Efficient Large-Scale Machine Learning Techniques for Rapid Motif Discovery in Energy Data Streams

K. K. Lykothanasi, S. Sioutas, K. Tsichlas

Efficient Large-Scale Machine Learning Techniques for Rapid Motif Discovery in Energy Data Streams

K.K. Lykothanasi[1, 2], S. Sioutas[1][0000-0003-1825-5565] and K. Tsichlas[1][0000-0003-4107-9520]

[1] Computer Engineering & Informatics Department, University of Patras, Patras, Greece
[2] Department of Computer Science, KU Leuven, Leuven, Belgium
sioutas@ceid.upatras.gr

Abstract. Domestic appliance power consumption measurement was, until recently, a problem without a satisfying solution. It required the use of a measuring device for each appliance to be studied, and thus the spending of a considerable amount of both money and time. The technological advancements made in the past few decades have enabled the engineering of smart devices that connect to the central panel of a building and log the features of the electrical current passing through it. Using Machine Learning algorithms, we can create models that extract individual appliance information ("signature") from the signals recorded by these measuring devices. This process can lead to the production of systems that could be particularly useful for the consumers. They would not only allow individuals to alter their power consumption profile to minimise their spending and environmental impact, but also notify them if an appliance seems to be malfunctioning. In addition, the energy providers could harness the potential of usage statistics collected from their customers to estimate the energy demand for any given moment within a day. This would prevent the production of excess energy or the overloading of the power supply network infrastructure. The objective of this work is the implementation of an efficient Deep Neural Network (DNN) model that will be able to predict the state (On/Off) of a set of electrical appliances during a specific time span, based on the aggregate power signal of the house within which they operate. The contribution of this work concerns the use of a Recurrent Neural Network (RNN) that categorises the behaviour of multiple appliances (multi-label classification). The results are quite promising and pave the way for a more in-depth treatment of the problem.

Keywords: Machine Learning, Energy Consumption, Non-intrusive Load Monitoring, Energy Disaggregation, Recurrent Neural Networks.

1 Introduction

Until recently, it was not feasible for consumers to know their precise home power consumption and more specifically to know exactly the amount of energy used by each appliance. However, the technological advancements of the last few decades enable consumers to discover their power consumption profile with the use of smart

meters. With this knowledge, the consumer can adjust their behaviour and their power consumption, in order to profit financially and improve their quality of life.

The term Non-Intrusive Load Monitoring – NILM means "a process that analyses the variations in the features of the electricity of a house and concludes which of the appliances are operating as well as the energy consumption of each one" [1]. Alternatively, it is referred to as energy disaggregation. Formally, the problem is stated as:

Let $\boldsymbol{T}$ = {1, 2, ..., T} *be a set of time points and* $\boldsymbol{X} = \{X_1, X_2, \ldots, X_T\}$ *be the set of measurements of the total energy consumption of* $\boldsymbol{N}$ *appliances (aggregate signal) that correspond to these points. Then, the objective of NILM is for a given point* $\boldsymbol{t} \in \boldsymbol{T}$ *to disaggregate the consumption* $\boldsymbol{y_t^i}$ *of appliance* $\boldsymbol{i}$*, so that at every time point* $\in \boldsymbol{T}$ *it holds that:*

$$X_t = \sum_{i=1}^{N} y_t^i + \sigma(t) \quad (1)$$

where $\boldsymbol{\sigma(t)}$ *represents the consumption of unknown devices or noise.*

The applications of NILM are numerous and significant. Firstly, the average consumer has access to a detailed analysis of their energy consumption, according to which they can adapt their energy consumption habits, with multiple environmental and financial benefits. Additionally, they are warned when an appliance operates anomalously (e.g. because of a malfunction or of excessive use due to negligence). Secondly, the energy provider can predict the energy requirements of their network more accurately using the more detailed consumption data. Thus, they can avoid excess energy production or network overload. In addition, the energy provider can propose flexible programs to the consumer, with cheaper charge per kWh depending on current energy demand, to balance the network load.

The aggregate energy data is collected from a smart meter that is connected to the central switchboard of a house. Various methods have been used for energy disaggregation, mainly [30] Hidden Markov Models, Optimization Methods, Template Matching Methods (e.g., Dynamic Time Warping), Source Separation Methods (e.g., Matrix Factorisation), Shallow Learning (e.g. Random Forests), Deep Learning (e.g. Convolutional Neural Networks), Graph Signal Processing. This work is focused on deep learning methods. These methods dominate the most recent NILM literature due to their superior performance in identifying appliances using the aggregate signal. The dominant approach in shallow and deep learning is the construction of different models, one for each appliance. This is preferred because each appliance has different consumption characteristics, and a single model can fit to them better than one model can fit to all appliances. However, multi-label classification has been recently [30][31] introduced to NILM research mainly on the grounds of better efficiency and deployment simplicity, even though it lacks effectiveness in terms of appliance identification when compared to the single-label classification approach. To the best of the authors' knowledge, only Convolutional Neural Networks (CNNs) have been used for multi-label classification [32].

In this work, the objective is the design and training of an RNN that takes as input the aggregate signal, and outputs the operational state (On/Off) of a group of appliances. The contribution of this extended abstract is the use of an RNN for multi-label energy disaggregation by a single neural network that has high precision and very low classification error in new (unknown) data, that may come from other sources. For the authors, these results are the first step towards an in-depth treatment of energy disaggregation where efficiency is considered of equal importance to effectiveness. This is because, as highlighted in [30], the computational cost of high-performance methods with respect to effectiveness is prohibitive for large-scale deployment.

The paper is structured as follows: Section 2 briefly presents the state-of-the-art, while Section 3 describes the system design and data pre-processing. Section 4 presents the implementation and the experimental results. Section 5 contains the conclusion, with extensions and future work.

2 Related Work

The concept of NILM was first introduced by G. W. Hart in [2], where he defined the problem, and presented an algorithm based on clustering of the electric measurements that provided encouraging results. Since then, many approaches have been developed. An extensive review of these techniques can be found in [3] [4][30], while a more detailed analysis is presented in [5]. The rest of this section is only a shallow review of the research done on NILM.

Hidden Markov Models (HMM) were initially used for the NILM problem based on the assumption that the hidden states correspond to the different analysed appliances in the house, and the observations to the features of the aggregated signal that is measured. The authors of [6], developed an HMM where the hidden states are the points of the set that is produced by the Cartesian product of the operating state labels of all the appliances of interest in the house. This state coding involves calculations with sparse matrices. Thus, the derived algorithm is effective for data with low sampling rate. Despite this, the algorithm is not sufficient for modelling all the appliances of a typical house, since the complexity increases exponentially over the number of appliances. In [7], a "Difference Factorial HMM (FHMM)" is described. This model does not need data labels for training, it is computationally effective and not affected by local extrema. Based on FHMMs, the authors of [8] propose an estimation algorithm that achieves good generalisation for different but similar appliances. The disadvantage of this approach is that during the training on a specific appliance, it is assumed that no other appliance changes state.

Recurrent Neural Networks (RNN) [9] are a type of neural network (NN) that do not only use the current input, but also the previous output to produce the new output. They use state vectors that operate as internal "memory units" of the neurons. Thus, RNNs have been utilised in a variety of applications where data sequence processing is needed [10]. A special type of RNNs, the Long Short-Term Memory (LSTM) networks, have a more complex internal structure, that enables them to correlate points of the sequence with larger relative distance between them compared to a convention-

al RNN. This characteristic resolves the problem of vanishing gradient. Thus, LSTM networks are more effective for time series processing as is the case of NILM. Such applications are presented in [11] and [12], where the networks are provided with the aggregated signal as an input and disaggregate it to a signal of a specific appliance in the output, obtaining very promising results for specific types of appliances. Recently, a new type of RNN was introduced as another solution to the vanishing gradient problem: the Gated Recurrent Unit (GRU) [13]. GRU networks require less calculations, resulting in less training time. Simulation results show that the effectiveness of GRUs is comparable to that of LSTMs [14].

The Convolutional Neural Network (CNN) [15] is a type of Multi-Layer Perceptron (MLP), that performs convolution of the inputs in at least one layer. CNNs have been used for NILM in order to extract the signal of one appliance from the aggregated signal [16].

Another type of NN that has been used for NILM is the Denoising Autoencoder (DAE) [17]. A DAE assumes that the input contains noise, and the output is an attempt to reconstruct the input without this noise (unsupervised learning). A disadvantage of this method is that every network can isolate the signal of only one appliance, thus requiring multiple DAEs for a group of appliances.

Finally, the combination of NNs and Markov Models has been explored. In [18], a CNN is used to extract the features of one appliance, and it is combined with a Hidden Semi-Markov Model (HSMM) that models this appliance.

The evaluation of these techniques is based on various publicly available datasets. These datasets vary widely with respect to sampling rate, number of features and types of appliances. One of the most used public datasets is the Reference Energy Disaggregation Data Set (REDD) [19]. It contains measurements from 6 houses, in a time span that ranges from a few days to a few months, with a sampling rate 15 KHz for the aggregated signal, 0.5 Hz for circuits with a single appliance and 1 Hz for circuits with more than one appliance (each house has 10-24 different appliances). The Almanac of Minutely Power dataset (AMPds) [20] contains samples with a time distance of 1 minute (0.0167 Hz), that were being collected for one year from a house with 19 appliances. There is also a second edition, AMPds2, with the data of one more year of observations. UK Domestic Appliance-Level Electricity (UK-DALE) [21] is a dataset that contains measurements of 5 houses in the UK, for a period of 2.5 years. The sampling rates for the aggregate signal and for each appliance are 16 KHz and 0.167 Hz respectively. Finally, ENERTALK [22] is a data set collected from 22 houses in Korea. The aggregate signal, as well as the consumption of the up-to-7 measured appliances, are sampled at 15Hz for time periods that range between 29 and 122 days.

3 System Design and Data Pre-processing

Due to the sequential nature of the aggregate signal in NILM, an RNN seems to be the most promising approach. Thus, a Deep RNN was used in this study, with two different types of neurons, LSTM and GRU. Preliminary tests with both types of neurons

showed that their performance was comparable. Therefore, the GRU type was selected since it is the most computationally efficient. The number of network inputs was equal to the number of features of each dataset. The number of hidden layers was determined by trial and error. In the architectures tested in the majority of the design experiments, the number of hidden neurons decreased from the input to the output layer, while the activation function for all the hidden neurons was the hyperbolic tangent (tanh).

The network outputs were binary class labels, with each label corresponding to exactly one appliance. Consequently, the output layer was a dense layer comprised of a number of neurons equal to the number of appliances used in each dataset. The activation function of the output layer was the sigmoid, thus the output values are in the range [0, 1]. Binary cross-entropy was selected as the loss function, since every sample can either belong or not to each of the more-than-one classes (multi-label classification task) and the objective was to optimize the performance for every class separately. Finally, several optimisers were tested: Stochastic Gradient Descent (SGD), AdaGrad, Adam and RMSprop. The best results were obtained using Adam [24] with learning rate equal to 0.001.

3.1 Datasets

ENERTALK. ENERTALK was the publicly available dataset chosen for the experiments. The second dataset was collected from the DinRail Cerberus Smart Meter of Meazon S.A. [25]. Among the 22 houses that are included in ENERTALK, only houses that has measurements of 3 to 5 appliances were considered (houses 01, 02, 04, 05, 06, 08, 17, 18 and 21). Ultimately, house 02 was selected because of the types of contained appliances: a fridge, a TV, a washing machine, and a rice cooker. The measurements were logged for 31 days and represent the everyday use of all appliances. The recorded features both the aggregated signal and each device were the following:

- **Timestamp:** The time instant of the measurement at Unix milliseconds timestamp format.
- **Active Power:** The real value of the power that is consumed from an AC circuit. It is measured in Watts (W) and given by the formula: $P = V_{RMS} \times I_{RMS} \times \cos\varphi$, where V_{RMS} is the active voltage, I_{RMS} is the active current intensity and φ is the angle between the voltage and current phasors.
- **Reactive Power:** The energy that flows towards the load and in reverse along a wire in an AC circuit. It is measured in Volt-Ampere Reactive (var) and given by the formula: $\mathrm{Q} = V_{RMS} \times I_{RMS} \times \sin\varphi$.

MEAZON. The DinRail Cerberus device is installed in the central panel of a house and records the aggregate signal. The sampling rate of the smart meter is 50 Hz, which makes it a precise tool for data collection for NILM. It has the following operation states: power analysis and harmonic analysis for three harmonic frequencies or for all available harmonics. The dataset contains measurements of a house, with 5

sampled devices: AC, oven, press, stove, and vacuum cleaner. There are gaps between the recorded operation of each device. The smart meter records 13 features for each sample out of which 7 were selected for the authors' experiments, in order to compare the results with those of [25]. These features were timestamp, active power, and reactive power as defined above, in addition to:

- **RMS Current:** The active current intensity in Amperes (A), given by: $I_{RMS} = \frac{I_{peak}}{\sqrt{2}}$.
- **Phase Shift:** The angle φ between the voltage and current phasors (phase difference).
- **Apparent Power:** The power that is theoretically consumed by an AC circuit. It is measured in Volt-Amperes (VA), and it is given by the formula: $S = V_{RMS} \times I_{RMS}$.
- **Crest Factor:** It is calculated using the formula: $CF = \frac{I_{peak}}{I_{RMS}}$.

Regarding the ground truth, separate smart meters were not used for each appliance, since during the operation logging of each appliance, no other appliances of interest were in use. Operation labels (On/Off) have been added manually for each appliance. As a result, there are 5 editions of the dataset with the same data and different labels, since in [25], the objective was the construction of one separate classifier for each appliance.

3.2 Data Pre-processing

ENERTALK Dataset. The first 10 overlapping days of measurements for all selected houses were utilised. The dataset is provided in the Parquet format (distributed systems file format). The data for each day is stored in a separate directory, which contains one file with the measurements corresponding to each device. All data except for the timestamp was kept. The first 4 days were used as the training set, the next 4 days as the validation set and the last 2 days as the test set. The labels for every device were binary-encoded, with 0 corresponding the Off state and 1 to the On state. For every sample each appliance was individually assigned the On label if the sampled value of active power was greater than 15 Watts. This threshold is used in [21] to calculate the ratio of On to Off states.

According to [11], NN performance is better if the data is normalised in the range [0, 1]. The authors also tested other methods, such as the transformation to a Gaussian distribution with a mean of 0 and a variance of 1, but the performance was not improved. Thus, [0, 1] normalisation was applied in this work. It should be noted that the normalization step followed the division of the data to the three subsets. Otherwise, there would have been an information leak from the training to the test set and the model would have been biased [26].

DinRail Cerberus Dataset. The data pre-processing had already been performed in [25]. The labels had been divided into different files per appliance. These files were merged by the authors to create the final data set, which included measurements of all

appliances. Then, the data was split into training and test sets with a ratio of 85/15 and both sets were normalized separately to the range [0, 1], as was done with ENERTALK.

4 System Implementation and Training

The language of choice for the implementation of the RNN model was Python 3.7, with NumPy (1.19.5), pandas (1.2.4), PyArrow (4.0.0) and scikit-learn (0.22.2.post1) being the packages used for reading, pre-processing, and storing the data. Matplotlib (3.2.1) was used to produce the graphs. The models were created, trained and evaluated using TensorFlow (2.4.1) [27] as the backend and Keras as the high-level interface. The experiments were conducted on the Google Colaboratory platform.

The hidden layers of the RNN were implemented using GRU cells (tensorflow.keras.layers.GRU) with tanh as the activation function, whereas the output layer was a simple densely connected one (tensorflow.keras.layers.Dense) with sigmoid as the activation function. Adam (tensorflow.keras.optimizers.Adam) was chosen as the optimization algorithm for the training process. In the rest of this section, the authors present their experiments to discover the optimal structure of the NN (the number of hidden layers and the number of neurons in each layer, since the number of inputs and outputs are defined by the training set). The network output is represented by a (n×1) binary vector, where n is the number of the appliances, and each output corresponds to an appliance state. When the appliance is On, then the respective bit is 1, otherwise it is 0. When training the NN on ENERTALK, the additional techniques of validation and early stopping (tensorflow.keras.callbacks.EarlyStopping) were used. More specifically, at the end of every epoch the network was fed the validation samples and the classification error and accuracy values were recorded. If the validation error increased or the accuracy decreased from epoch to epoch then training stopped, in order to achieve better generalization. In the experiments, the training process stopped when the accuracy did not improve more than 0.001 after 20 epochs. This technique was not used for the Meazon dataset, since it is very small. Figure 1 depicts a flowchart that describes the model training and selection process.

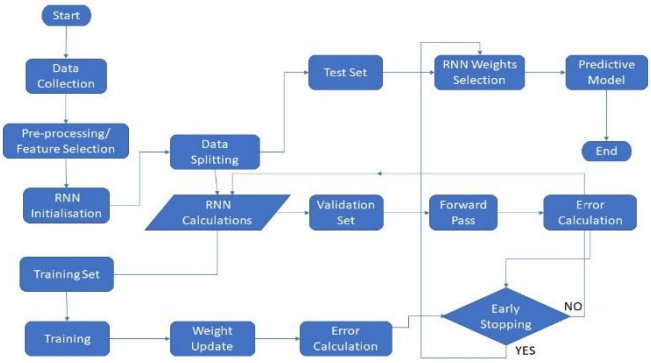

Fig. 1. Experimental process flow chart

As already mentioned, binary cross-entropy was the chosen loss function. Consequently, binary accuracy was chosen as the accuracy function. Additionally, the values of Mean Squared Error (MSE) and Mean Absolute Error (MAE) were calculated for each epoch.

Since the training of a NN is a stochastic process (due to the random initialisation), the results presented below are the mean values of each metric over 10 runs of every experiment. Experiments with more than 10 runs were performed, but there was no significant difference in the results. The presented graphs depict the values obtained from one experiment instance.

4.1 1st Model

The first experiments were focused on a structure with two hidden layers with a decreasing number of cells from the input to the output layer, since they performed better in preliminary tests. The architecture with the best experimental results consisted of 64 cells in the first and 32 cells in the second hidden layer respectively. Table 1 shows the mean values of the metrics in the testing phase for both datasets, while Figure 2 depicts the evolution of the metrics during training.

Table 1. Test results for 1st model

	Binary Cross-entropy (Loss)	**Binary Accuracy**
ENERTALK	0.5059	0.7755
Meazon	0.7338	0.7873

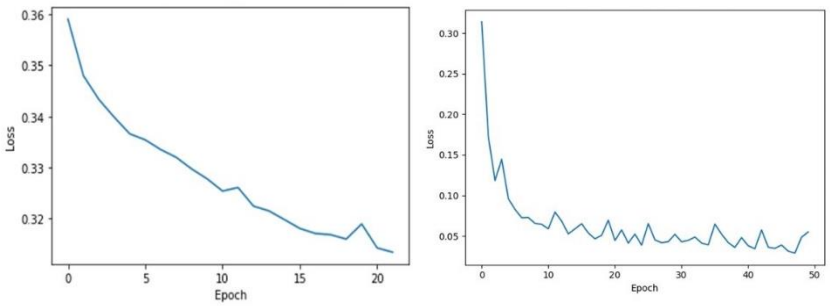

Fig. 2. Training loss for ENERTALK (left) and Meazon (right) datasets – 1st model

During the training of the network for the ENERTALK dataset, it was observed that, while the achieved accuracy was sufficient (over 80% after the 5th epoch) and the error values were relatively small, there was a significant decrease in validation accuracy after very few epochs (usually 5). Thus, the network was overfitting on the data of the training set. This fact was also confirmed by the prediction accuracy on the test set, which was smaller, although the opposite result was expected due to the suf-

ficient amount of the data used and the assumed complexity of this classification task. Overfitting was also observed in the Meazon dataset. In this case it was expected, since every sample corresponds to the operation of one or none of the appliances, and therefore the classification task was "easier".

4.2 2nd Model

The next group of experiments focused on improving generalization for both datasets. Overfitting was reduced via regularization, by applying the Dropout technique [28]. Dropout is a strategy based on the "dropping/deletion" of some randomly selected neurons during training, meaning that their output is ignored. In this way, the capacity of the network is reduced, and the rest of the neurons are forced to learn a more general classification strategy. To apply Dropout to a network layer, a new parameter P_d is introduced, which is the probability of a neuron being dropped. After several experiments with different values of P_d, the results indicated that the best value was $P_d = 0.4$ for every hidden layer of the network. Table 2 shows the mean values of the testing metrics, while Figure 2 depicts the evolution of the metrics during training.

Table 2. Test results for 2nd model

	Binary Cross-entropy (Loss)	**Binary Accuracy**
ENERTALK	0.5345	0.8091
Meazon	0.0198	0.9683

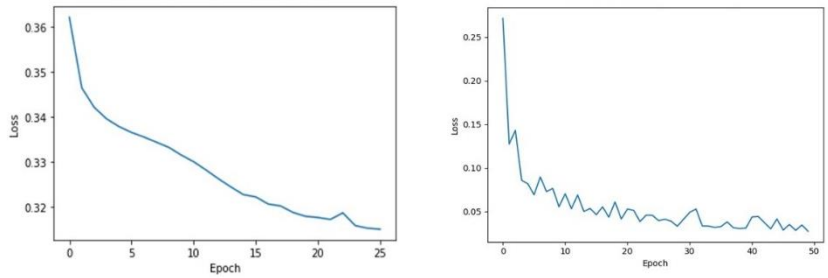

Fig. 3. Training loss for ENERTALK (left) and Meazon (right) datasets – 2nd model

For the ENERTALK dataset there was no significant difference between the two models. The classification accuracy was better by up to 3% but the error metrics slightly increased. These results showed that overfitting had been handled but not to the expected extent: the difference between training and testing accuracy had been reduced, but the improvement in every training epoch was not reflected in the validation.

In contrast, the training of the 2nd model with the Meazon dataset presented significant improvement with very good results overall. Of course, we must note that the

Meazon dataset is “clean”, since every device operates separately. Additionally, the test data is a small sample of one relatively small dataset. However, the results were very promising for future work on more data from DinRail Cerberus or other smart meters with sampling rates of around 50Hz.

It was determined that the 2nd model performed in an almost optimal manner for the Meazon dataset. As a result, the next set of experiments continued to explore models tailored to the ENERTALK dataset. However, in most of the new experiments both datasets were used for the sake of completeness. The first approach to decrease overfitting was the further decrease of the numbers of neurons of the network, considering that a network with lower elasticity (capacity) is more capable to generalize instead of adapting to the data. However, the results of these experiments were the opposite. In most of the runs, all scores were lower, and the test accuracy did not even come close to the training accuracy. Similar results were obtained when the dropout technique was used.

Subsequently, some of the more basic hyperparameters of the network were reconsidered. Firstly, the Rectified Linear Unit (ReLU) activation function was used, as it has proved to be very effective in DNN training while avoiding the vanishing gradient problem [29]. Finally, additional experiments with different values of the learning rate were performed, in order to confirm that the overfitting was not caused by the granularity of the weight updates. Unfortunately, all these attempts were not successful at eliminating, or even decreasing overfitting.

5 Conclusions and Future Work

In this work, a deep RNN was designed, implemented, trained, and evaluated that receives as input the instant aggregated signal from the central electricity switchboard of a house and predicts the operational state of a group of appliances. In other words, it decides whether each of the appliances of interest is functioning or not (On/Off) in each time point. The final model was extracted via extensive experimentation, using several architectures as well as many combinations of different values of the hyperparameters. The final model was tested on two datasets with high sampling rate: the publicly available ENERTALK (4 appliances) and the dataset in [25] with data recorded by the Meazon smart meter (5 appliances). The Meazon dataset is simpler since it contains 6 features and at every time step only one device was in operation. The ENERTALK dataset has two features and simultaneous operation of more than one device at every time step. The classification performance was adequate for both datasets but, as expected, the highest performance was achieved for the easier problem of the Meazon dataset. However, in the case of the ENERTALK dataset, the phenomenon of overfitting was observed, which meant that the classification capability of the network was lower than expected.

The proposed model can be improved in several ways. First, for the Meazon dataset, it would be very interesting to test the trained model with a new dataset containing measurements when 2 or more of the 5 appliances are operating concurrently.

Note that the sampling rate of the Meazon smart meter is 50 Hz, while it also generates a set of features that may make the classification task more efficient when compared to the ENERTALK dataset. The ENERTALK dataset is perhaps a more challenging problem due to the lower sampling rate and the small number of features. A way to face the problem is to use data from more than one house, and for testing to use data from houses that have not been used in training.

Acknowledgments This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE – INNOVATE (project code:T2EDK-00127)».

References

1. Nonintrusive load monitoring, https://en.wikipedia.org/w/index.php?title=Nonintrusive_load_monitoring&oldid=993516814, (2020).
2. Hart, G.W.: Nonintrusive appliance load monitoring. Proceedings of the IEEE. 80, 1870–1891 (1992). https://doi.org/10.1109/5.192069.
3. Verma, A., Anwar, A.: A Comprehensive Review on the NILM Algorithms for Energy Disaggregation. arXiv:2102.12578 [cs, eess]. (2021).
4. Faustine, A., Mvungi, N.H., Kaijage, S., Michael, K.: A Survey on Non-Intrusive Load Monitoring Methodies and Techniques for Energy Disaggregation Problem. arXiv:1703.00785 [cs]. (2017).
5. Liu, H.: Non-intrusive Load Monitoring: Theory, Technologies and Applications. Springer Singapore (2020). https://doi.org/10.1007/978-981-15-1860-7.
6. Makonin, S. et al.: Exploiting HMM Sparsity to Perform Online Real-Time Nonintrusive Load Monitoring (NILM). IEEE Transactions on Smart Grid. (in press), 1–11 (2015). https://doi.org/10.1109/TSG.2015.2494592.
7. Kolter, J.Z., Jaakkola, T.: Approximate Inference in Additive Factorial HMMs with Application to Energy Disaggregation. In: Artificial Intelligence and Statistics. pp. 1472–1482 PMLR (2012).
8. Parson, O. et al.: Non-intrusive load monitoring using prior models of general appliance types. Presented at the Twenty-Sixth Conference on Artificial Intelligence (AAAI-12) (26/07/12) July 24 (2012).
9. Haykin, S.S.: Neural Networks and Learning Machines. Prentice Hall (2009).
10. Recurrent neural network, https://en.wikipedia.org/w/index.php?title=Recurrent_neural_network&oldid=1027494214, (2021).
11. Kelly, J., Knottenbelt, W.: Neural NILM: Deep Neural Networks Applied to Energy Disaggregation. Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments. 55–64 (2015). https://doi.org/10.1145/2821650.2821672.
12. He, W., Chai, Y.: An Empirical Study on Energy Disaggregation via Deep Learning. Presented at the 2016 2nd International Conference on Artificial Intelligence and Industrial Engineering (AIIE 2016) November (2016). https://doi.org/10.2991/aiie-16.2016.77.
13. Cho, K. et al.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv:1406.1078 [cs, stat]. (2014).

14. Gated recurrent unit, https://en.wikipedia.org/w/index.php?title=Gated_recurrent_unit&oldid=997015931, (2020).
15. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol. Cybernetics. 36, 4, 193–202 (1980). https://doi.org/10.1007/BF00344251.
16. Shin, C. et al.: Data Requirements for Applying Machine Learning to Energy Disaggregation. Energies. 12, 9, 1696 (2019). https://doi.org/10.3390/en12091696.
17. Kramer, M.A.: Nonlinear principal component analysis using autoassociative neural networks. AIChE Journal. 37, 2, 233–243 (1991). https://doi.org/10.1002/aic.690370209.
18. Huss, A.: Hybrid Model Approach to Appliance Load Disaggregation : Expressive appliance modelling by combining convolutional neural networks and hidden semi Markov models. (2015).
19. Kolter, J., Johnson, M.: REDD: A Public Data Set for Energy Disaggregation Research. Artif. Intell. 25, (2011).
20. Makonin, S. et al.: AMPds: A public dataset for load disaggregation and eco-feedback research. In: 2013 IEEE Electrical Power Energy Conference. pp. 1–6 (2013). https://doi.org/10.1109/EPEC.2013.6802949.
21. Kelly, J., Knottenbelt, W.: The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes. Sci Data. 2, 1, 150007 (2015). https://doi.org/10.1038/sdata.2015.7.
22. Shin, C. et al.: The ENERTALK dataset, 15 Hz electricity consumption data from 22 houses in Korea. Scientific Data. 6, 1, 193 (2019). https://doi.org/10.1038/s41597-019-0212-5.
23. Batra, N. et al.: NILMTK: An Open Source Toolkit for Non-intrusive Load Monitoring. Proceedings of the 5th international conference on Future energy systems. 265–276 (2014). https://doi.org/10.1145/2602044.2602051.
24. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs]. (2014).
25. Koutroumpina, C.: Intelligent way of managing energy data flow. University of Patras (2020).
26. sklearn.preprocessing.minmax_scale — scikit-learn 0.24.2 documentation, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.minmax_scale.html, last accessed 2021/06/22.
27. Google, Inc: tensorflow: TensorFlow is an open source machine learning framework for everyone.
28. Hinton, G.E. et al.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580 [cs]. (2012).
29. Glorot, X. et al.: Deep Sparse Rectifier Neural Networks. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. pp. 315–323 JMLR Workshop and Conference Proceedings (2011).
30. Angelis, G.F. et al.: NILM Applications: Literature Review of Learning Approaches, Recent Developments and Challenges. Energy and Buildings, 261:111951 (2022).
31. Tabatabaei, S.M. et al.: Toward Non-Intrusive Load Monitoring via Multi-Label Classification. IEEE Transactions on Smart Grid. 8, 1, 26–40 (2017). https://doi.org/10.1109/TSG.2016.2584581.
32. Ayub M. and El-Alfy E.S.M.: Multi-Target Energy Disaggregation using Convolutional Neural Networks. Int. Journal of Advanced Computer Science and Applications, 11(10):684-693 (2020).