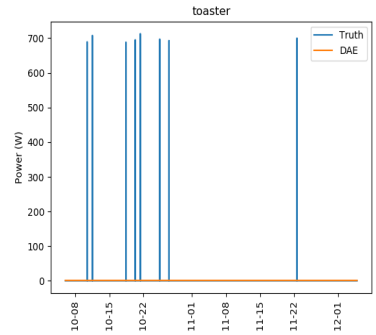


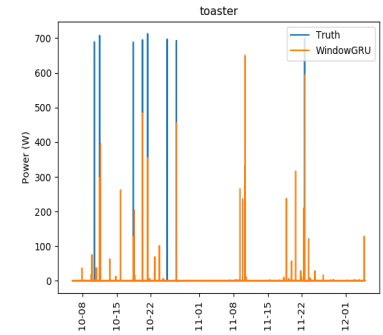




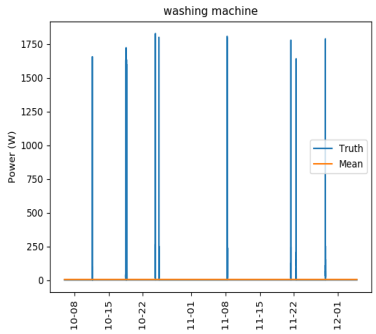


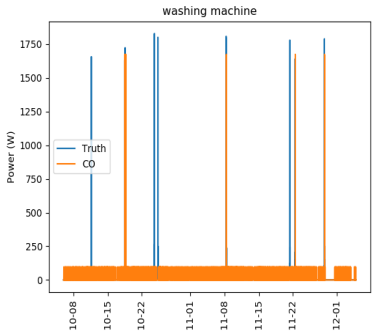


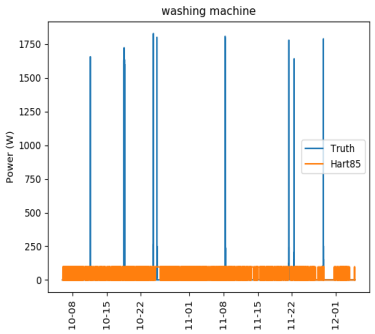


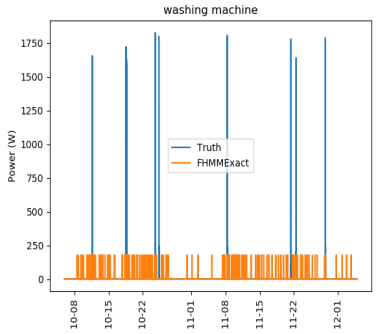


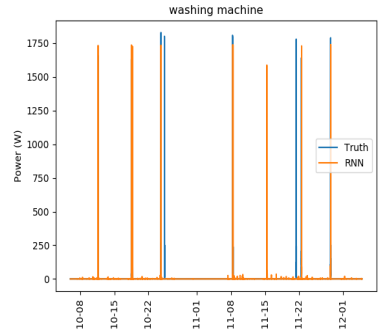
A.23 DRED - Washing machine

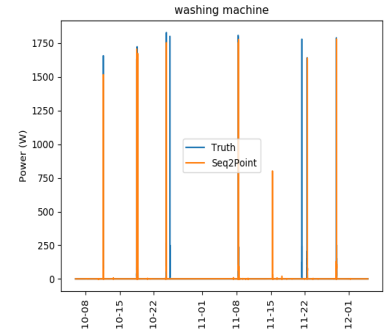


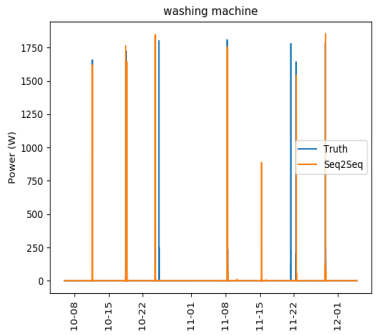


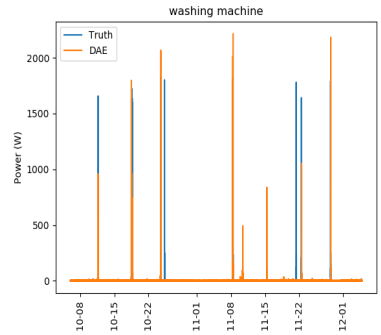


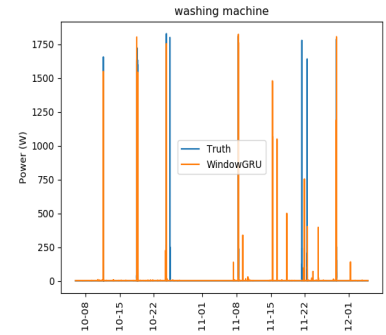




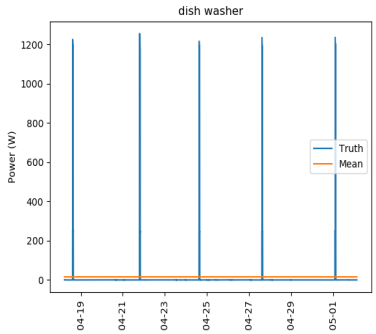


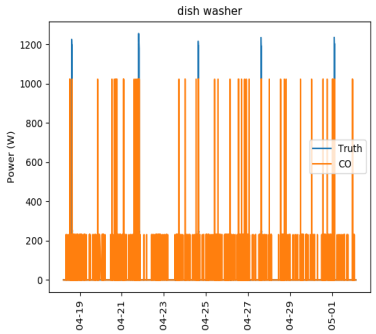


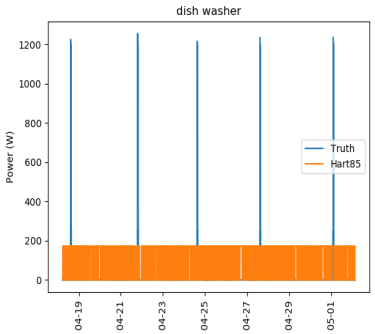


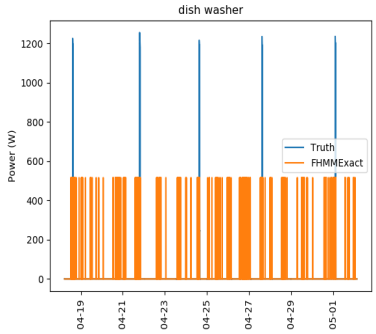


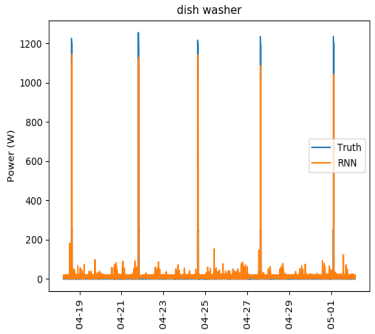
A.24 REDD - Dish washer

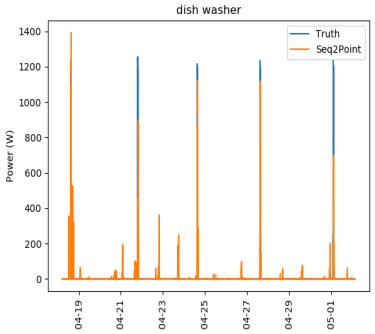


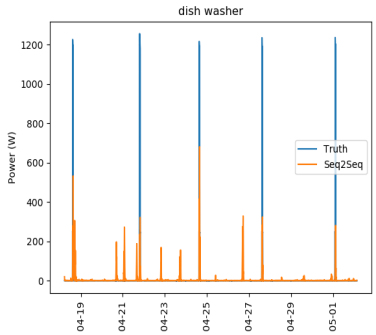


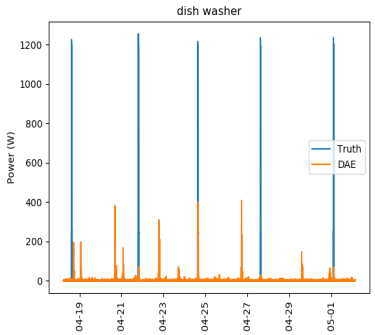


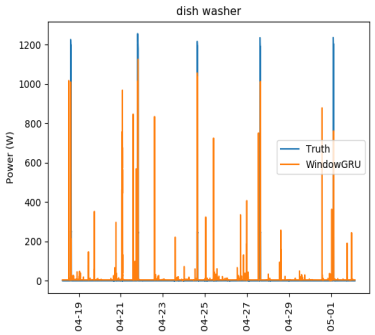




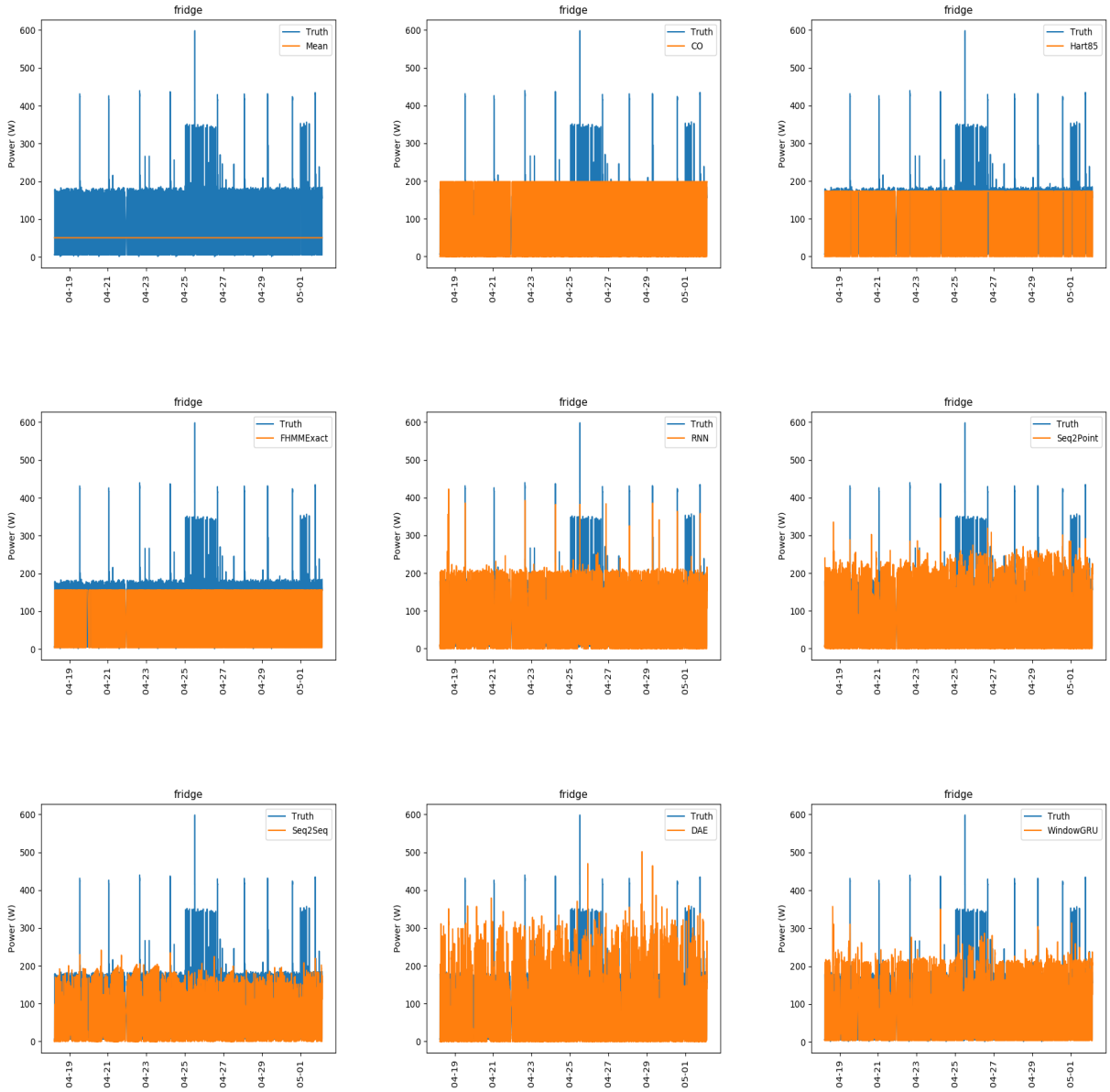




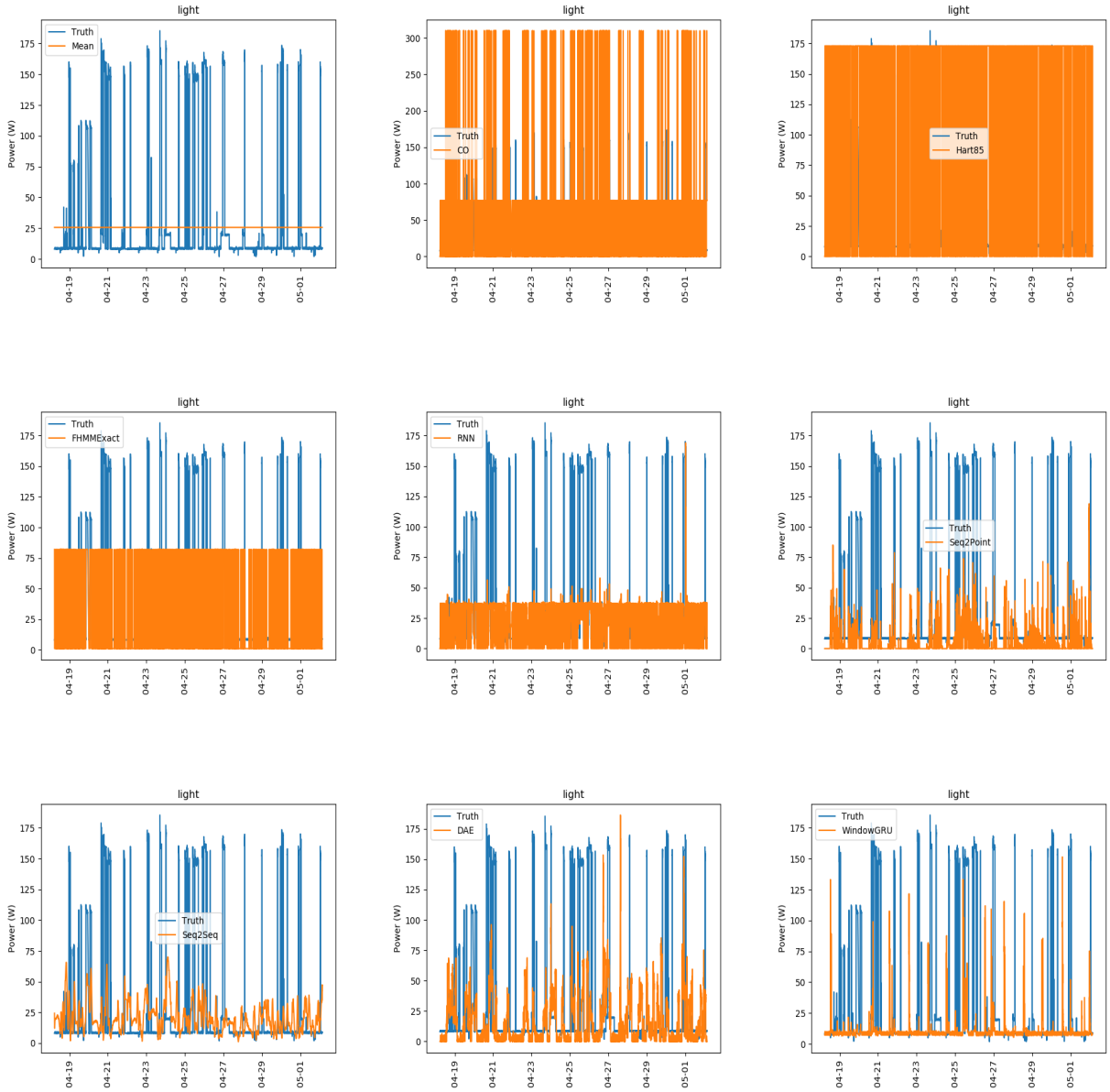




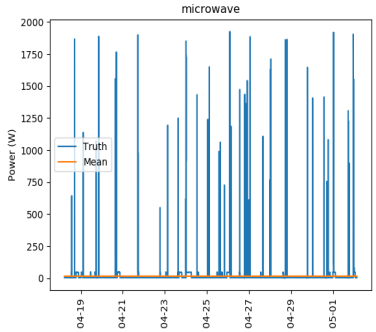
A.25 REDD - Fridge

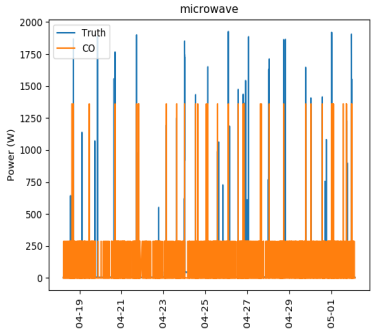


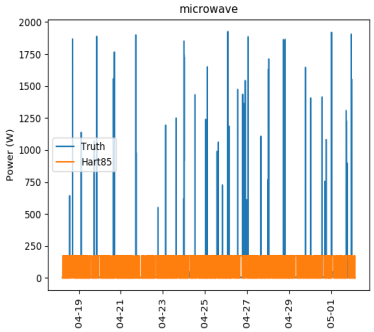
A.26 REDD - Light

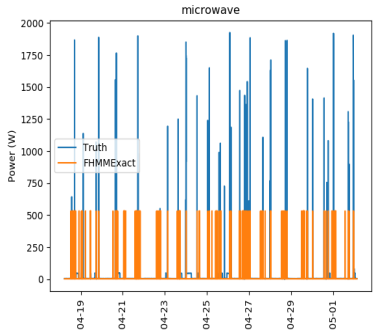


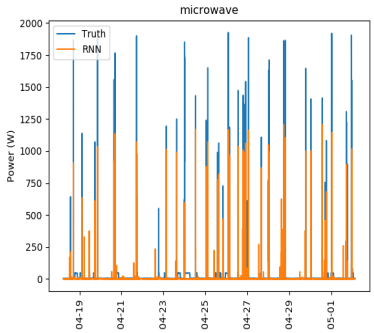
A.27 REDD - Microwave

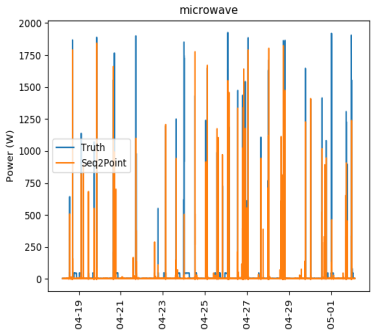


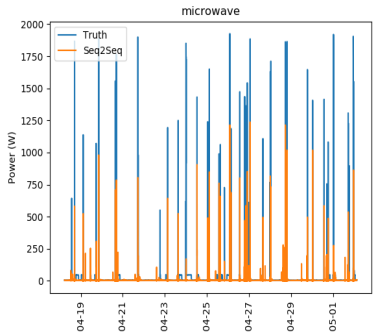


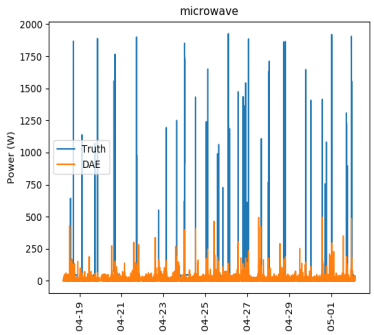


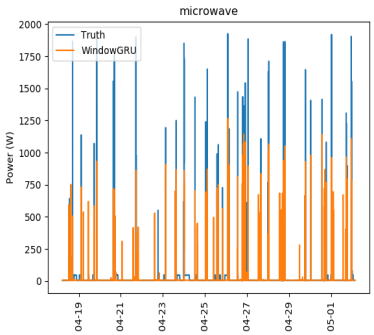




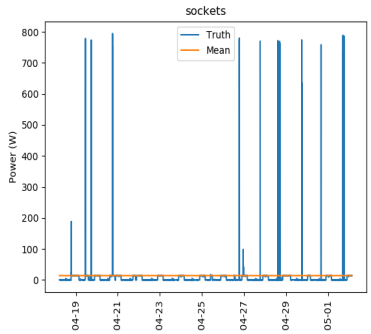


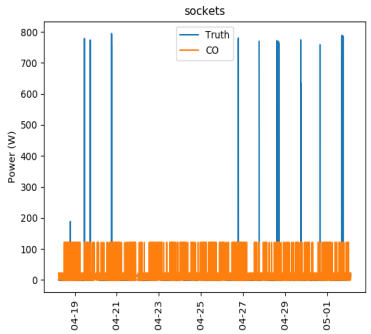


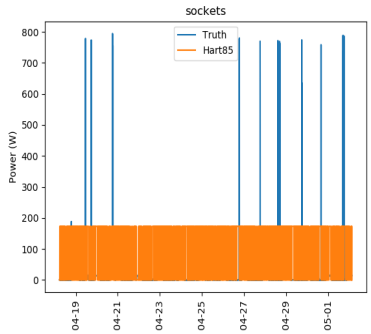


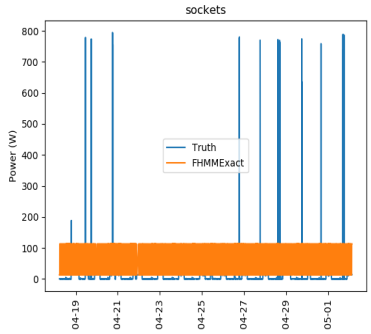


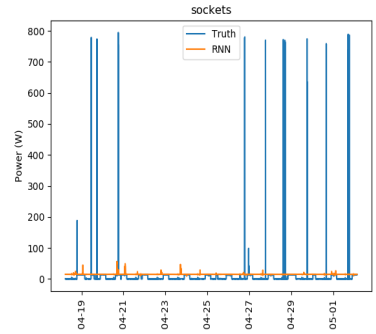
A.28 REDD - Sockets

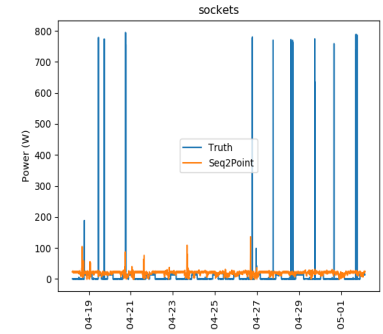


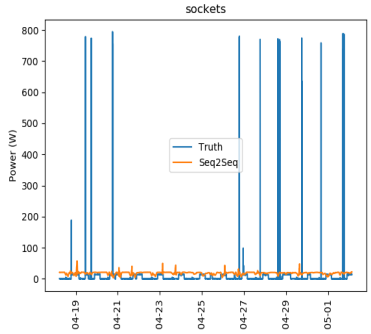


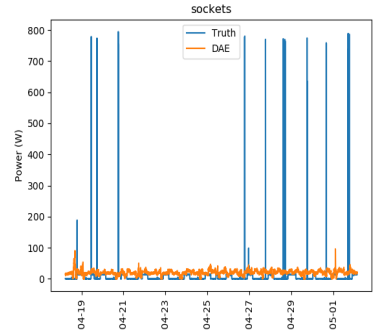


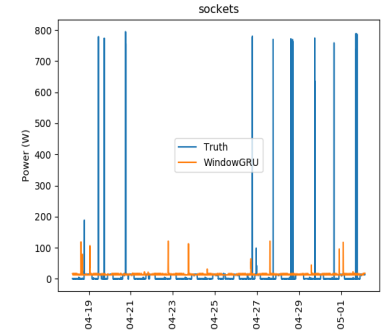




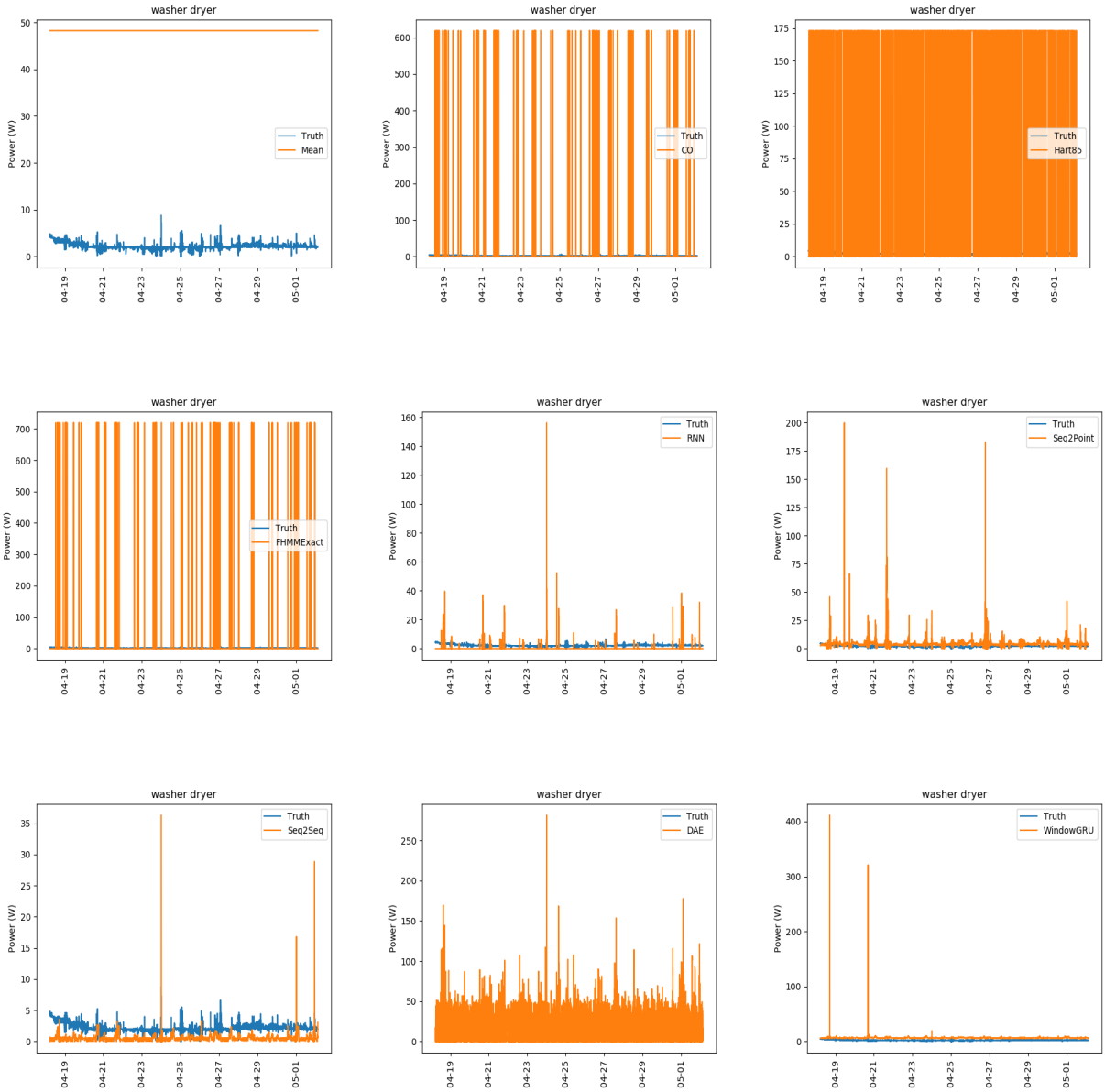




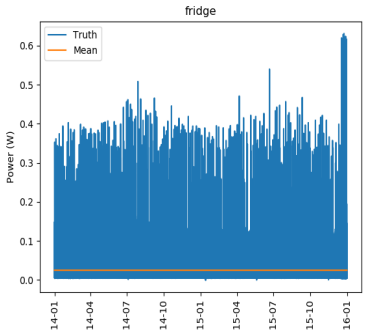


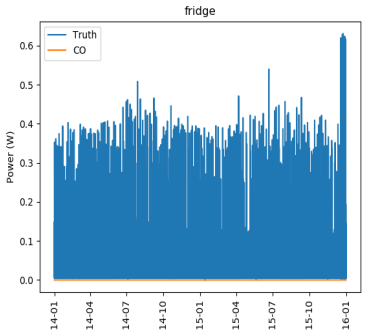


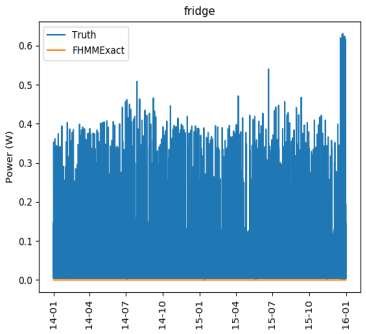
A.29 REDD - Washer dryer

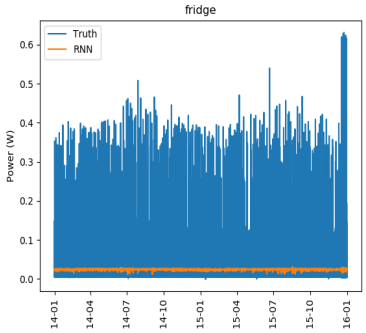


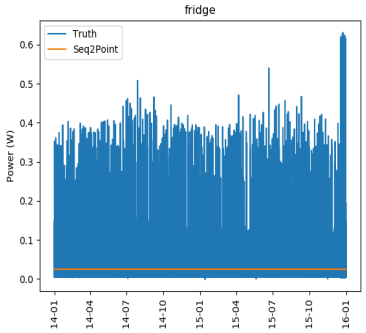
A.30 SMART - Fridge

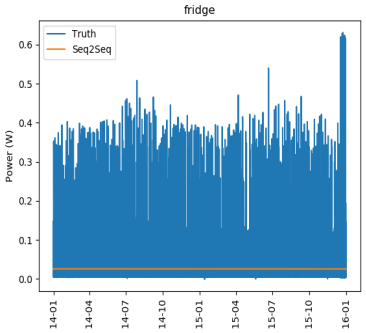


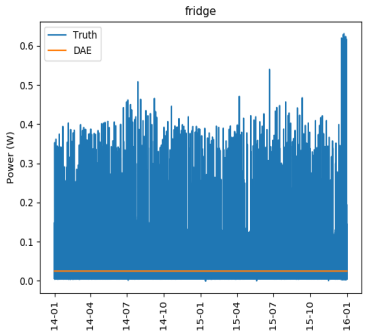


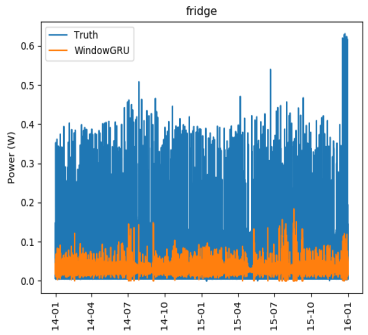




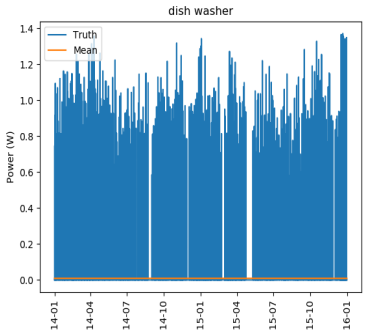


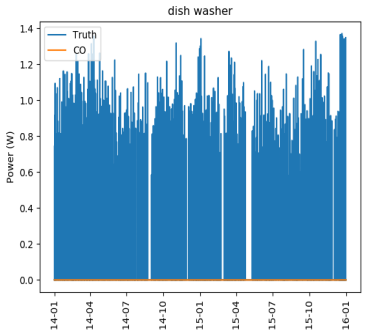


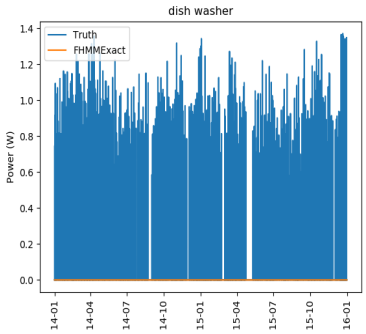


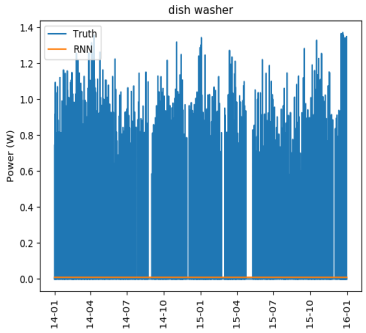


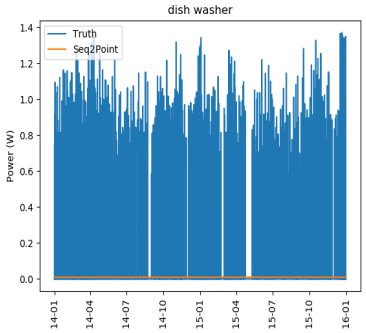
A.31 SMART - Dish washer

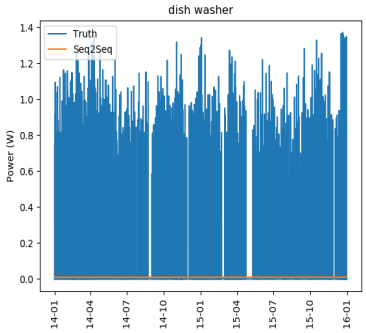


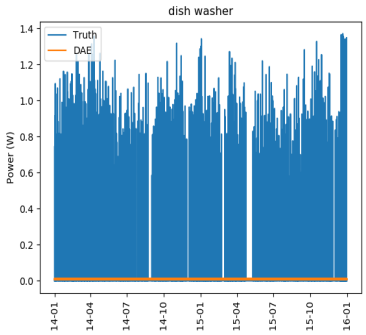


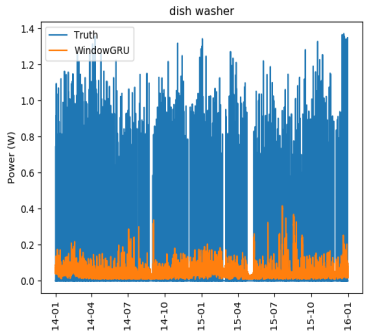




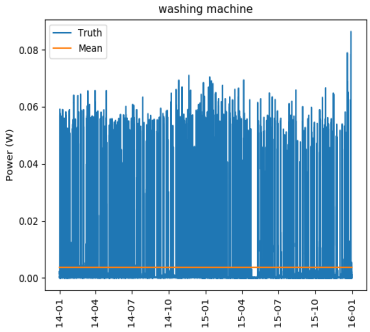


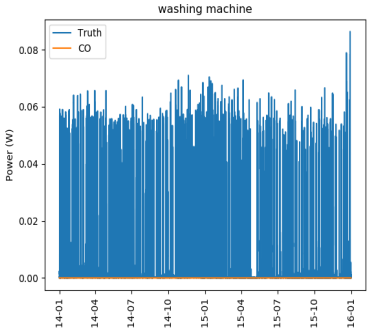


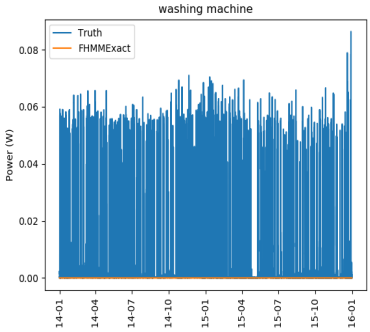


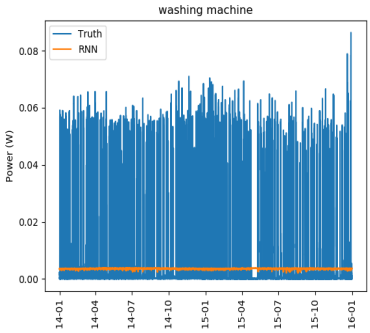


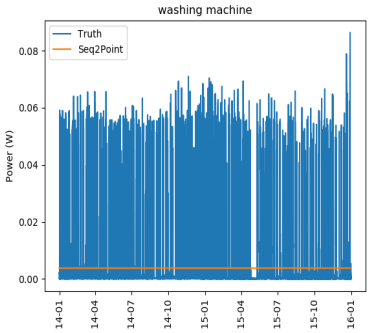
A.32 SMART - Washing machine

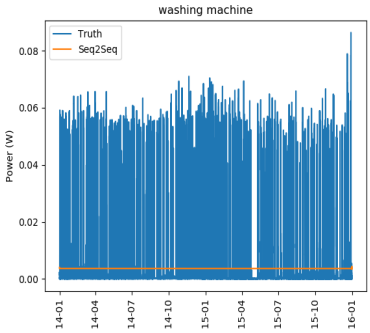


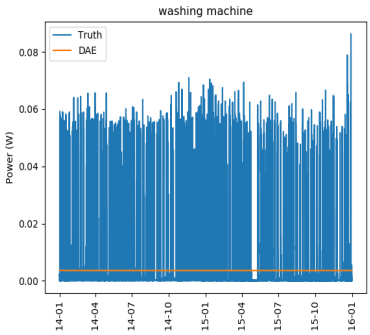


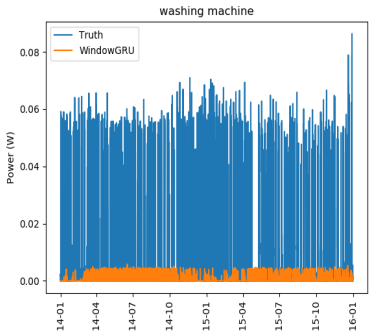




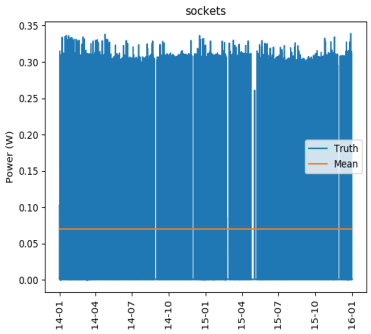


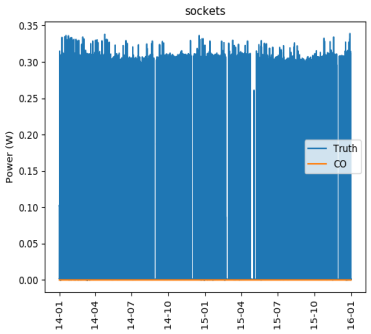


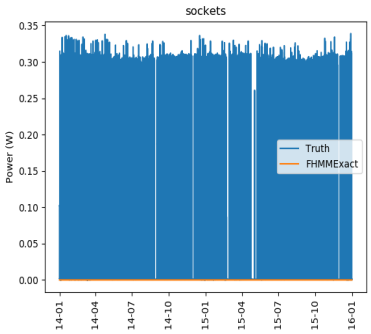


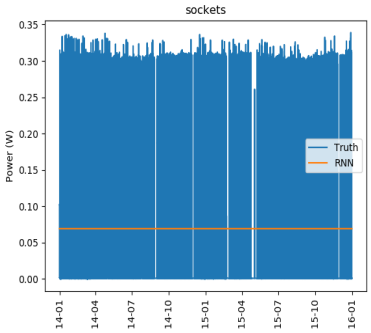


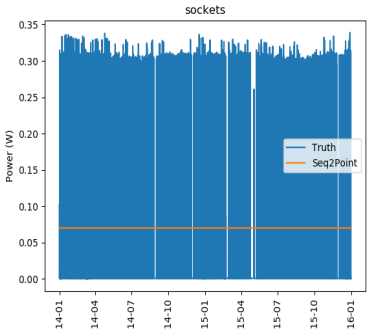
A.33 SMART - Sockets

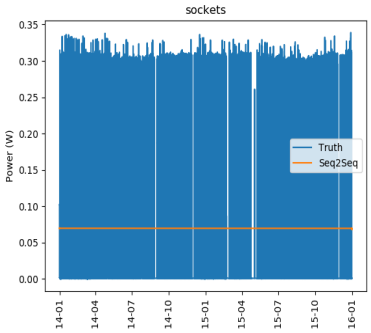


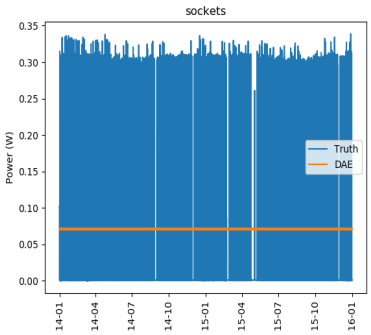




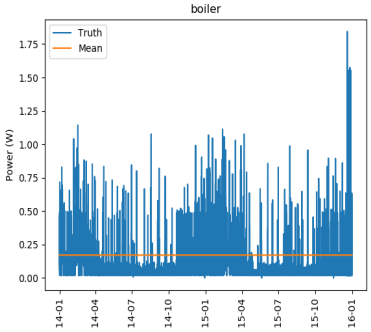


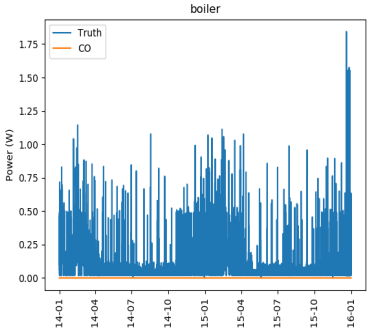


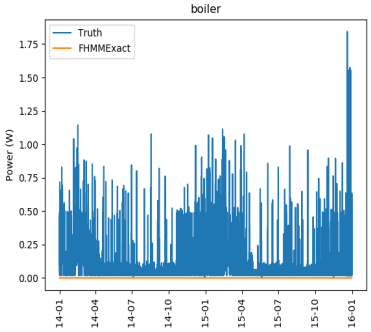


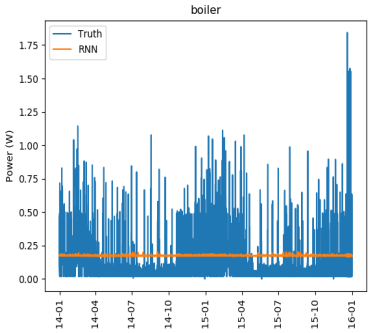


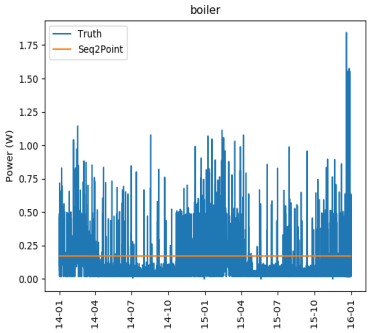
A.34 SMART - Boiler

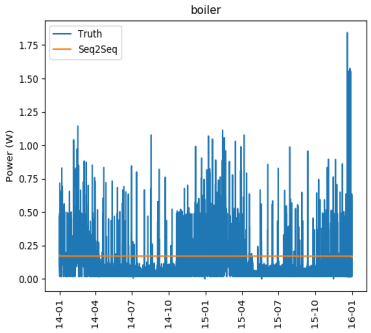


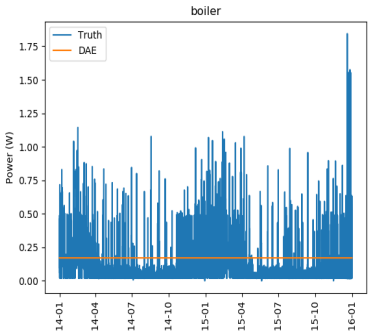


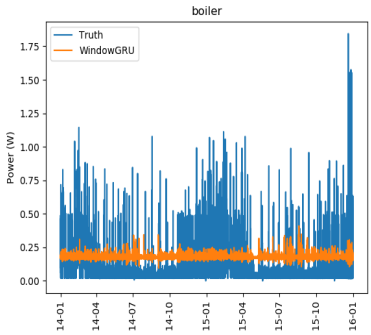






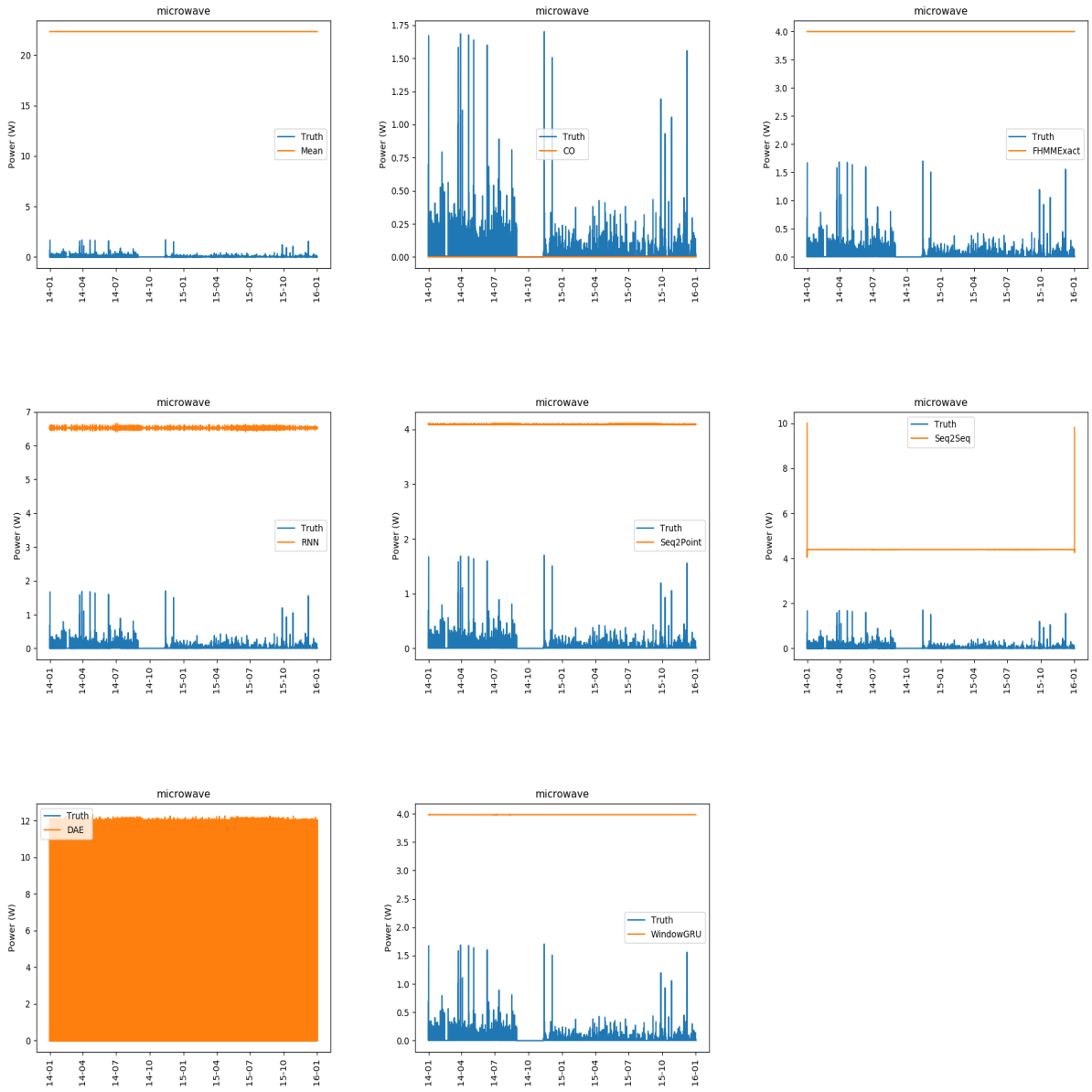




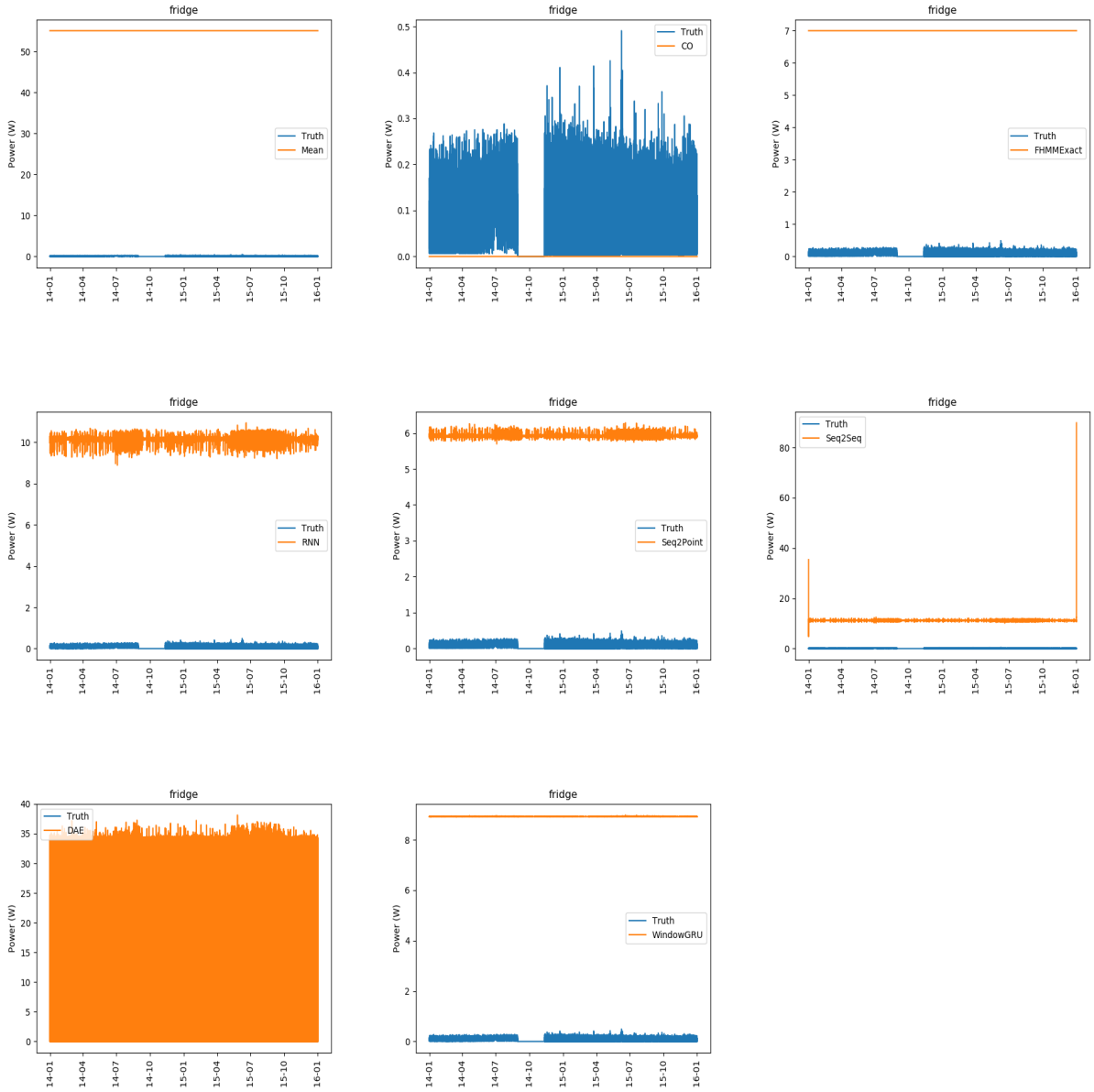


B Stage 2: Appliance graphs

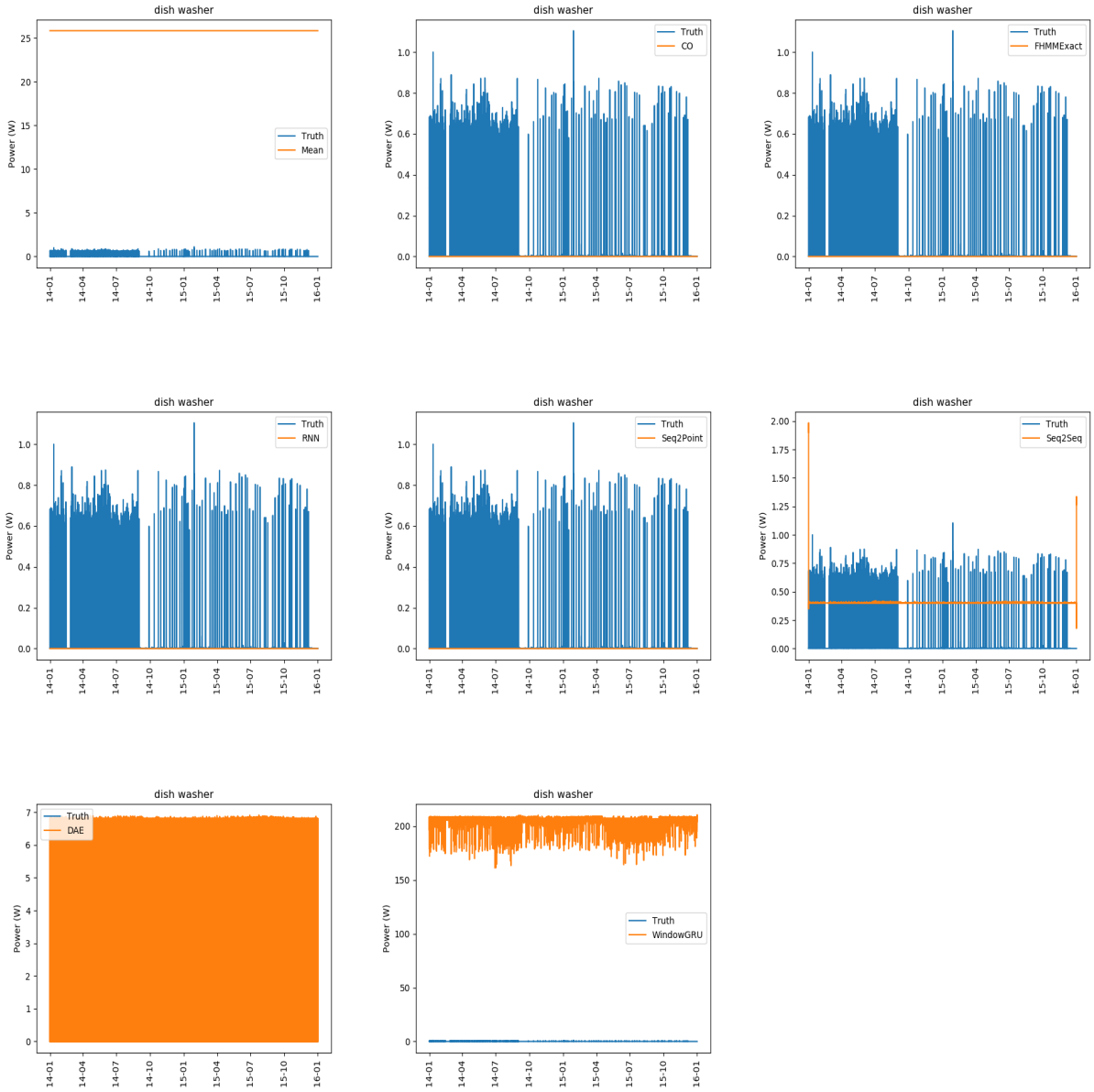
B.1 Train on REDD, test on SMART - Microwave



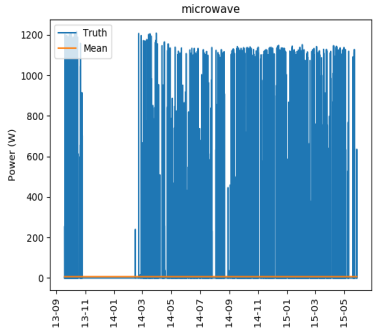
B.2 Train on REDD, test on SMART - Fridge

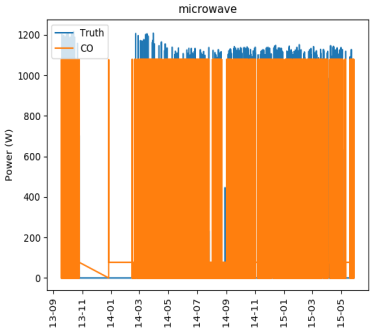


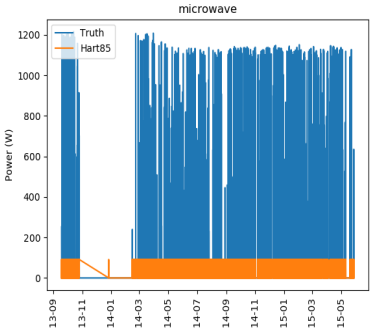
B.3 Train on REDD, test on SMART - Dish washer

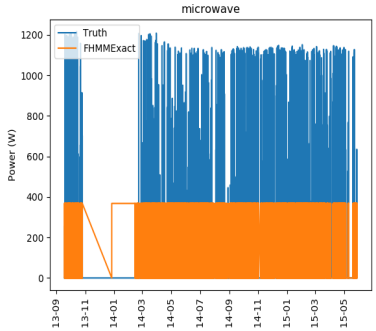


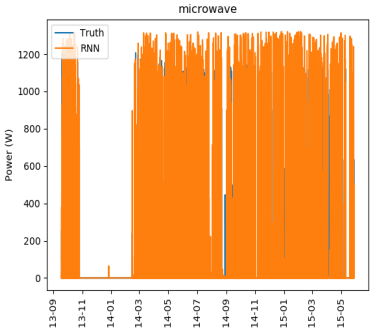
B.4 Train on UKDALE, test on REFIT - Microwave

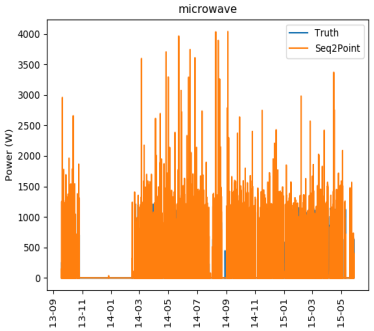


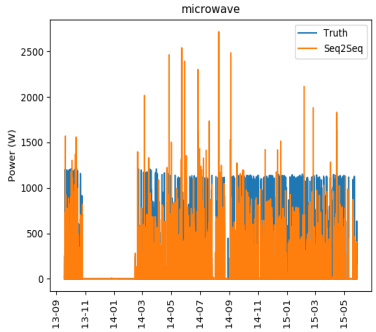


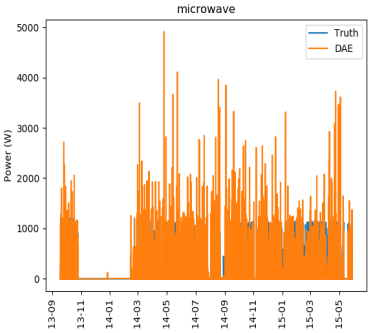


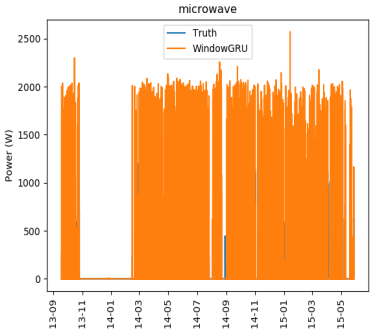




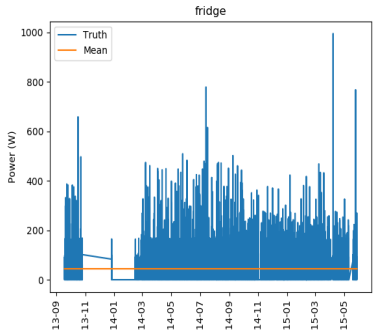


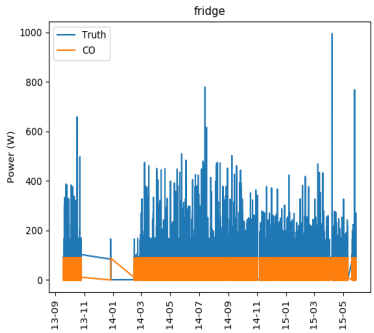


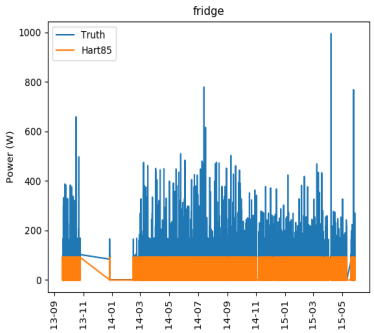


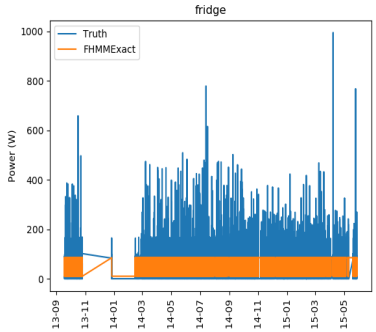


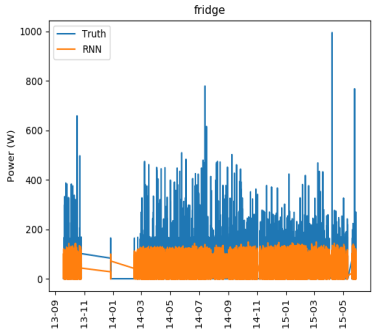
B.5 Train on UKDALE, test on REFIT - Fridge

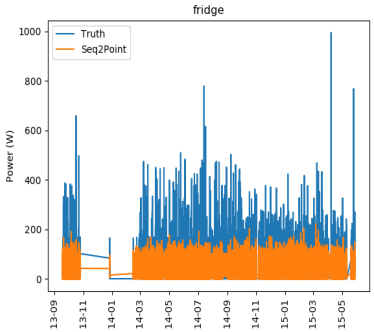


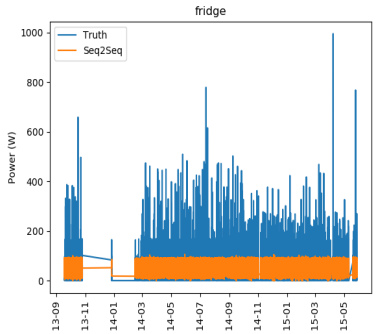


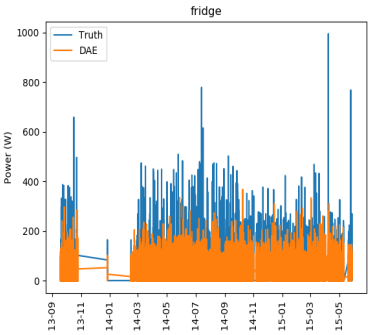


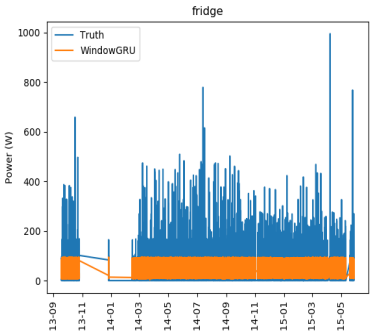




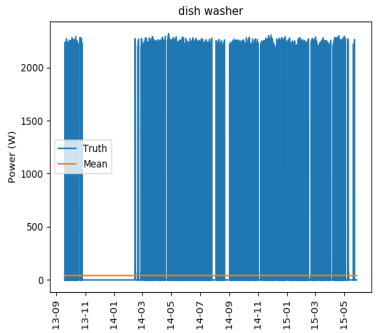


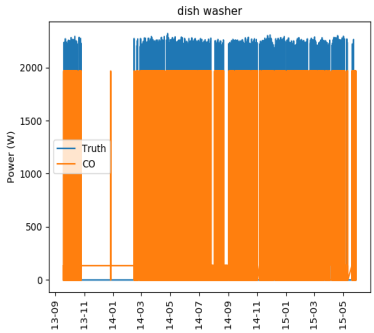


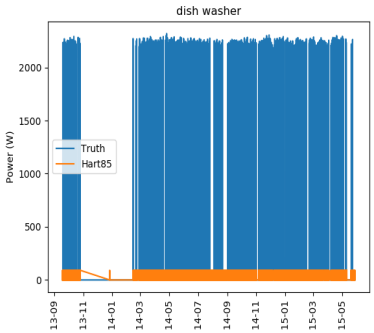


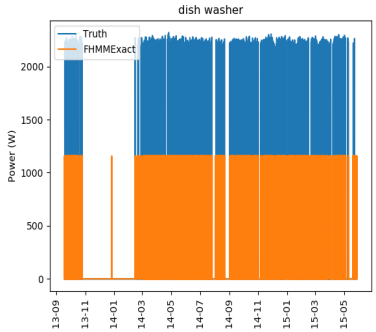


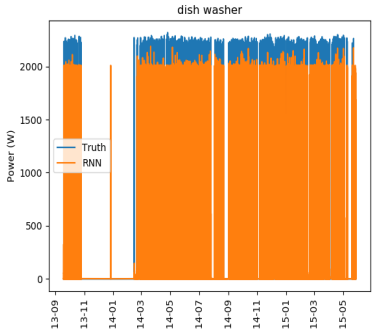
B.6 Train on UKDALE, test on REFIT - Dish washer

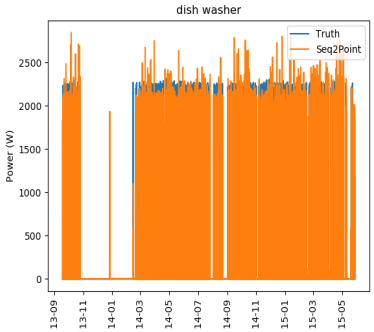


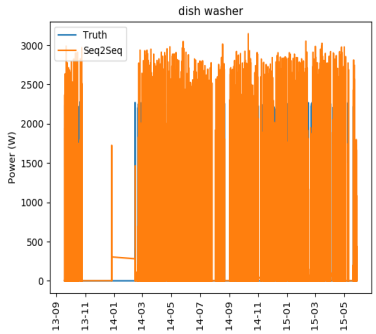


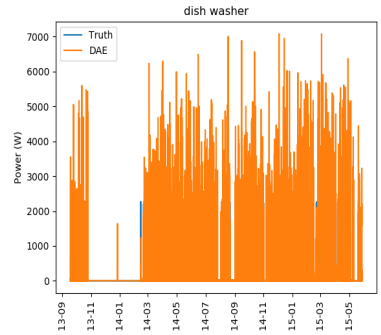


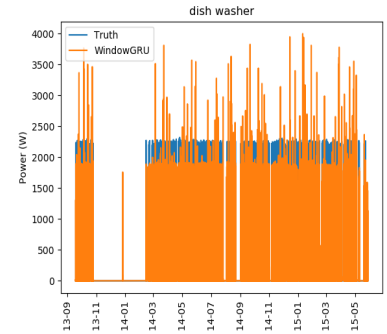






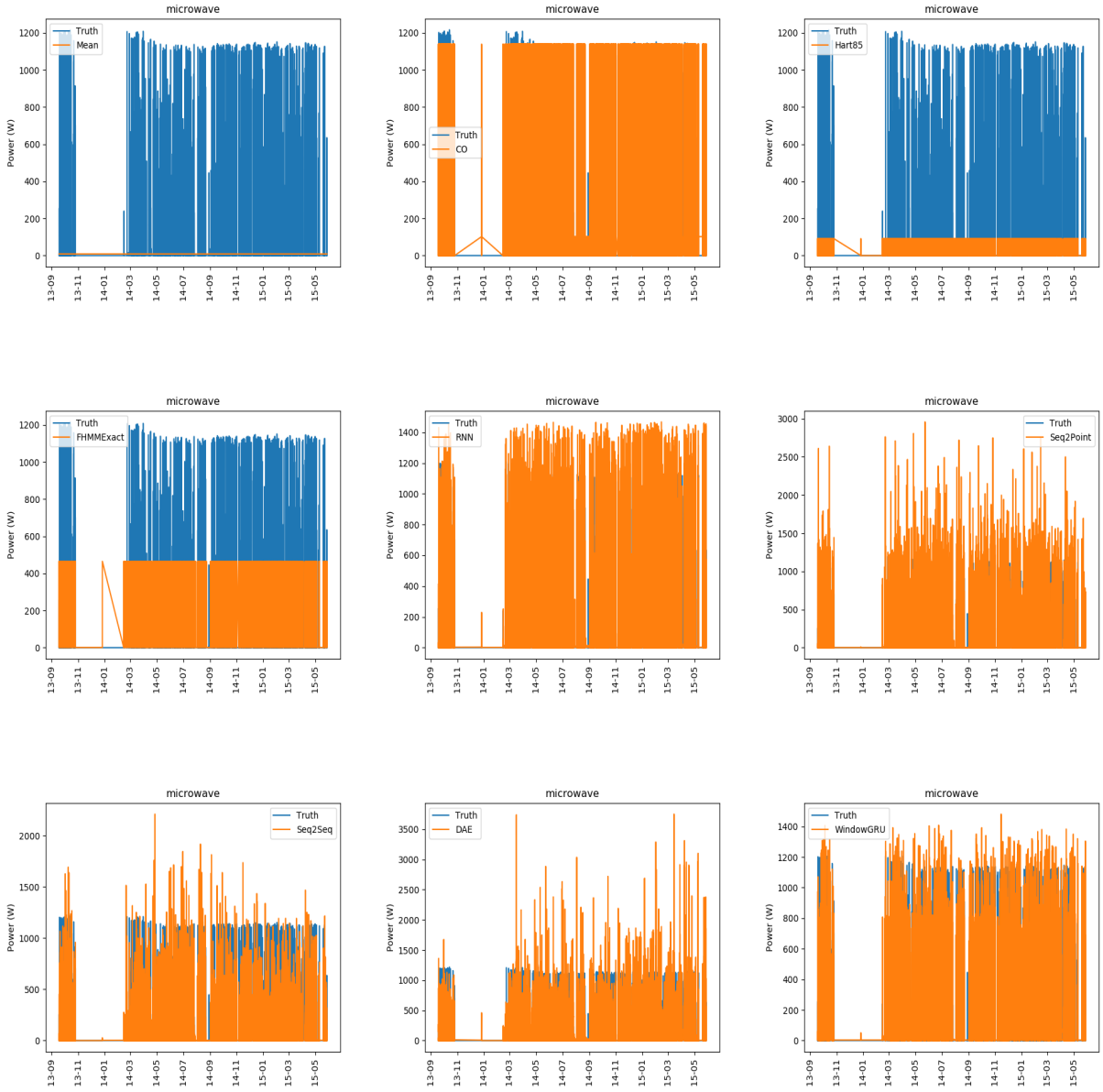




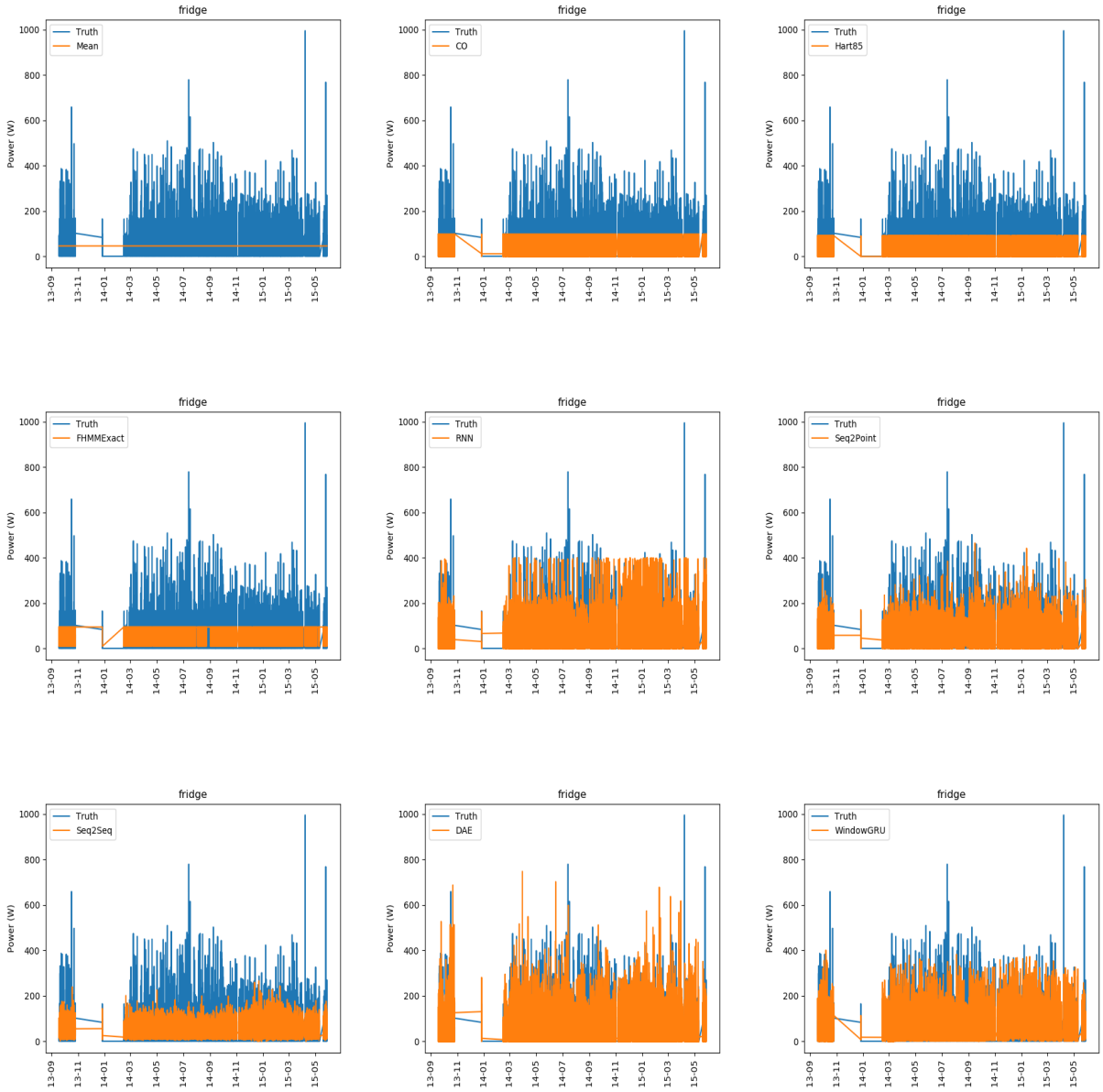


C Stage 3: Appliance graphs

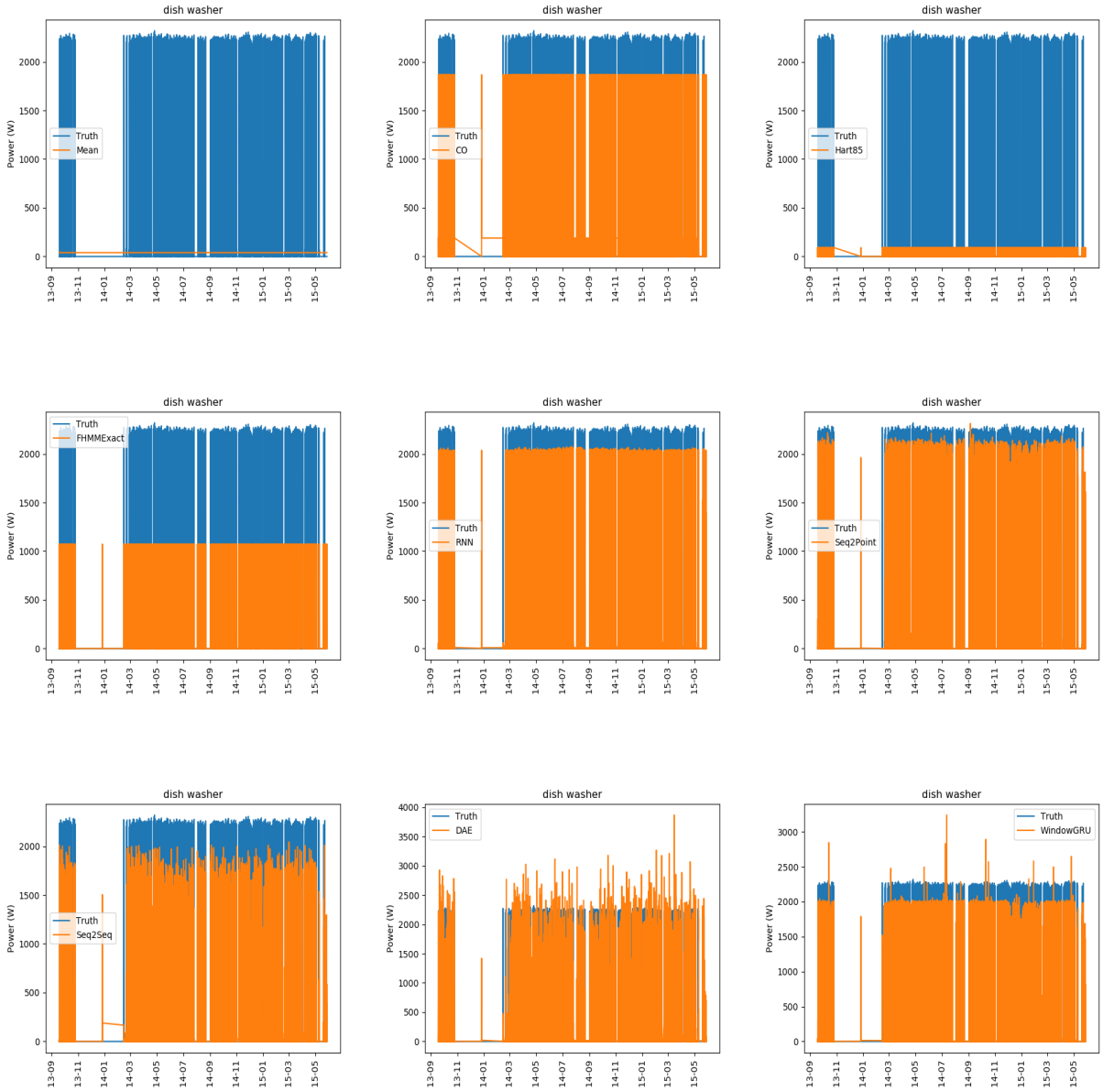
C.1 Train on UKDALE and REDD, test on REFIT - Microwave



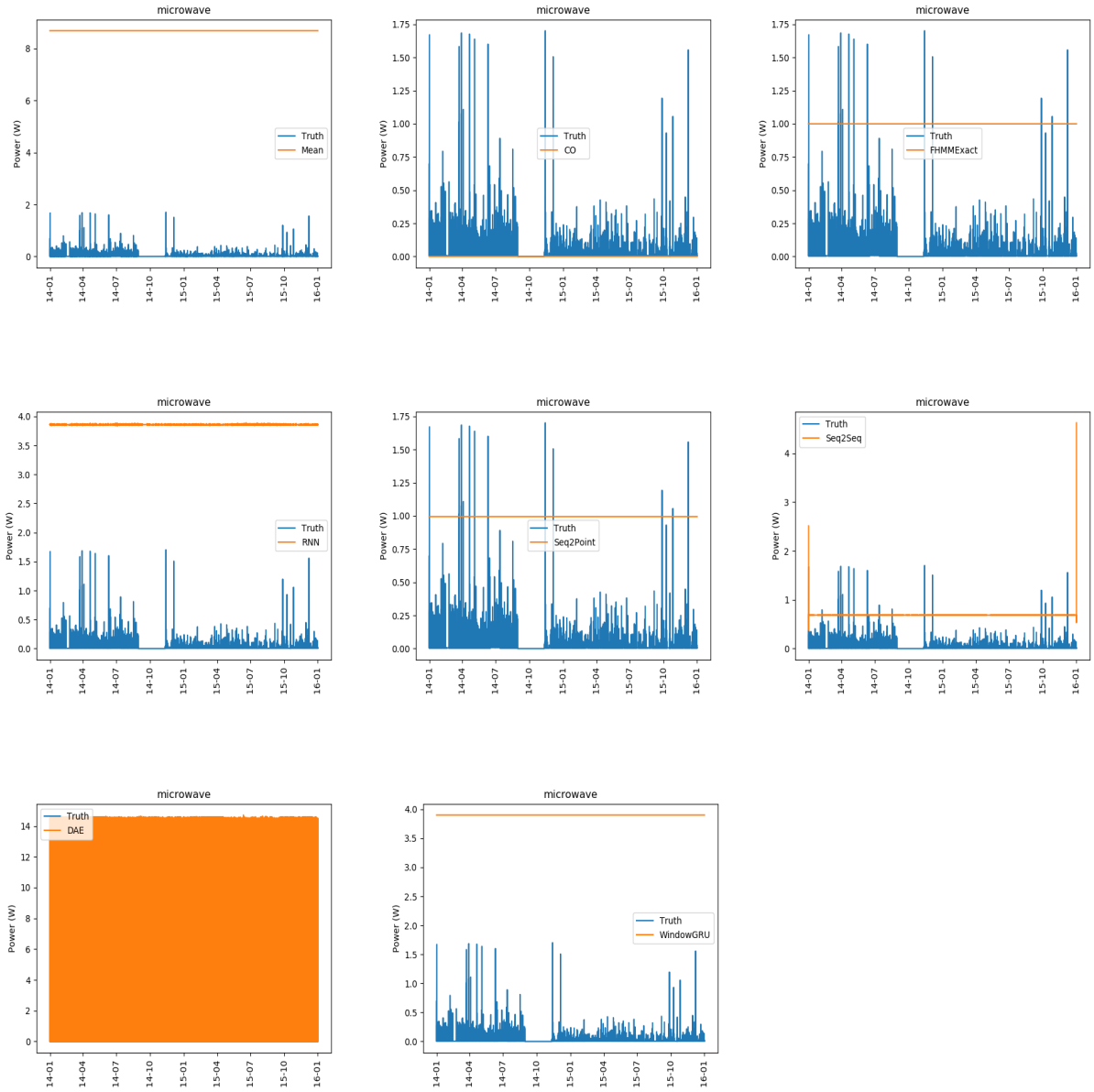
C.2 Train on UKDALE and REDD, test on REFIT - Fridge



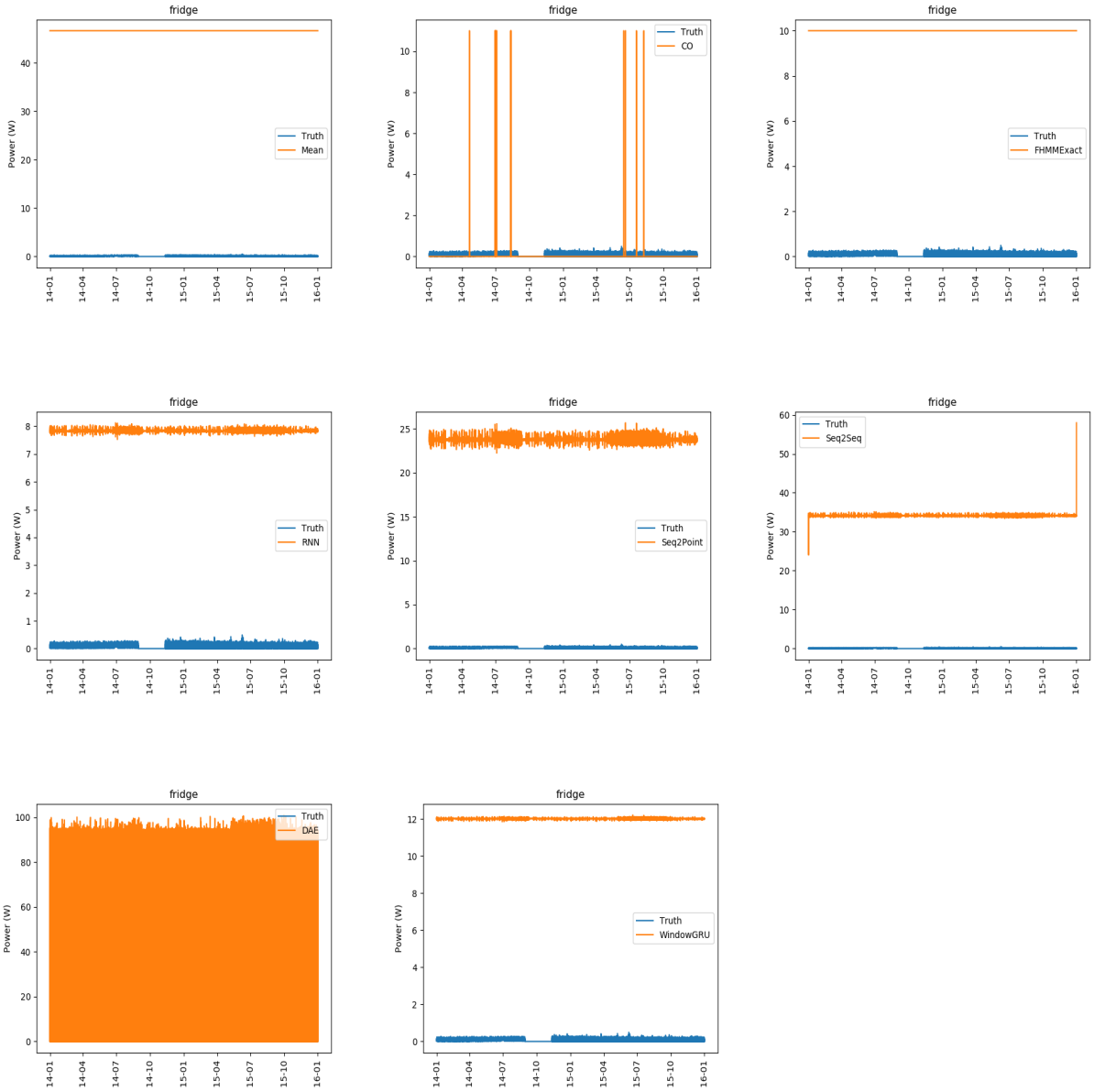
C.3 Train on UKDALE and REDD, test on REFIT - Dish washer



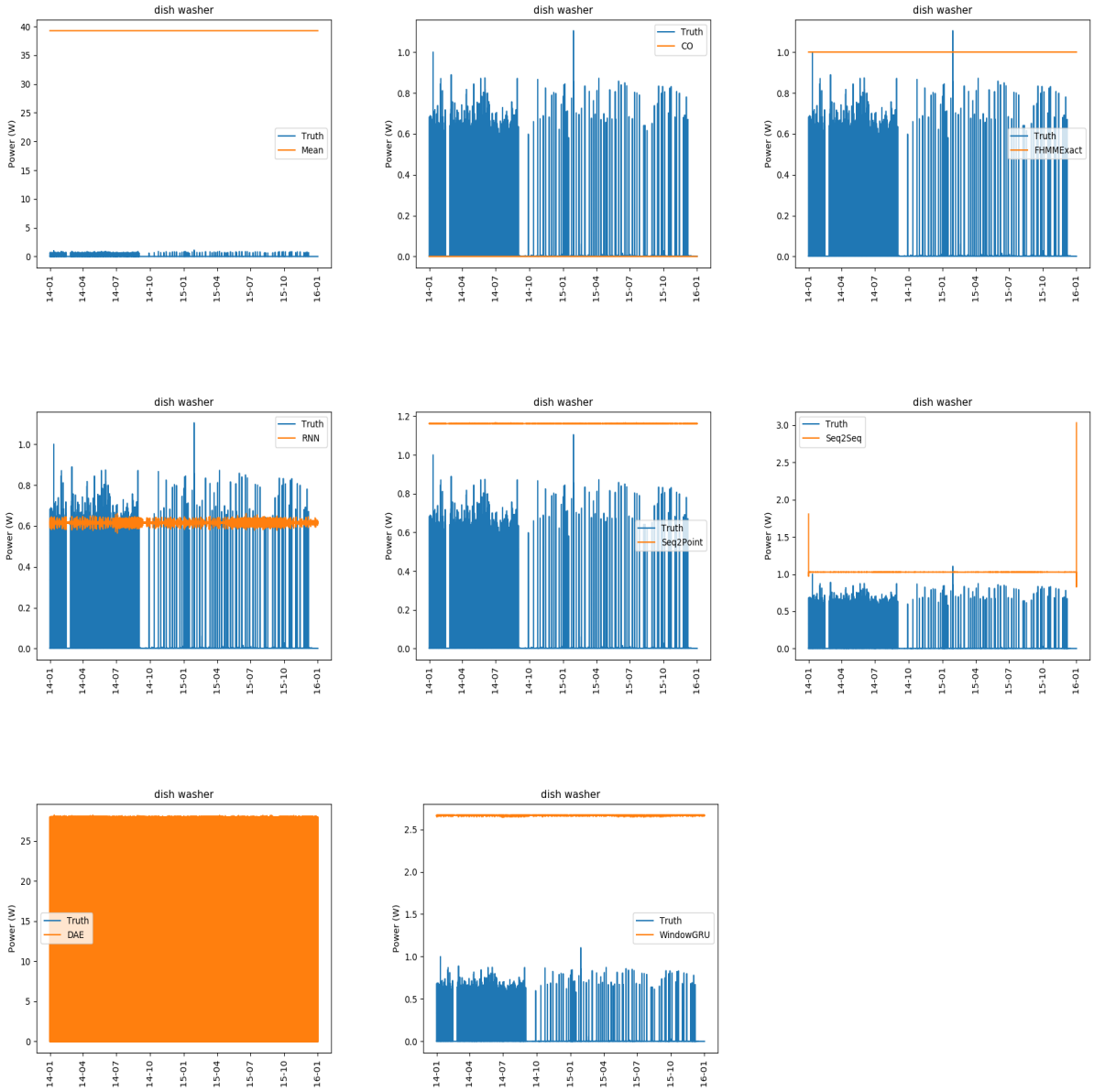
C.4 Train on UKDALE and REDD, test on SMART - Microwave



C.5 Train on UKDALE and REDD, test on SMART - Fridge



C.6 Train on UKDALE and REDD, test on SMART - Dish washer



D Python code

MA981-7-FY: MSc Dissertation jupyter notebook

Outputs of the code are not featured in the Jupyter notebook due to the requirement of using the High Performance Computing unit at the University of Essex which only allows .py files and .sh wrapped scripts

Imports the data sets, NILMTK API, NILMTK models and Matplotlib for graphs

In [ ]:
```
import os
cwd = os.getcwd()
cwd
redd = "/home/rpolea/redd_test.h5"
iawe = "/home/rpolea/iawe.h5"
ukdale = "/home/rpolea/ukdale.h5"
dred = "/home/rpolea/DRED.h5"
refit = "/home/rpolea/refit.h5"
smart = "/home/rpolea/SMART_03.h5"
from nilmtk.api import API
import warnings
warnings.filterwarnings("ignore")
from nilmtk_contrib.disaggregate import DAE,Seq2Point, Seq2Seq, RNN, WindowGRU, DSC, AFHMM
from nilmtk.disaggregate import FHMMExact, Mean, CO, Hart85
from matplotlib import pyplot as plt
```

Function created to take the model results, cycle through appliances and plot the power usage ground truth/predictions over time

In [ ]:
```
def plot_test(API, model,dataset_name):
    for i in API.gt_overall.columns:
        plt.figure()
        plt.plot(API.gt_overall[i],label='Truth')
        plt.plot(API.pred_overall[model][i], label = model)
        plt.xticks(rotation=90)
        plt.title(i)
        plt.legend(['Truth',model])
        plt.xlabel('Time')
        plt.ylabel('Power (W)')
        plt.savefig(dataset_name+model+i)
```

Trial 1: Training and testing the non-neural networks (NN) for UKDALE (NN and non-NN models have to be split due to processing times)

In [ ]:
```
ukdale_non_nn_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['fridge','kettle','computer','microwave','toaster',
                 'dish washer','broadband router', 'external hard disk', 'laptop computer'
  'methods': {
      'Mean': Mean({}),"CO": CO({}),'Hart85': Hart85({}),"FHMMExact": FHMMExact({}), "AFHM
  },
   'train': {
    'datasets': {
            'UKDALE': {
                'path': ukdale,
                                'buildings': {
                                1: {
```

```
                                        'start_time': '2016-04-26',
                                        'end_time': '2017-04-26'
                                }}}
        }},
        'test': {
        'datasets': {
                'UKDALE': {
                        'path': ukdale,
                        'buildings': {
                                2: {
                                        'start_time': '2013-02-17',
                                        'end_time': '2013-10-10'
                                },
                        }
        }
},
        'metrics':['f1score','mae','rmse','relative_error','r2score','nde','nep']
}
}
```

In [ ]:
```
non_nn_uk = API(ukdale_non_nn_api)
```

In [ ]:
```
for i in non_nn_uk.pred_overall: plot_test(non_nn_uk,i,'UKDALE')
```

Trial 1: Training and testing the neural networks (NN) for UKDALE (NN and non-NN models have to be split due to processing times)

In [ ]:
```
ukdale_nn_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['fridge','kettle','computer','microwave','toaster',
                 'dish washer','broadband router', 'external hard disk', 'laptop computer'
  'methods': {
      "RNN":RNN({'n_epochs':50,'batch_size':1024}),"Seq2Point":Seq2Point({'n_epochs':50,'b
      "Seq2Seq":Seq2Seq({'n_epochs':50,'batch_size':1024}),"DAE":DAE({'n_epochs':50,'batch
  },
   'train': {
    'datasets': {
            'UKDALE': {
                'path': ukdale,
                                'buildings': {
                                1: {
                                        'start_time': '2016-04-26',
                                        'end_time': '2017-04-26'
                                }}}
        }},
        'test': {
        'datasets': {
                'UKDALE': {
                        'path': ukdale,
                        'buildings': {
                                2: {
                                        'start_time': '2013-02-17',
                                        'end_time': '2013-10-10'
                                },
                        }
        }
},
```

```
        'metrics':['f1score','mae','rmse','relative_error','r2score','nde','nep']
}
}
```

In [ ]:
```
nn_uk = API(ukdale_nn_api)
```

In [ ]:
```
for i in nn_uk.pred_overall: plot_test(nn_uk,i,'UKDALE')
```

Trial 2: Training and testing the non-neural networks (NN) for REFIT (NN and non-NN models have to be split due to processing times)

In [ ]:
```
refit_co_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['television',
                 'dish washer', 'microwave', 'kettle'],
  'methods': {
      'Mean': Mean({}),"CO": CO({}),'Hart85': Hart85({}),"FHMMExact": FHMMExact({})
  },
   'train': {
    'datasets': {
            'REFIT': {
                'path': refit,
                                'buildings': {
                                3: {
                                        'start_time': '2013-12-17',
                                        'end_time': '2015-06-02'
                                },
                                5: {
                                        'start_time': '2013-12-17',
                                        'end_time': '2015-06-02'
                                },
                                9: {
                                        'start_time': '2013-12-17',
                                        'end_time': '2015-06-02'
                                }}}
        }},
        'test': {
        'datasets': {
                'REFIT': {
                        'path': refit,
                        'buildings': {
                                2: {
                                        'start_time': '2013-09-17',
                                        'end_time': '2015-05-28'
                                },
                        }
        }
},
        'metrics':['f1score','mae','rmse','relative_error','r2score','nde','nep']
}
}
```

In [ ]:
```
refit_co = API(refit_co_api)
```

In [ ]:
```
for i in refit_co.pred_overall: plot_test(refit_co,i,'REFIT')
```

Trial 2: Training and testing the neural networks (NN) for REFIT (NN and non-NN models have to be split due to processing times)

In [ ]:
```
refit_nn_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['television',
                 'dish washer', 'microwave', 'kettle', ],
  'methods': {
      "RNN":RNN({'n_epochs':50,'batch_size':1024}),"Seq2Point":Seq2Point({'n_epochs':50,'k
      "Seq2Seq":Seq2Seq({'n_epochs':50,'batch_size':1024}),"DAE":DAE({'n_epochs':50,'batch
  },
   'train': {
    'datasets': {
            'REFIT': {
                'path': refit,
                                'buildings': {
                                3: {
                                        'start_time': '2013-12-17',
                                        'end_time': '2015-06-02'
                                },
                                5: {
                                        'start_time': '2013-12-17',
                                        'end_time': '2015-06-02'
                                },
                                9: {
                                        'start_time': '2013-12-17',
                                        'end_time': '2015-06-02'
                                }}}
        }},
        'test': {
        'datasets': {
                'REFIT': {
                        'path': refit,
                        'buildings': {
                                2: {
                                        'start_time': '2013-09-17',
                                        'end_time': '2015-05-28'
                                },
                        }
        }
},
        'metrics':['f1score','mae','rmse','relative_error','r2score','nde','nep']
}
}
```

In [ ]:
```
refit_nn = API(refit_nn_api)
```

In [ ]:
```
for i in refit_nn.pred_overall: plot_test(refit_nn,i,'REFIT')
```

Trial 3: Training and testing the non-neural networks (NN) for DRED (NN and non-NN models have to be split due to processing times)

In [ ]:
```
dred_co_api = {
  'power': {
    'mains': ['apparent','active'],
```

```
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['cooker','electric heating element',
                 'fan','fridge', 'laptop computer','microwave','oven','sockets','televisio
  'methods': {
      'Mean': Mean({}),"CO": CO({}),'Hart85': Hart85({}),"FHMMExact": FHMMExact({})
  },
   'train': {
    'datasets': {
            'DRED': {
                'path': dred,
                                'buildings': {
                                1: {
                                        'start_time': '2015-07-05',
                                        'end_time': '2015-10-05'
                                }}}
        }},
        'test': {
        'datasets': {
                'DRED': {
                        'path': dred,
                        'buildings': {
                                1: {
                                        'start_time': '2015-10-06',
                                        'end_time': '2015-12-05'
                                },
                        }
        }
},
        'metrics':['f1score','mae','rmse','relative_error','r2score','nde','nep']
}
}
```

In [ ]:
```
dred_co = API(dred_co_api)
```

In [ ]:
```
for i in dred_co.pred_overall: plot_test(dred_co,i,'DRED')
```

Trial 3: Training and testing the neural networks (NN) for DRED (NN and non-NN models have to be split due to processing times)

In [ ]:
```
dred_nn_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['cooker','electric heating element',
                 'fan','fridge', 'laptop computer','microwave','oven','sockets','televisio
  'methods': {
      "RNN":RNN({'n_epochs':50,'batch_size':1024}),"Seq2Point":Seq2Point({'n_epochs':50,'b
      "Seq2Seq":Seq2Seq({'n_epochs':50,'batch_size':1024}),"DAE":DAE({'n_epochs':50,'batch
  },
   'train': {
    'datasets': {
            'DRED': {
                'path': dred,
                                'buildings': {
                                1: {
                                        'start_time': '2015-07-05',
                                        'end_time': '2015-10-05'
```

```
                                }}}
        }},
        'test': {
        'datasets': {
                'DRED': {
                        'path': dred,
                        'buildings': {
                                1: {
                                        'start_time': '2015-10-06',
                                        'end_time': '2015-12-05'
                                },
                        }
        }
},
        'metrics':['f1score','mae','rmse','relative_error','r2score','nde','nep']
}
}
```

In [ ]:
```
dred_nn = API(dred_nn_api)
```

In [ ]:
```
for i in dred_nn.pred_overall: plot_test(dred_nn,i,'DRED')
```

Trial 4: Training and testing the non-neural networks (NN) for REDD (NN and non-NN models have to be split due to processing times)

In [ ]:
```
redd_co_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['fridge','microwave',
                 'dish washer','washer dryer', 'light', 'sockets'],
  'methods': {
      'Mean': Mean({}),"CO": CO({}),'Hart85': Hart85({}), "FHMMExact": FHMMExact({})
  },
   'train': {
    'datasets': {
            'REDD': {
                'path': redd,
                                'buildings': {
                                1: {
                                        'start_time': '2011-04-18',
                                        'end_time': '2011-05-22'
                                },
                                3: {
                                        'start_time': '2011-04-18',
                                        'end_time': '2011-05-22'
                                },
                                5: {
                                        'start_time': '2011-04-18',
                                        'end_time': '2011-05-22'
                                }}}
        }},
        'test': {
        'datasets': {
                'REDD': {
                        'path': redd,
                        'buildings': {
                                2: {
                                        'start_time': '2011-04-18',
```

```
                                        'end_time': '2011-05-22'
                                    },
                            }
                    }
},
        'metrics':['f1score','mae','rmse','relative_error','r2score','nde','nep']
}
}
```

In [ ]:
```
redd_co = API(redd_co_api)
```

In [ ]:
```
for i in redd_co.pred_overall: plot_test(redd_co,i,'REDD')
```

Trial 4: Training and testing the neural networks (NN) for REDD (NN and non-NN models have to be split due to processing times)

In [ ]:
```
redd_nn_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['fridge','microwave',
                 'dish washer','washer dryer', 'light', 'sockets'],
  'methods': {
      "RNN":RNN({'n_epochs':50,'batch_size':1024}),"Seq2Point":Seq2Point({'n_epochs':50,'k
      "Seq2Seq":Seq2Seq({'n_epochs':50,'batch_size':1024}),"DAE":DAE({'n_epochs':50,'batch
  },
   'train': {
    'datasets': {
            'REDD': {
                'path': redd,
                                'buildings': {
                                1: {
                                        'start_time': '2011-04-18',
                                        'end_time': '2011-05-22'
                                },
                                3: {
                                        'start_time': '2011-04-18',
                                        'end_time': '2011-05-22'
                                },
                                5: {
                                        'start_time': '2011-04-18',
                                        'end_time': '2011-05-22'
                                }}}
        }},
        'test': {
        'datasets': {
                'REDD': {
                        'path': redd,
                        'buildings': {
                                2: {
                                        'start_time': '2011-04-18',
                                        'end_time': '2011-05-22'
                                },
                        }
        }
},
        'metrics':['f1score','mae','rmse','relative_error','r2score','nde','nep']
}
}
```

In [ ]:
```
redd_nn = API(redd_nn_api)
```

In [ ]:
```
for i in redd_nn.pred_overall: plot_test(redd_nn,i,'REDD')
```

Trial 5: Training and testing the non-neural networks (NN) for SMART (NN and non-NN models have to be split due to processing times)

In [ ]:
```
smart_co_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['fridge','boiler',
                 'dish washer','washing machine', 'sockets'],
  'methods': {
      'Mean': Mean({}),"CO": CO({}),"FHMMExact": FHMMExact({}),
  },
   'train': {
    'datasets': {
            'SMART': {
                'path': smart,
                                'buildings': {
                                1: {
                                        'start_time': '2013-12-31',
                                        'end_time': '2016-12-31'
                                }}}
        }},
        'test': {
        'datasets': {
                'SMART': {
                        'path': smart,
                        'buildings': {
                                3: {
                                        'start_time': '2013-12-31',
                                        'end_time': '2015-12-31'
                                },
                        }
        }
},
        'metrics':['f1score','mae','rmse','relative_error','r2score','nde','nep']
}
}
```

In [ ]:
```
smart_co = API(smart_co_api)
```

In [ ]:
```
for i in smart_co.pred_overall: plot_test(smart_co,i,'SMART')
```

Trial 5: Training and testing the neural networks (NN) for SMART (NN and non-NN models have to be split due to processing times)

In [ ]:
```
smart_same_nn_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
```

```
  'appliances': ['fridge','boiler',
                 'dish washer','washing machine', 'sockets'],
  'methods': {
      "RNN":RNN({'n_epochs':50,'batch_size':1024}),"Seq2Point":Seq2Point({'n_epochs':50,'b
      "Seq2Seq":Seq2Seq({'n_epochs':50,'batch_size':1024}),"DAE":DAE({'n_epochs':50,'batch
  },
   'train': {
    'datasets': {
            'SMART': {
                'path': smart,
                                'buildings': {
                                1: {
                                        'start_time': '2013-12-31',
                                        'end_time': '2016-12-31'
                                }}}
        }},
        'test': {
        'datasets': {
                'SMART': {
                        'path': smart,
                        'buildings': {
                                3: {
                                        'start_time': '2013-12-31',
                                        'end_time': '2015-12-31'
                                },
                        }
        }
},
        'metrics':['f1score','mae','rmse','relative_error','r2score','nde','nep']
}
}
```

In [ ]:
```
smart_same_nn = API(smart_same_nn_api)
```

In [ ]:
```
for i in smart_same_nn.pred_overall: plot_test(smart_same_nn,i,'SMART')
```

Stage 2

Trial 1: Training and testing the non-neural networks (NN) for REDD and SMART (NN and non-NN models have to be split due to processing times)

In [ ]:
```
redd_smart_co_fhmm_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['fridge','microwave',
                 'dish washer'],
  'methods': {
      'Mean': Mean({}),"CO": CO({}), 'FHMMExact': FHMMExact({}),
  },
 'train': {
    'datasets': {
      'REDD': {
        'path': redd,
        'buildings': {
              1: {
                    'start_time': '2011-04-18',
                    'end_time': '2011-05-24'
          }
```

```
        }
      }
    }
  },

  'test': {
    'datasets': {
      'SMART': {
        'path': smart,
        'buildings': {
              2: {
                    'start_time': '2013-12-31',
                                          'end_time': '2016-12-31'
          }
        }
      }
    },
    'metrics': ['f1score','mae','rmse','relative_error','r2score','nde','nep']
  }
}
```

In [ ]:
```
redd_smart_co_fhmm = API(redd_smart_co_fhmm_api)
```

In [ ]:
```
for i in redd_smart_co_fhmm.pred_overall: plot_test(redd_smart_co_fhmm,i,'S2REDDSMART_CO_F
```

Trial 1: Training and testing the neural networks (NN) for REDD and SMART (NN and non-NN models have to be split due to processing times)

In [ ]:
```
redd_smart_nn_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['fridge','microwave',
                 'dish washer'],
  'methods': {
      "RNN":RNN({'n_epochs':50,'batch_size':1024}),"Seq2Point":Seq2Point({'n_epochs':50,'b
      "Seq2Seq":Seq2Seq({'n_epochs':50,'batch_size':1024}),"DAE":DAE({'n_epochs':50,'batch
  },
 'train': {
    'datasets': {
      'REDD': {
        'path': redd,
        'buildings': {
              1: {
                    'start_time': '2011-04-18',
                    'end_time': '2011-05-24'
          }
        }
      }
    }
  },

  'test': {
    'datasets': {
      'SMART': {
        'path': smart,
        'buildings': {
              2: {
                    'start_time': '2013-12-31',
```

```
                                        'end_time': '2016-12-31'
            }
          }
        }
    },
    'metrics': ['f1score','mae','rmse','relative_error','r2score','nde','nep']
  }
}
```

In [ ]:
```
redd_smart_nn = API(redd_smart_nn_api)
```

In [ ]:
```
for i in redd_smart_nn.pred_overall: plot_test(redd_smart_nn,i,'S2REDDSMART_NN_')
```

Trial 2: Training and testing the non-neural networks (NN) for UKDALE and REFIT (NN and non-NN models have to be split due to processing times)

In [ ]:
```
uk_refit_co_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['fridge','microwave',
                 'dish washer'],
  'methods': {
      'Mean': Mean({}),"CO": CO({}),'Hart85': Hart85({}), 'FHMMExact': FHMMExact({}),
  },
 'train': {
    'datasets': {
      'UKDALE': {
        'path': ukdale,
        'buildings': {
              2: {
                'start_time': '2013-04-16',
                'end_time': '2013-10-10'
              },

        }
      }
    }
  },

  'test': {
    'datasets': {
      'REFIT': {
        'path': refit,
        'buildings': {
              2: {
                    'start_time': '2013-09-17',
                                        'end_time': '2015-05-28'
          }
        }
      }
    },
    'metrics': ['f1score','mae','rmse','relative_error','r2score','nde','nep']
  }
}
```

In [ ]:
```
uk_refit_co = API(uk_refit_co_api)
```

In [ ]:
```
for i in uk_refit_co.pred_overall: plot_test(uk_refit_co,i,'S2UKREFIT_CO_FHMM_')
```

Trial 2: Training and testing the neural networks (NN) for UKDALE and REFIT (NN and non-NN models have to be split due to processing times)

In [ ]:
```
uk_refit_nn_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['fridge','microwave',
                 'dish washer'],
  'methods': {
      "RNN":RNN({'n_epochs':50,'batch_size':1024}),"Seq2Point":Seq2Point({'n_epochs':50,'b
      "Seq2Seq":Seq2Seq({'n_epochs':50,'batch_size':1024}),"DAE":DAE({'n_epochs':50,'batch
  },
 'train': {
    'datasets': {
      'UKDALE': {
        'path': ukdale,
        'buildings': {
              2: {
                'start_time': '2013-04-16',
                'end_time': '2013-10-10'
              },

        }
      }
    }
  },

  'test': {
    'datasets': {
      'REFIT': {
        'path': refit,
        'buildings': {
              2: {
                    'start_time': '2013-09-17',
                                  'end_time': '2015-05-28'
          }
        }
      }
    },
    'metrics': ['f1score','mae','rmse','relative_error','r2score','nde','nep']
  }
}
```

In [ ]:
```
uk_refit_nn = API(uk_refit_nn_api)
```

In [ ]:
```
for i in uk_refit_nn.pred_overall: plot_test(uk_refit_nn,i,'S2UKREFIT_NN_')
```

Stage 3

Trial 1: Training and testing the non-neural networks (NN) for UKDALE and REFIT (NN and non-NN models have to be split due to processing times)

In [ ]:
```
uk_redd_refit_co_api = {
```

```
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['fridge','microwave',
                 'dish washer'],
  'methods': {
      'Mean': Mean({}),"CO": CO({}),'Hart85': Hart85({}), 'FHMMExact': FHMMExact({}),
  },
 'train': {
    'datasets': {
      'UKDALE': {
        'path': ukdale,
        'buildings': {
              2: {
                'start_time': '2013-04-16',
                'end_time': '2013-10-10'
              },

        }
      },
      'REDD': {
        'path': redd,
        'buildings': {
              1: {
                    'start_time': '2011-04-18',
                    'end_time': '2011-05-24'
          }
        }
      }
    }
  },

  'test': {
    'datasets': {
      'REFIT': {
        'path': refit,
        'buildings': {
              2: {
                    'start_time': '2013-09-17',
                                  'end_time': '2015-05-28'
          }
        }
      }
    },
    'metrics': ['f1score','mae','rmse','relative_error','r2score','nde','nep']
  }
}
```

In [ ]:
```
uk_redd_refit_co = API(uk_redd_refit_co_api)
```

In [ ]:
```
for i in uk_redd_refit_co.pred_overall: plot_test(uk_redd_refit_co,i,'S3UKREDDREFIT_CO')
```

Trial 1: Training and testing the neural networks (NN) for UKDALE and REFIT (NN and non-NN models have to be split due to processing times)

In [ ]:
```
uk_redd_refit_nn_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
```

```
  },
  'sample_rate': 60,
  'appliances': ['fridge','microwave',
                 'dish washer'],
  'methods': {
      "RNN":RNN({'n_epochs':50,'batch_size':1024}),"Seq2Point":Seq2Point({'n_epochs':50,'b
      "Seq2Seq":Seq2Seq({'n_epochs':50,'batch_size':1024}),"DAE":DAE({'n_epochs':50,'batch
  },
 'train': {
    'datasets': {
      'UKDALE': {
        'path': ukdale,
        'buildings': {
              2: {
                'start_time': '2013-04-16',
                'end_time': '2013-10-10'
              },

        }
      },
      'REDD': {
        'path': redd,
        'buildings': {
              1: {
                    'start_time': '2011-04-18',
                    'end_time': '2011-05-24'
          }
        }
      }
    }
  },

  'test': {
    'datasets': {
      'REFIT': {
        'path': refit,
        'buildings': {
              2: {
                    'start_time': '2013-09-17',
                                  'end_time': '2015-05-28'
          }
        }
      }
    },
    'metrics': ['f1score','mae','rmse','relative_error','r2score','nde','nep']
  }
}
```

In [ ]:
```
uk_redd_refit_nn = API(uk_redd_refit_nn_api)
```

In [ ]:
```
for i in uk_redd_refit_nn.pred_overall: plot_test(uk_redd_refit_nn,i,'S3UKREDDREFIT_NN_')
```

Trial 2: Training and testing the non-neural networks (NN) for UKDALE AND REDD (NN and non-NN models have to be split due to processing times)

In [ ]:
```
uk_redd_smart_co_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
```

```
  'appliances': ['fridge','microwave',
                 'dish washer'],
  'methods': {
      'Mean': Mean({}),"CO": CO({}), 'FHMMExact': FHMMExact({}),
  },
 'train': {
    'datasets': {
      'UKDALE': {
        'path': ukdale,
        'buildings': {
              2: {
                'start_time': '2013-04-16',
                'end_time': '2013-10-10'
              },

        }
      },
      'REDD': {
        'path': redd,
        'buildings': {
              1: {
                    'start_time': '2011-04-18',
                    'end_time': '2011-05-24'
          }
        }
      }
    }
  },

  'test': {
    'datasets': {
      'SMART': {
        'path': smart,
        'buildings': {
              2: {
                    'start_time': '2013-12-31',
                                  'end_time': '2016-12-31'
          }
        }
      }
    },
    'metrics': ['f1score','mae','rmse','relative_error','r2score','nde','nep']
  }
}
```

In [ ]:
```
uk_redd_smart_co = API(uk_redd_smart_co_api)
```

In [ ]:
```
for i in uk_redd_smart_co.pred_overall: plot_test(uk_redd_smart_co,i,'S3UKREDDSMART_CO')
```

Trial 2: Training and testing the neural networks (NN) for UKDALE and REDD (NN and non-NN models have to be split due to processing times)

In [ ]:
```
uk_redd_smart_nn_api = {
  'power': {
    'mains': ['apparent','active'],
    'appliance': ['apparent','active']
  },
  'sample_rate': 60,
  'appliances': ['fridge','microwave',
                 'dish washer'],
  'methods': {
```

```
      "RNN":RNN({'n_epochs':50,'batch_size':1024}),"Seq2Point":Seq2Point({'n_epochs':50,'b
      "Seq2Seq":Seq2Seq({'n_epochs':50,'batch_size':1024}),"DAE":DAE({'n_epochs':50,'batch
  },
 'train': {
    'datasets': {
      'UKDALE': {
        'path': ukdale,
        'buildings': {
              2: {
                'start_time': '2013-04-16',
                'end_time': '2013-10-10'
              },

        }
      },
      'REDD': {
        'path': redd,
        'buildings': {
              1: {
                    'start_time': '2011-04-18',
                    'end_time': '2011-05-24'
          }
        }
      }
    }
  },

  'test': {
    'datasets': {
      'SMART': {
        'path': smart,
        'buildings': {
              2: {
                    'start_time': '2013-12-31',
                                  'end_time': '2016-12-31'
          }
        }
      }
    },
    'metrics': ['f1score','mae','rmse','relative_error','r2score','nde','nep']
  }
}
```

In [ ]:
```
uk_redd_smart_nn = API(uk_redd_smart_nn_api)
```

In [ ]:
```
for i in uk_redd_smart_nn.pred_overall: plot_test(uk_redd_smart_nn,i,'S3UKREDDSMART_NN')
```

The authors acknowledge the use of the High Performance Computing Facility (Ceres) and its associated support services at the University of Essex in the completion of this work.