NILMTK-Contrib: Towards reproducible state-of-the-art energy disaggregation

Rithwik Kukunuri[1*] , **Nipun Batra**[1] , **Ayush Pandey**[2] , **Raktim Malakar**[3] , **Rajat Kumar**[4] , **Odysseas Krystalakos**[5] , **Mingjun Zhong**[6] , **Paulo Meira**[7] and **Oliver Parson**[8]
[1]IIT Gandhinagar, India
[2]NIT Agartala, India
[3]IIEST Shibpur, India
[4]DAIICT, India
[5]Aristotle University of Thessaloniki,Greece
[6]University of Lincoln, UK
[7]University of Campinas, Brazil
[8]Centrica Hive Limited, UK
{kukunuri.sai, nipun.batra}@iitgn.ac.in, {ayush20pandey,raktimmalakar2015}@gmail.com, 201811024@daiict.ac.in, odysseaskrst@gmail.com, MZhong@lincoln.ac.uk, MZhong@lincoln.ac.uk, oliver.parson@hivehome.com

Abstract

Research shows that providing an appliance-wise energy breakdown can help users save up to 15% of their energy bills. Non-intrusive load monitoring (NILM) or energy disaggregation is the task of estimating the household energy measured at the aggregate level for each constituent appliances in the household. The problem was first was introduced in the 1980s by Hart. Over the past three decades, NILM has been an extensively researched topic by researchers. NILMTK was introduced in 2014 to the NILM community in order to motivate reproducible research. Even after the introduction of the NILMTK toolkit to the community, there has been a little contribution of recent state-of-the-art algorithms back to the toolkit. In this paper, we propose a new disaggregation API, which further simplifies the process for the rapid comparison of different state-of-the-art algorithms across a wide range of datasets and algorithms. We also propose a new rewrite for writing the new disaggregation algorithms for NILMTK, which is similar to Scikit-learn. We demonstrate the power of the new API by conducting various complex experiments using the API.

1 Introduction

Buildings consume roughly one-thirds of the total energy consumption across the world [Pérez-Lombard *et al.*, 2008]. Prior research shows that providing households with a per-appliance energy consumption can help them save upto 15% on their energy bills [Darby and others, 2006]. Non-Intrusive load monitoring is the task of estimating appliance-wise energy consumption using aggregate reading. The estimated appliance reading can be used by the household for understanding the appliances of their home and can make decisions for optimizing the household's power consumption. The problem was initially studied by George Hart [Hart, 1985] in the 1980s. Recently the problem gained more recognition owing to the smart meter availability and the environmental impact of the problem.

Over the past three decades, there has been extensive research on the NILM topic by the research community. Many of the recent algorithmic developments in NILM were not reproducible due to the lack of open-source implementations of them. Researchers were using several baseline algorithms due to the lack of online implementations of the recent developments. The metrics and datasets used by the algorithms were also different, resulting in an uneven comparison of the algorithms. Due to the above problems, it was hard for a new researcher to decide which algorithm is the best.

In order to address this problem, NILMTK [Batra *et al.*, 2014] was proposed in 2014, for motivating the researchers to use the toolkit for developing the algorithms and using the toolkit for reproducing the results across a variety of datasets and metrics. Despite the introduction of the toolkit in 2014, there has been a little contribution from the NILM community for adding new algorithms that are supported by NILMTK. In order to develop an algorithm for the previous NILMTK, the researcher needed to have an understanding of Meters, MeterGroups, Buildings, and Datasets, which are critical objects in NILMTK. In this paper, we propose a new rewrite of the NILM algorithms which can be developed by any user using the knowledge of Pandas. We believe that this step will encourage more users to develop new algorithms for the toolkit.

In this paper, we also propose a new Disaggregation API, which can be used for rapid experimentation by the users against a benchmark of datasets, algorithms, and metrics that are provided in NILMTK. Researchers can now develop their algorithms using the knowledge of Pandas. In this paper, we demonstrate the power of the API by using three baseline

[*]Contact Author

```
 1 experiment = {
 2   'power': {'mains': ['active'], 'appliance': ['active']},
 3   'sample_rate': 60, 'artificial_aggregate':False,
 4   'appliances': ['fridge', 'washing machine'],
 5   'methods': {'CO': {},
 6     'FHMM_EXACT': {'num_of_states': 2},
 7     'Seq2Point': {
 8       'n_epochs': 1,
 9       'pre-processing': {
10         'appliance_params': {
11           'washing machine': {
12             'mean': 400,'std': 700},
13           'fridge': {
14             'mean': 200,'std': 400 },
15         }
16       },
17     }
18     },
19   'train': {
20     'datasets': {
21       'REDD': {
22         'path': '/data/REDD/redd.h5',
23         'buildings': {
24           1: {'start_time': '2011-04-01','end_time': '2011-04-30'},
25           2: {'start_time': '2011-04-01','end_time': '2011-04-30'}
26         }
27       }
28     }
29   },
30   'test': {
31     'datasets': {
32       'IAWE': {
33         'path': '/data/IAWE/iawe.h5',
34         'buildings': {
35           1: {'start_time': '2015-08-05','end_time': '2015-08-10'}
36         }
37       }
38     },
39     'metrics': ['mae', 'rmse']
40   }
41 }
```

Listing 1: `ExperimentAPI`: Simplifying the definition of algorithm comparison experiments

algorithms and nine recent disaggregation algorithms on the task of transfer learning.

This paper is a shortened version of our accepted Buildsys '19 paper [Batra *et al.*, 2019] .

2 Experiment API

In this section, we describe the new Disaggregation API provided by NILMTK. The new rapid experimentation API that we have introduced `ExperimentAPI`; can lower the entry barrier for developing NILM algorithms.

2.1 Experiment Interface

Listing 1 shows the new experiment interface implemented in NILMTK. This interface is more user-friendly and can also be used by non-NILM experts for running different algorithms. The API has been designed such that it provides the maximum flexibility to the user for developing an algorithm and minimizes the time taken for benchmarking against the existing algorithms. Listing 1 describes an experiment where NILM algorithms such as Seq2Point, FHMM-EXACT and Combinatorial Optimization are trained and tested in the REDD dataset. The following lines explain the the interface in detail

- mains and appliances use active power (L2).
- with a sampling rate of 60 seconds and not using artificial aggregate, i.e. using true aggregate reading (L3).
- for appliances: fridge and washing machine (L4).
- three algorithms are used for disaggregation (CO, FHMM and Seq2Point – L5, 6, 7 respectively) and their corresponding parameters are specified (L6 for FHMM and L8-14 for Seq2Point).
- training parameters are specified on L19-29, where the different training data sets are specified: REDD data set specified from L21-26, where the path for the data set is specified in L22 and the start and end time for building number 1 and 2 are specified in L24 and 25.
- the test parameters are added in a similar format to the training parameters from L31-38.
- the set of evaluation metrics are on L39.

3 NILMTK-contrib

This section describes the algorithms that are a part of the NILMTK-CONTRIB repository. Many of the recent state-of-the-art algorithms in the NILM domain, have been made compatible with NILMTK and are added here. It should be noted that we have not proposed the following algorithms, rather, created a unified toolkit where everyone can use them.

3.1 Mean

The Mean algorithm estimates the output usage of an appliance to be the mean computed over the training data. Mean algorithm can be used as a strong baseline for evaluating the power of an algorithm against a sparse appliance.

3.2 Edge Detection

Hart's algorithm [Hart, 1985] is one of the most used baseline NILM methods. The algorithm uses edge detection over a 2-D signal. Edge refers to the difference in power between two adjacent timestamps, and the edges correspond to transient states and steady states. Although the algorithm is unsupervised, we use training data, to choose a mapping from edges to appliances such that accuracy is maximized.

3.3 Combinatorial Optimisation

The combinatorial optimisation (CO) algorithm [Hart, 1992] has served as a baseline algorithm in the NILM literature [Uttama Nambi *et al.*, 2015; Batra *et al.*, 2015] . The CO algorithm is similar to the well-studied knapsack and subset sum problem. The main assumption in CO is that each appliance can be in a given state (1 of K where K is a small number), where each state has an associated power consumption. The goal of the algorithm is to assign states to appliances in a way that the difference between the household aggregate reading and the sum of power usage of the different appliances is minimised. CO's time complexity is exponential in the number of appliances and thus does not scale well.

3.4 Discriminative Sparse Coding

Sparse coding estimates the usage of every appliance, by representing the aggregate usage as a product of over-complete bases and activations. The bases are computed by first learning the bases for every appliance individually and then concatenating them. Then, the bases are trained using a discriminative method [Kolter *et al.*, 2010] for resulting in activations that result in a solution closer to optimal activations.

3.5 Exact FHMM

For every appliance, we learn a hidden Markov model, which returns the hidden states and the transition probabilities. Each state of the appliance is associated with a power value. The models from different appliances are combined to create a Super HMM, which can represent the states of every appliance. The aggregate signal is passed through the Super HMM, which returns the states of every appliance present in it. The usage of the appliance during a particular timestamp is the same as the power consumed during the particular state.

3.6 Approximate FHMM

The Exact FHMM assigns a discrete power to every appliance using the Super HMM. The approximate FHMM, assigns continuous power to every appliance, by using an extension over the Exact FHMM.

3.7 Approximate FHMM-SAC

The AFHMM-SAC [Zhong *et al.*, 2014] is an extension over the Approximate FHMM, where the model has a signal aggregate constraint. The constraint says that the usage of an appliance over a particular time period is less than a particular constant, which is computed using training data.

3.8 Denoising Auto Encoder

Denoising Auto Encoder [Kelly and Knottenbelt, 2015] takes a window of aggregate reading as an input and outputs the appliance reading for every timestamp in the window. This was on the first few algorithms that used neural networks for the task of disaggregation.

3.9 RNN

RNN was one of the models proposed in Neural-NILM paper [Kelly and Knottenbelt, 2015] . The model used Bidirectional LSTM's for processing the windows and outputs a single value corresponding to the appliance usage.

3.10 Seq2Point

Seq2Point [Zhang *et al.*, 2018] uses 1-Dimensional convolutions to process the input windows and outputs the reading corresponding to the mid-point of the window.

3.11 Seq2Seq

Seq2Seq [Zhang *et al.*, 2018] uses 1-Dimensional convolutions to process the input windows and outputs the reading corresponding to the whole window.

3.12 Online GRU

WindowGRU [Krystalakos *et al.*, 2018] uses Bidirectional Gated Recurrent Units, to process the input windows and outputs the appliance usage at the end of the window.

4 Experimental results

We now describe experiments which we conducted using the new Experiment API. An extensive discussion on the discussion of the results is beyond the scope of this paper.

4.1 Settings

The neural network algorithms were run on virtual machines with 2x8 GB vRAM Nvidia Tesla M60 GPU's. The CPU intensive models such as DSC, Approx-FHMM, and Approx-FHMM with SAC were run on computers with 100 cores. The sample period was 60 seconds. All of the neural networks were run 50 epochs, except OnlineGRU, which was trained for 30 epochs.

The notebooks can be found at https://github.com/nilmtk/buildsys2019-paper-notebooks

4.2 Train and test across buildings from the same data set

In this experiment, we train and test across multiple buildings from the Dataport data set. We trained the models on 10 buildings and then tested them on 5 unseen buildings. The training duration was 14 days and the testing duration was 7 days, with the models training and disaggregating 4 appliances for each building.

Algorithms	Fridge	Air Conditioner	Electric Furnace	Washing Machine
Mean	63.3±07.7	224.8±16.4	81.5±01.6	5.07±00.8
Edge detection	41.1±18.1	86.8±30.5	30.2±11.2	4.8±01.3
CO	65.7±42.3	98.5±85.7	56.9±55.4	105±19.0
DSC	78.4±56.5	71.5±36.0	39.1±17.9	6.5±05.7
ExactFHMM	66.7±23.5	45.5±44.6	95.3±110.5	59.9±17.5
ApproxFHMM	63.8±08.0	139.9±130.2	**26.5±12.0**	30.7±21.3
FHMM+SAC	59.2 ±05.7	97.0±40.3	35.1±19.0	3.8±00.7
DAE	32.2±11.8	39.3±27.9	29.4±15.3	3.1±01.6
RNN	38.4±07.9	46.6±30.6	33.9±20.6	3.5±01.2
Seq2Seq	28.1±09.5	32.3±25.2	27.9±15.3	**2.3±01.2**
Seq2Point	**23.5±12.1**	**24.8±20.9**	27.5±15.0	2.4±00.9
OnlineGRU	28.8±11.4	25.3±17.1	34.5±15.0	3.0±01.4

Table 1: MAE Mean ± Std. Error: train/test on different set of buildings, same data set

The main results for this experiment can be found in Table 1. The neural network models perform comparably, with Seq2Point and Seq2Seq achieving the best performance. Interestingly, the edge detection algorithm achieves good performance for fridges. This can be explained by the fact that for simple appliances with a single ON-OFF component (in this case, a compressor-controlled duty cycle), simple edge detection is likely to work well. This is an important finding, in that it appears that only more complex appliances motivate the use of complex disaggregation algorithms. The mean algorithm performs reasonably well for washing machine. This finding suggests that accurately disaggregating sparsely used appliances is still non-trivial for modern algorithms. However, a different metric that is better suited to handle class imbalance would reveal the inaccuracy of the Mean model.

4.3 Train and test across multiple buildings from multiple data sets across data sets

In this experimental setup, we trained models on 2 appliances across a building from UK-DALE and then tested them on 1 building from DRED and 1 building from REDD. The total training duration was 2 months, and the testing duration was 10 days for each building. The previous version of NILMTK,

Algorithms	REDD - Home 1		DRED - Home 1	
	Fridge	Washing Machine	Fridge	Washing Machine
Mean	62.3	47.2	43.4	25.6
Edge detection	37.0	57.1	21.8	40.7
CO	99.3	171.1	45.9	47.2
DSC	61.5	48.9	34.3	12.1
ExactFHMM	95.9	179.6	32.3	19.1
ApproxFHMM	67.0	227.9	34.6	94.5
FHMM+SAC	48.1	30.4	31.1	19.0
DAE	41.7	50.1	**16.9**	3.8
RNN	50.9	**19.2**	27.8	7.3
Seq2Seq	40.9	23.3	18.5	**2.9**
Seq2Point	44.1	25.5	17.1	3.1
OnlineGRU	**36.4**	29.5	24.3	6.9

Table 2: MAE: train/test across multiple buildings and data sets

Algorithms	True Aggregate		Artificial Aggregate	
	Fridge	AC	Fridge	AC
Mean	43±04	176±40	43±04	176±40
Edge Detection	43±04	156±31	15±06	89±65
CO	104±08	90±34	18±03	15±07
DSC	67±05	130±29	53±14	55±20
ExactFHMM	56±10	83±16	14±01	24±03
ApproxFHMM	46±06	210±78	41±07	77±24
FHMM+SAC	32±04	132±47	36±07	125±28
DAE	22±05	34±08	14±01	8±02
RNN	31±01	78±24	11±01	10±03
Seq2Point	**14±02**	**20±06**	**5±01**	**4 ±01**
Seq2Seq	16±02	22±05	9±01	9±02
WindowGRU	20±04	23±09	8±02	7±03

Table 3: MAE: train/test across same buildings with true and artificial aggregate

needed the user to train a model for every algorithm explicitly and then compute the results for each of them. The API for generating the results for this experiment is similar to Listing 1.

Clearly, all the recent neural network based algorithms such as DAE, Seq2Point, Seq2Seq, and OnlineGRU perform the best in each of the cases. Another observation is that the difference in errors on the REDD dataset and the DRED dataset. The best error on the DRED dataset is much lower than the corresponding error on the REDD dataset. Since the algorithm was also trained on a European dataset, the model is generating better predictions for the DRED. Also, the washing machine error on the DRED dataset it very low, choosing another metric such as F1-Score might be useful to reveal the inaccuracies of the models. The API can also be useful for doing experiments in the transfer learning domain [Batra *et al.*, 2018] .

4.4 Train and test on artificial aggregate

In the this experiment, we train and test on 2 buildings from the Dataport data set, using true (obtained from smart meter) and artificial aggregate (calculated by summing the power readings of the appliances to be disaggregated). The artificial aggregate does not contain either structured noise from appliances which were not sub-metered or unstructured noise contributed by the mains sensor hardware. This scenario is often used for algorithm comparisons despite its lack of realism. This is due to the requirement for training data to be available for all appliances present in the aggregate signal, which is a common issue in data sets due to the practical difficultly in sub-metering all appliances within a building. The training was done on the first 20 days and the testing was done on the next 7 days for each building. The 2 most commonly used appliances were chosen for disaggregation.

Table 3 shows the main result where we can notice the superior disaggregation performance on the artificial aggregate for all algorithms. The only exception being the mean algorithm, that is independent of the aggregate data. The performance of neural networks on the air conditioner has improved significantly with artificial aggregate. However, the most significant improvement comes from the FHMM variants. This can be explained by the fact that FHMM variants have a state space that is exponential in the number of appliances and FHMM can explain the noise in true aggregate via a wrong appliance state space combination. In the absence of noise, the probability of estimating the correct state space combination is much more likely. This experiment represents the ideal scenario for energy disaggregation and might be prevalent in geographies where majority of the energy consumption can be attributed to a small set of appliances.

5 Conclusion

In this paper, we have have described two key improvements to NILMTK; a rewritten model interface to simplify authoring of new disaggregation algorithms, and a new experiment API through which algorithmic comparisons can be specified with relatively little model knowledge. In addition, we have introduced NILMTK-contrib, a new repository containing 3 benchmarks and 9 modern disaggregation algorithms. Furthermore, we have demonstrated these contributions through the most comprehensive algorithmic comparison to date. Taken together, these toolkit contributions enable empirical evaluations to be easily reproduced, therefore increasing the rate of progress within the field.

In the short-term, future work will focus on an exhaustive empirical evaluation of the algorithms presented in NILMTK-contrib across all publicly available data sets and a range of accuracy metrics. Longer-term future work will include collaboration with the community to ensure new algorithmic advances are incorporated within the NILMTK-contrib repository. In addition, such algorithms will be continuously evaluated in a range of pre-defined scenarios to produce an ongoing NILM competition.

One of the future directions for NILM is to disaggregate sparsely used appliances such as washing machine or microwave. The models tend to predict zero for all the timestamps. Neural network based models are accurate for NILM, but they are computationally very expensive, hence in order to do real-time disaggregation at a huge scale, one needs access to powerful GPU based systems. Hence, we need to explore techniques that can optimize these neural networks and reduce the computation power consumed by them.

References

[Batra *et al.*, 2014] Nipun Batra, Jack Kelly, Oliver Parson, Haimonti Dutta, William Knottenbelt, Alex Rogers, Amarjeet Singh, and Mani Srivastava. NILMTK: an open source toolkit for non-intrusive load monitoring. In *Proceedings of the 5th international conference on Future energy systems*, pages 265–276. ACM, 2014.

[Batra *et al.*, 2015] Nipun Batra, Amarjeet Singh, and Kamin Whitehouse. If you measure it, can you improve it? exploring the value of energy disaggregation. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 191–200. ACM, 2015.

[Batra *et al.*, 2018] Nipun Batra, Yiling Jia, Hongning Wang, and Kamin Whitehouse. Transferring decomposed tensors for scalable energy breakdown across regions. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Batra *et al.*, 2019] Nipun Batra, Rithwik Kukunuri, Ayush Pandey, Raktim Malakar, Rajat Kumar, Odysseas Krystalakos, Mingjun Zhong, Paulo Meira, and Oliver Parson. Towards reproducible state-of-the-art energy disaggregation. In *Proceedings of the 6th ACM International Conference on Embedded Systems for Energy-Efficient Built Environments (BuildSys' 19). ACM, New York, NY, USA*. ACM, November 2019.

[Darby and others, 2006] Sarah Darby et al. The effectiveness of feedback on energy consumption. *A Review for DEFRA of the Literature on Metering, Billing and direct Displays*, 486(2006):26, 2006.

[Hart, 1985] George W Hart. Prototype nonintrusive appliance load monitor. In *MIT Energy Laboratory Technical Report, and Electric Power Research Institute Technical Report*. 1985.

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

[Kelly and Knottenbelt, 2015] Jack Kelly and William Knottenbelt. 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*, BuildSys '15, pages 55–64, New York, NY, USA, 2015. ACM. event-place: Seoul, South Korea.

[Kolter *et al.*, 2010] J. Z. Kolter, Siddharth Batra, and Andrew Y. Ng. Energy Disaggregation via Discriminative Sparse Coding. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1153–1161. Curran Associates, Inc., 2010.

[Krystalakos *et al.*, 2018] Odysseas Krystalakos, Christoforos Nalmpantis, and Dimitris Vrakas. Sliding window approach for online energy disaggregation using artificial neural networks. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, 2018.

[Pérez-Lombard *et al.*, 2008] Luis Pérez-Lombard, José Ortiz, and Christine Pout. A review on buildings energy consumption information. *Energy and buildings*, 40(3):394–398, 2008.

[Uttama Nambi *et al.*, 2015] Akshay S.N. Uttama Nambi, Antonio Reyes Lua, and Venkatesha R. Prasad. Loced: Location-aware energy disaggregation framework. In *Proceedings of the 2Nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, BuildSys '15, pages 45–54, New York, NY, USA, 2015. ACM.

[Zhang *et al.*, 2018] Chaoyun Zhang, Mingjun Zhong, Zongzuo Wang, Nigel Goddard, and Charles Sutton. Sequence-to-Point Learning With Neural Networks for Non-Intrusive Load Monitoring. In *Thirty-Second AAAI Conference on Artificial Intelligence*, April 2018.

[Zhong *et al.*, 2014] Mingjun Zhong, Nigel Goddard, and Charles Sutton. Signal Aggregate Constraints in Additive Factorial HMMs, with Application to Energy Disaggregation. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3590–3598. Curran Associates, Inc., 2014.