LightNILM: Lightweight neural network methods for non-intrusive load monitoring

Zhenyu Lu
luzhenyu@bit.edu.cn
Beijing Institute of Technology
Haidian, Beijing, China

Yurong Cheng
yrcheng@bit.edu.cn
Beijing Institute of Technology
Haidian, Beijing, China

Mingjun Zhong
mingjun.zhong@abdn.ac.uk
University of Aberdeen
Aberdeen, United Kingdom

Wenpeng Luan
wenpeng.luan@tju.edu.cn
Tianjin University
Tianjing, China

Yuan Ye
yuan-ye@bit.edu.cn
Beijing Institute of Technology
Haidian, Beijing, China

Guoren Wang
wanggrbit@126.com
Beijing Institute of Technology
Haidian, Beijing, China

ABSTRACT

The aim of non-intrusive load monitoring (NILM) is to infer the energy consumed by the appliances in a house given only the total power consumption. Recently, literature have shown that deep neural networks are the state-of-the-art approaches for tacking NILM. For example, both sequence-to-sequence (seq2seq) and sequence-to-point (seq2point) learning models are the popular frameworks with typical network architectures such as convolutional neural networks (CNNs). However, these deep neural network approaches are computationally expensive and require huge storage for the purpose of prediction, and consequently would not be capable of deploying on mobile/edge devices. This paper addresses these issues for seq2point learning models by employing specifically designed network architectures which can be processed by using TensorFlow Lite to deploy on mobile phones. We show that our models only require 0.5% number of the parameters used in original seq2point models, whilst achieve comparable accuracy. Our models are then successfully tested on mobile phones with reasonable accuracy performance.

CCS CONCEPTS

• **Computing methodologies** → *Machine learning*; **Neural networks**.

KEYWORDS

Energy disaggregation, lightweight NILM, deep learning, edge NILM, sequence-to-point learning

ACM Reference Format:
Zhenyu Lu, Yurong Cheng, Mingjun Zhong, Wenpeng Luan, Yuan Ye, and Guoren Wang. 2022. LightNILM: Lightweight neural network methods for non-intrusive load monitoring. In *6th International Workshop on Non-Intrusive Load Monitoring (NILM '22), November 9–10, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3563357.3566152

NILM '22, November 9–10, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9890-9/22/11…$15.00
https://doi.org/10.1145/3563357.3566152

1 INTRODUCTION

The purpose of NILM is to disaggregate the total energy consumption of a household into the individual appliances. It is very helpful for householders to save energy by providing them with how energy was used in their houses. For example, research has shown that NILM can help to reduce energy consumption in a household by 15% [5]. In addition, integrating NILM into smart meters could help energy providers to optimize grid operation models and customize specific tariffs for users based on their energy consumption habits [10].

Practically, there would be two approaches for the roll-out of a NILM system: Firstly, the NILM algorithms could be implemented by cloud computing which means the appliances are disaggregated on the cloud and then smart devices read the disaggregated signals from there; Secondly, the NILM algorithms are implemented by edge computing which means that NILM algorithms are directly embedded into edge devices including smart meters, smart phones, and embedded systems. The first approach would tolerate substantial computational costs required by NILM algorithms, but need significant resources for cloud computing and as well as high quality network connections. The scalability of the approach using cloud computing would be limited as the technique may be deployed in every house which thus require immense communications. Perhaps, data privacy must be seriously considered to ensure cloud computing not to put the privacy of personal data at risk when collecting, storing, transfering and sharing data. By contrast, the second approach may not have these limitations, and thus is fast, secure, and more reliable for real-world applications. For example, scalability would not be an issue as the edge computing device is standalone and thus can be independently distributed into every house; data privacy would not be an issue any more as the smart/edge device could be encrypted.

In this paper, we investigate three lightweight architectures on the seq2point learning frameworks [29] for deployment on mobile devices and so potentially other edge devices in general. In the following we call our approaches as LightNILM. These approaches can be compressed by using Tensorflow Lite and then deployed on mobile phone. Our approaches are most relevant to [19] which combines MobileNet [9] and TensorFlow Lite. [30] proposed a method similar to [19], which uses pooling layer and dropout to achieve available accuracy with fewer parameters. These approaches are deemed as preliminary results in our view as no comparisons with variants were provided in their paper. What is more, based on the

previous two methods, [3] proposed a NILM method for embedded systems using model compression methods and multi-task learning. The algorithm is not explicitly described in their papers. Indeed, our paper compares three approaches for LightNILM, which are sliced recurrent neural networks (SRNN), depthwise separable convolutional neural networks (DepthWCNN), and residual neural networks (ResNet). We test our LightNILM approaches by applying them to three different data sets which are UK-DALE, REFIT, and REDD. They were tested on mobile phone devices. This provides a thorough analysis over LightNILM showing that ResNet would be the most promising approach for mobile devices and potentially edge devices in general. In the rest of our paper, we briefly review the related work of NILM in Section 2. Then in Section 3, we introduce our three LightNILM approaches in detail. In Section 4, we introduce the data sets used in the experiments. In Section 5, we report the results and corresponding analysis of our LightNILM methods tested on three datasets. Finally, we conclude our paper in Section 6.

2 NON-INTRUSIVE LOAD MONITORING

This section briefly reviews relevant methods for NILM. Given a time series of the measured power readings of the mains electricity in a household represented by $Y = \{y_1, y_2, \ldots\ldots, y_T\}$ where T denotes the time, the aim of NILM is to infer the energy consumption of all the individual appliances by only using Y. NILM can be viewed as a single channel blind source separation problem as the total power reading is the sum of the individual appliances: $Y_t = \sum_{i=1}^{I} x_{it} + \epsilon$, where x_{it} is the power reading of the i^{th} appliance and ϵ is usually a Gaussian noise. The aim is then to infer x_{it}.

Solutions to NILM include statistical methods, e.g., hidden Markov models (HMM) [2, 16, 33, 34], signal processing algorithms, e.g., graph signal processing (GSP) [7, 32], and deep neural networks [4, 6, 12, 20, 28, 29]. Among them, HMM and GSP algorithms would be scalable for edge computing; however, literature suggests that these approaches would not have better performance, compared to deep neural networks. The seq2seq and seq2point learning models [12, 29] are popular frameworks among those deep learning approaches. There have been various deep neural network architectures for implementing these frameworks in literature [11, 12, 20, 24, 26, 28, 29, 35]. However, these models could only be implemented with cloud computing due to the requirements of high computational costs and huge memory storage. There have been some work to extend seq2point learning models to edge computing. For example, the works of [1, 15, 25, 31] targeted to compress the seq2point model by pruning the model parameters; the work of [19] employs MobileNet [9] and Tensorflow Lite to compress the seq2point model. These approaches are promising for deployment on edge devices; for example, [19] has already tested the model on mobile phones. Some other lightweight NILM algorithms use compression methods and as well as traditional machine learning algorithms [18, 21–23].

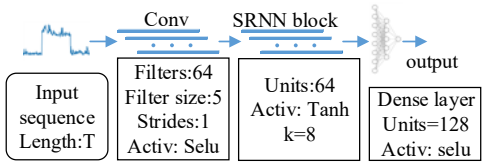
Our aim in this paper is to adapt the seq2point learning model by using sliced recurrent neural network [27], depthwise convolutional neural networks [9], and residual neural network [8], which are then compressed by Tensorflow Lite so that they can be deployed on mobile devices.

3 LIGHTWEIGHT NILM MODELS

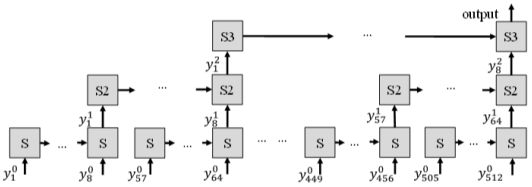
In this section, we describe the proposed lightweight NILM models which are adapted from the seq2point framework by using the following architecture strategies for the purpose of deploying on mobile devices.

3.1 Sliced recurrent neural networks

The first approach is to replace the CNN layers in seq2point except the first CNN layer by a SRNN block [27] with other layers being unchanged, leading to the SRNN architecture (Figure 1). The advantages are that SRNN has fewer parameters comparing with CNN, and its parallelism accelerates training. Notably, the size of the dense layer is also largely reduced. Given an input sequence $Y = \{y_1^0, y_2^0, \ldots\ldots, y_T^0\}$, we set a same recurrent unit S for each reading $y_i \in Y$. Let k readings form a subsequence, e.g., $k = 8$ in Figure 1(b). So the output of the first layer has $\lceil T/k \rceil$ units denoted by $\{y_1^1, y_2^1, \ldots, y_{\lceil T/k \rceil}^1\}$ which in turn is as the input of the second layer. This process repeats until the output has a single unit.



(a) The seq2point model with a SRNN block



(b) The SRNN block

Figure 1: The sliced recurrent neural network for NILM

3.2 Depthwise separable convolutional neural networks

We can also apply depthwise convolutional neural networks (DepthWCNN) method [9] to break the large convolution kernels into the sum of several small convolution kernels, for the convolutional layers and dense layer. This method can reduce the number of parameters and the computational complexity of the convolutional layers and dense layer.

The input $Y = \{y_1 \ldots y_T\}$ is treated as a $T \times 1$ sequence. Denote the feature map as F, the input size of F as $T \times M$, and the output size of F as $T \times N$, where M (resp. N) is the number of input (resp. output) channels. Let the convolution kernel be K with size D_K. If we apply standard convolution, the output of each layer G is computed as $G = \sum_{i,m} K_{i,m,n} F_{k+i-1,m}$ where $i \in [1, D_K]$, $m \in [1, M]$, $n \in [1, N]$. The number of parameters of K is $D_K \times M \times N$. So the computational complexity is $D_K \cdot M \cdot N \cdot T$.

The computational complexity is too large, so we apply the depthwise separable convolutions which is composed of a channel-wised convolution and a 1×1 standard convolution, which is called pointwise convolution. Let the convolution kernel be $\hat{K}$, the output of each layer $\hat{G}$ assuming padding would be computed as $\hat{G} = \sum_i \hat{K}_{i,m} F_{k+i-1,m}$. Then the computational complexity of depthwise separable convolution is $D_K \cdot M \cdot T + M \cdot N \cdot T$, which is the sum of the depthwise and pointwise convolution. We get a reduction in computational complexity of $\frac{D_K \cdot M \cdot N \cdot T}{D_K \cdot M \cdot T + M \cdot N \cdot T} = \frac{1}{N} + \frac{1}{D_K}$. What is more, the fully connected layer accounts for more than 99% of the parameters in the seq2point model which has more than 30 million parameters.

The fully connected layer could also be regard as a larger convolution layer. The input and the parameters of the fully connected layer are $a_{11}, \ldots, a_{1s}, a_{21}, \ldots, a_{2s}, \ldots, a_{m1}, \ldots, a_{ms}$ and $w_{11}, \ldots, w_{1s}, w_{21}, \ldots, w_{2s}, \ldots, w_{m1}, \ldots, w_{ms}$ respectively. So the fully connected layer can be expressed as matrices $A = (a_{ij})$ and $W = (w_{ij})$ $(i = 1, \cdots, m; j = 1, \ldots, s)$.

If the kernel size of convolution equals to the sequence length, then the convolution progress could be express as: $U = \sum_i \sum_j a_{ij} w_{ij} + b$, where U denotes one unit of fully connected layer and b denotes the bias of unit. Thus, similar to the above process we can also apply depthwise separable convolution to compress the fully connected layer. This model is shown in Figure 2.

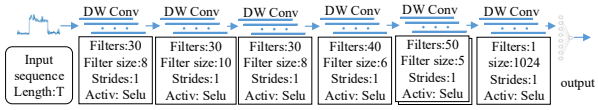


Figure 2: The DepthWCNN model

3.3 Residual neural network

The final approach is to employ the residual neural network (ResNet) [8]. For each convolutional layer in the DepthWCNN based method, ResNet can help to reuse the results of features in the last layer, which can improve the computational accuracy.

The framework of ResNet is shown in Figure 3. The idea is simply to use Res-block to replace depthwise separable convolution layer. Res-block has two structures, where the first is shown in Figure 3(a) and the second is shown in Figure 3(b). The block1 is similar to dense-block except that res-block add the input feature map to the output of convolutional layer instead of concatenation. Thus the building block1 is defined as: $G = H(F, W) + F$, where F and G are the input and output feature maps of the block, and W are the parameters of the convolutional layer. The function $H(F, W)$ denotes the convolutional layer. The process requires much less computational complexity than convolution without extra parameters. But as is shown in Figure 3(a), it could not add the input feature maps to the output of convolutional layer directly because the input and output feature maps have different dimensions. So we use a pointwise convolution P to increase the dimension of input to be the same as the output dimension of convolutional layer. Thus the building block2 is defined as: $G = H(F, W) + P(F, \hat{W})$. The framework of ResNet is shown in Figure 3(c). When the input and output are of the same dimensions, block1 is used, otherwise block2 is used. A large separable convolution layer is used at the end of the model instead of a fully connected layer for the output.

4 DATA SETS

Three data sets UK-DALE [13], REFIT [17] and REDD [14] are used to evaluate the performance of the proposed models for NILM.

UK-DALE: The active power acquisition time of the UK-DALE data set is 1s-6s. The house 1 is used for training and validation, and house 2 used for testing. Five appliances are used which are kettle, microwave, fridge, dishwasher, and washing machine.

REFIT: The active power sampling time of all appliances in the REFIT data set is 8s. The houses assigned for training, validation and test are shown in the Table 1.

REDD: The data acquisition time is 1s-3s. The houses 2&3 are used for training, house 3 used for validation and house 1 for testing. Four appliances were used: microwave, fridge, dish washer and washing machine.

Firstly, some data may not be appropriate for our analysis, e.g., houses 13 and 21 in REFIT data were using solar panel energy production; they were not used for our experiments. Secondly, all the time series were re-sampled to 8 seconds. Finally, all the data were standardized using $\frac{x_t - \bar{x}}{\sigma}$ where x_t is the power reading at time t, $\bar{x}$ is an mean power, and σ is a standard deviation of power reading. It could accelerate the convergence of model weight parameters. The means and standard deviations are shown in Table 1.

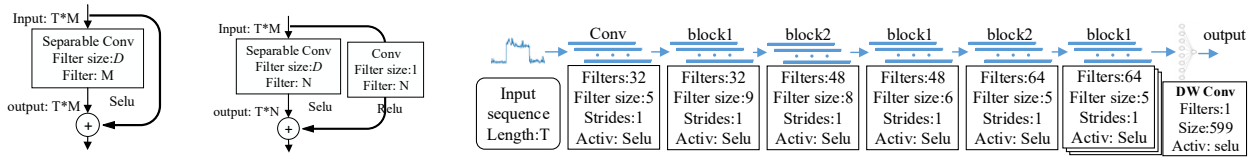
Table 1: Splitting of REFIT data and the mean, standard deviation values for preprocessing on all datasets.

Splitting of REFIT data used for the models				Deviation values	
Appliances	**training**	**validation**	**test**	**Mean**	**STD**
Mains	-	-	-	522	814
kettle	3,4,7,8,9,12,13,19,20	5	2	700	1000
microwave	10,12,19	17	4	500	800
fridge	2,5,9	12	15	200	400
dish washer	5,7,9,13,16	18	20	700	1000
washing machine	2,5,7,9,15,16,17	18	8	400	700

5 EXPERIMENTAL RESULTS

Metrics and Settings: The proposed models are applied to the three data sets. For evaluating the performance, mean absolute error (MAE) and normalized signal aggregate error (SAE) are used. MAE is defined as $\frac{1}{T} \sum_{t=1}^{T} |\hat{x}_t - x_t|$ where x_t and $\hat{x}_t$ are the true power reading at t and the predicted value, respectively. SAE is defined as $\frac{|\hat{r} - r|}{r}$ where r and $\hat{r}$ are the total energy and the predicted one, respectively. For training the deep learning models, the sliced mains windows with length T are used as inputs. We set $T = 512$ for RNN models, and $T = 599$ for CNN models. All the models are built on TensorFlow framework. We found that 25 epochs were enough for training the model for convergence with patience number 4 for early-stopping. We choose batch size 200 for all the models. The performance results are shown in Tables 2 and 3.

Accuracy: The results shown in Tables 2 and 3 indicate that in general our LightNILM models are comparable to the original seq2point in terms of MAE, but interestingly they outperform



(a) The ResNet block 1 (b) The ResNet block 2 (c) The residual neural network

Figure 3: The ResNet architecture

Table 2: The results of the seq2point in [29] and the proposed LightNILM models in this paper applied to UK-DALE, REFIT, REDD datasets running on a desktop computer and a mobile phone. Best results are denoted in bold across methods.

	UK-DALE								REFIT								REDD							
	Seq2point		SRNN		DepthWise		ResNet		Seq2point		SRNN		DepthWise		ResNet		Seq2point		SRNN		DepthWise		ResNet	
Metrics	MAE	SAE	MAE	SAE	MAE	SAE	MAE	SAE	MAE	SAE	MAE	SAE	MAE	SAE	MAE	SAE	MAE	SAE	MAE	SAE	MAE	SAE	MAE	SAE
Kettle	**7.4**	**0.069**	10.9	0.335	9.5	0.225	13.8	0.205	**6.8**	0.130	22.7	0.159	21.5	0.243	22.8	**0.060**	-	-	-	-	-	-	-	-
Microwave	8.6	0.486	9.4	0.431	**8.1**	0.389	9.5	**0.181**	**12.7**	0.170	14.4	0.154	14.8	**0.041**	16.3	0.107	28.2	**0.059**	23.9	0.176	28.5	0.072	**23.3**	0.112
Fridge	20.8	**0.121**	18.8	0.199	**18.4**	0.225	18.8	0.197	**20.0**	0.330	24.1	0.238	20.6	0.417	23.9	**0.162**	**28.1**	0.180	31.4	0.110	34.9	**0.069**	32.1	0.156
Dish w.	27.7	0.645	22.7	0.235	26.3	0.422	**19.7**	**0.181**	12.3	0.260	**11.7**	0.315	17.0	0.263	17.3	**0.162**	20.0	0.567	22.3	0.577	**16.0**	**0.374**	20.0	0.394
Washing m.	12.6	0.280	**10.4**	0.405	13.4	0.175	12.3	**0.098**	**16.9**	2.610	51.7	**0.167**	49.1	0.331	46.7	0.221	18.3	0.277	**11.1**	0.129	14.7	0.081	11.3	**0.057**
Mean	15.4	0.321	**14.3**	0.344	15.1	0.288	14.8	**0.172**	**13.7**	0.702	24.9	0.207	24.6	0.259	25.4	**0.142**	23.7	0.270	22.2	0.248	23.5	**0.149**	**21.7**	0.179

Table 3: The proposed LightNILM models in this paper applied to UK-DALE, REFIT, REDD datasets running on a mobile phone. Best results are denoted in bold across methods.

	UK-DALE				REFIT				REDD			
	DepthWise		ResNet		DepthWise		ResNet		DepthWise		ResNet	
Metrics	MAE	SAE	MAE	SAE	MAE	SAE	MAE	SAE	MAE	SAE	MAE	SAE
Kettle	33.7	0.051	**31.0**	**0.027**	**41.5**	0.020	43.0	**0.008**	-	-	-	-
Microwave	20.8	0.151	**9.5**	**0.110**	26.3	**0.028**	**16.7**	0.138	**27.3**	**0.020**	34.9	0.043
Fridge	34.5	**0.046**	**31.0**	0.104	37.4	0.113	**34.0**	**0.066**	84.6	0.093	**67.8**	**0.009**
Dish w.	**54.8**	0.121	55.7	**0.066**	**19.8**	**0.063**	20.1	0.074	**20.7**	0.136	28.1	**0.007**
Washing m.	**13.8**	0.148	16.7	**0.080**	38.7	**0.021**	**36.7**	0.065	39.0	0.045	**34.4**	**0.006**
Mean	31.5	0.103	**28.8**	**0.077**	32.8	**0.049**	**30.1**	0.070	42.9	0.074	**41.3**	**0.016**

seq2point largely in terms of SAE across all the data and methods. Most importantly, these LightNILM models have significantly smaller sizes than seq2point (See Table 4). By contrast, each of the LightNILM models only needs no more than 0.5% number of parameters and 0.2% memory of the original seq2point model. LightNILM algorithms outperform seq2point could be due to the huge number of parameters in seq2point which may have caused overfitting. The SRNN model has the lowest average value of MAE and the ResNet model has the lowest average value of SAE on UK-DALE data set. The SRNN model performs well especially on washing machine, which may be because the recurrent neural network is better at dealing with the regular state changes. Besides, seq2point method significantly outperform other methods on REFIT data. Among the lightNILM methods, the ResNet has the lowest average value of MAE on REDD data set. This is mainly because both densely connected convolutional layer and residual architecture can reuse the results of the features of all layers. Moreover, they can also solve the problem of disappearing/exploding gradient to a certain extent and improve the performance of the model.

We also tested LightNILM models based on Android platform and Tensorflow-Lite(TFLite) framework on the three data sets. We did not test the SRNN model on mobile phone because TFLite framework did not support RNN when we were working on these experiments. Tables 3 show that LightNILM models perform worse on mobile phones than on desktop computers due to quantization and weight pruning techniques required by TFLite. The difference of SAE is small but large for MAE. Thus, although the instantaneous estimation is not accurate enough, if we apply our LightNILM model to estimate a longer time of electricity, the error of total consumption would be quite small. The ResNet has best performance in terms of MAE and SAE on both UK-DALE and REDD, whilst has best performance in MAE on REFIT. This would suggest us to use ResNet for deployment on mobile devices.

Table 4: Numbers of parameters, and model sizes, and running time of the MobileNILM models.

	Seq2point	SRNN	DepthWCNN	ResNet
No. Parameters (Million)	30.708	0.149	0.083	0.108
TensorFlow Model size (MB)	449	2.03	1.04	1.42
TFLite Model size (MB)	149	0.340	0.11	0.14
Run time (ms)	–	–	20.58	27.51

To summarize, ResNet would be the best model among LightNILM by comprehensively considering those factors of accuracy, computation speed and the model size.

6 CONCLUSIONS

Several lightweight NILM models based on neural networks are proposed. We compared these models with seq2point model on UK-DALE, REDD and REFIT data sets. We found that each of our models only needs no more than 0.5% numbers of parameters required by the original seq2point model, and remarkably our methods are comparable comparing to the original model.

It is noticed in preliminary studies that the learned parameters for seq2point model are redundant [1]. Our LightNILM methods using specially designed neural network architecture blocks can effectively reduce the number of parameters, whilst achieve comparable performance. Nevertheless, the accuracy of the disaggregation on mobile devices need to be further improved in future research.

ACKNOWLEDGEMENT

This work is supported in part by the National Natural Science Foundation of China (Key Program), NSFC-SGCC (U2066207).

REFERENCES

[1] Jack Barber, Heriberto Cuayáhuitl, Mingjun Zhong, and Wenpeng Luan. 2020. Lightweight Non-Intrusive Load Monitoring Employing Pruned Sequence-to-Point Learning. In *Proceedings of the 5th International Workshop on Non-Intrusive Load Monitoring*. 11–15.

[2] Roberto Bonfigli, Stefano Squartini, Marco Fagiani, and Francesco Piazza. 2015. Unsupervised algorithms for non-intrusive load monitoring: An up-to-date overview. In *2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC)*. IEEE, 1175–1180.

[3] Mazen Bouchur and Andreas Reinhardt. 2022. Efficient Neural Network Representations for Energy Data Analytics on Embedded Systems. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems* (Virtual Event) *(e-Energy '22)*. Association for Computing Machinery, New York, NY, USA, 81–92. https://doi.org/10.1145/3538637.3538842

[4] Fabrizio Ciancetta, Giovanni Bucci, Edoardo Fiorucci, Simone Mari, and Andrea Fioravanti. 2020. A new convolutional neural network-based system for NILM applications. *IEEE Transactions on Instrumentation and Measurement* 70 (2020), 1–12.

[5] Fischer Corinna. 2008. Feedback on household electricity consumption: a tool for saving energy. *Energy Efficiency* 1, 1 (2008), 79–104.

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

[7] Renhai Feng, Wanqi Yuan, Leijiao Ge, and Siyu Ji. 2021. Nonintrusive Load Disaggregation for Residential Users Based on Alternating Optimization and Downsampling. *IEEE Transactions on Instrumentation and Measurement* 70 (2021), 1–12.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[9] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. (2017).

[10] Chen-Yu Hsu, Abbas Zeitoun, Guang-He Lee andDina Katabi, and Tommi Jaakkola. 2020. Self-Supervised Learning of Appliance Usage. In *In ICLR*.

[11] Jie Jiang, Qiuqiang Kong, Mark Plumbley, and Nigel Gilbert. 2019. Deep Learning Based Energy Disaggregation and On/Off Detection of Household Appliances. *arXiv preprint arXiv:1908.00941* (2019).

[12] Jack Kelly and William Knottenbelt. 2015. Neural NILM: Deep neural networks applied to energy disaggregation. In *BuildSys*. ACM.

[13] Jack Kelly and William Knottenbelt. 2015. The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes. *Scientific Data* 2, 150007 (2015). https://doi.org/10.1038/sdata.2015.7

[14] J Zico Kolter and Matthew J Johnson. 2011. REDD: A public data set for energy disaggregation research. In *SIGKDD Workshop on Data Mining Applications in Sustainability*, Vol. 25. Citeseer.

[15] Rithwik Kukunuri, Anup Aglawe, Jainish Chauhan, Kratika Bhagtani, Rohan Patil, Sumit Walia, and Nipun Batra. 2020. *EdgeNILM: Towards NILM on Edge Devices*. Association for Computing Machinery, New York, NY, USA, 90–99. https://doi.org/10.1145/3408308.3427977

[16] Stephen Makonin, Fred Popowich, Ivan V. Baji, Bob Gill, and Lyn Bartram. 2016. Exploiting HMM Sparsity to Perform Online Real-Time Nonintrusive Load Monitoring. *Smart Grid, IEEE Transactions on* 7, 6 (2016), 2575–2585.

[17] David Murray, Lina Stankovic, and Vladimir Stankovic. 2017. An electrical load measurements dataset of United Kingdom households from a two-year longitudinal study. *Scientific data* 4, 1 (2017), 1–12.

[18] Minh H Phan, Queen Nguyen, Son L Phung, Wei Emma Zhang, Trung D Vo, and Quan Z Sheng. 2021. CompactNet: A Light-Weight Deep Learning Framework for Smart Intrusive Load Monitoring. *IEEE Sensors Journal* (2021).

[19] Ahmed Shamim and Bons Marc. 2020. *Edge Computed NILM: A Phone-Based Implementation Using MobileNet Compressed by Tensorflow Lite*. Association for Computing Machinery, New York, NY, USA, 44–48. https://doi.org/10.1145/3427771.3427852

[20] Changho Shin, Sunghwan Joo, Jaeryun Yim, Hyoseop Lee, Taesup Moon, and Wonjong Rhee. 2019. Subtask Gated Networks for Non-Intrusive Load Monitoring. *AAAI* (2019).

[21] Enrico Tabanelli, Davide Brunelli, Andrea Acquaviva, and Luca Benini. 2022. Trimming Feature Extraction and Inference for MCU-Based Edge NILM: A Systematic Approach. *IEEE Transactions on Industrial Informatics* 18, 2 (2022), 943–952. https://doi.org/10.1109/TII.2021.3078186

[22] Enrico Tabanelli, Davide Brunelli, and Luca Benini. 2020. A Feature Reduction Strategy For Enabling Lightweight Non-Intrusive Load Monitoring On Edge Devices. In *2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)*. 805–810. https://doi.org/10.1109/ISIE45063.2020.9152277

[23] Enrico Tabanelli, Giuseppe Tagliavini, and Luca Benini. 2021. DNN is not all you need: Parallelizing Non-Neural ML Algorithms on Ultra-Low-Power IoT Processors. *arXiv preprint arXiv:2107.09448* (2021).

[24] Nikolaos Virtsionis-Gkalinikis, Christoforos Nalmpantis, and Dimitris Vrakas. 2021. SAED: self-attentive energy disaggregation. *Machine Learning* (2021), 1–20.

[25] XiaoYu Wang, Hao Zhou, Nikolaos M. Freris, Wangqiu Zhou, Xing Guo, Zhi Liu, Yusheng Ji, and Xiang-Yang Li. 2021. LCL: Light Contactless Low-delay Load Monitoring via Compressive Attentional Multi-label Learning. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. 1–6. https://doi.org/10.1109/IWQOS52092.2021.9521262

[26] Weijun Yang, Chengxin Pang, Jinhai Huang, and Xinhua Zeng. 2021. Sequence-to-Point Learning Based on Temporal Convolutional Networks for Nonintrusive Load Monitoring. *IEEE Transactions on Instrumentation and Measurement* 70 (2021), 1–10.

[27] Zeping Yu and Gongshen Liu. 2018. Sliced recurrent neural networks. *arXiv preprint arXiv:1807.02291* (2018).

[28] Zhenrui Yue, Camilo Requena Witzig, Daniel Jorde, and Hans-Arno Jacobsen. 2020. BERT4NILM: A Bidirectional Transformer Model for Non-Intrusive Load Monitoring. In *the 5th International Workshop on Non-Intrusive Load Monitoring*. 89–93.

[29] Chaoyun Zhang, Mingjun Zhong, Zongzuo Wang, Nigel Goddard, and Charles Sutton. 2018. Sequence-to-point learning with neural networks for nonintrusive load monitoring. In *AAAI*.

[30] Ruiqi Zhang, Wenpeng Luan, Bo Liu, and Mingjun Zhong. 2021. A Lightweight Neural Network for Energy Disaggregation Employing Depthwise Separable Convolution. In *2021 IEEE Sustainable Power and Energy Conference (iSPEC)*. 4109–4114. https://doi.org/10.1109/iSPEC53008.2021.9735713

[31] Yu Zhang, Guoming Tang, Qianyi Huang, Yi Wang, Xudong Wang, and Jiadong Lou. 2021. FedNILM: Applying Federated Learning to NILM Applications at the Edge. *arXiv preprint arXiv:2106.07751* (2021).

[32] Bochao Zhao, Kanghang He, Lina Stankovic, and Vladimir Stankovic. 2018. Improving event-based non-intrusive load monitoring using graph signal processing. *IEEE Access* 6 (2018), 53944–53959.

[33] Mingjun Zhong, Nigel Goddard, and Charles Sutton. 2014. Signal Aggregate Constraints in Additive Factorial HMMs, with Application to Energy Disaggregation. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (Eds.), Vol. 27. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2014/file/2d1b2a5ff364606ff041650887723470-Paper.pdf

[34] Mingjun Zhong, Nigel Goddard, and Charles Sutton. 2015. Latent Bayesian melding for integrating individual and population models. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*. 3618–3626.

[35] Gan Zhou, Zhi Li, Meng Fu, Yanjun Feng, Xingyao Wang, and Chengwei Huang. 2020. Sequence-to-sequence load disaggregation using multiscale residual neural network. *IEEE Transactions on Instrumentation and Measurement* 70 (2020), 1–10.