Learning to Automatically Identify Home Appliances

Dan Lorbek Ivančič[1], Blaž Bertalanič[1,2], Gregor Cerar[1], Carolina Fortuna[1]

[1]*Jozef Stefan Institute, Ljubljana, Slovenia*
[2]*Faculty of Electrical Engineering, University of Ljubljana, Slovenia*
E-mail: dl0586@student.uni-lj.si

Abstract. Appliance load monitoring (ALM) is a technique that enables increasing the efficiency of domestic energy usage by obtaining appliance specific power consumption profiles. While machine learning have been shown to be suitable for ALM, the work on analyzing design trade-offs during the feature and model selection steps of the ML model development is limited. In this paper we show that 1) statistical features capturing the shape of the time series, yield superior performance by up to 20 percentage points and 2) our best deep neural network-based model slightly outperforms our best gradient descent boosted decision trees by 2 percentage points at the expense of increased training time.

1 Introduction

Household energy consumption accounts for a large proportion of the world's total energy consumption. The first studies, conducted as early as the 1970s, showed that as much as 25% of national energy was consumed by our domestic appliances alone. This figure rose to 30% in 2001 [1] and continues to increase with an exponential rate. Some researchers even predict that these numbers will double by 2030 [2].

In support of rationalizing consumption, appliance load monitoring (ALM) has been introduced. It aims to help solve domestic energy usage related issues by obtaining appliance specific power consumption profiles. Such data can help devise load scheduling strategies for optimal energy utilization [2]. Additionally, data about appliance usage can provide useful insight into daily activities of residents which can be useful for long-distance monitoring of elderly people who prefer to stay at home rather than going to retirement homes [2]. Other applications include theft detection, building safety monitoring, etc.

The two different ways of realizing ALM are intrusive load monitoring (ILM) and non-intrusive load monitoring (NILM). While ILM is known to be more accurate, it requires multiple sensors throughout the entire building to be installed which incurs extra hardware cost and installation complexity. NILM, however, is a cost-effective, easy to maintain process for analyzing changes in the voltage [3] and current going into a building without having to install any additional sensors on different household devices, since it operates using only data obtained from the single main smart meter in a building. The obtained data is then disaggregated and each individual appliance and its energy consumption are detected.

One promising approach to ILM for automatic identification of home appliances is the use of machine learning (ML). For instance, in [4] they used ML to find patterns in the data and extract useful information such as type of load, electricity consumption detail and the running conditions of appliances [4]. More recently, [5] focused on the study of design trade-offs during the feature and model selection steps of the development of the ML-based classifier for ILM. In their study they considered various statistical summaries for feature engineering and classical machine learning techniques for model selection. We complement the work in [5] by extending the feature set with additional shape capturing values and considering deep learning (DNN) and gradient boosted trees (XGBoost) as promising modelling techniques. The contributions of this paper are as follows:

- We explore a variety of different statistical features and show the ones capturing the shape of the time series, such as *longest strike above mean, longest strike below mean, absolute energy and kurtosis* yield superior performance by up to 20 percentage points.
- We show that our best DNN based model slightly outperforms our best XGBoost by 2 percentage points at the expense of increased training time. We also show that our models outperform the results from [5] by 5 percentage points.

The paper is organized as follows. Section 2 summarizes related work, Section 3 formulates the problem and provides methodological details, Section 4 focuses on the study of feature selection trade-offs, while Section 5 discusses model selection. Concluding remarks are drawn in Section 6.

2 Related Work

Existing work that uses machine learning for ALM, such as in [6] investigates the performance of deep learning neural networks on NILM classification tasks and builds a model that is able to accurately detect activations of common electrical appliances using data from the smart meter. More complex DNNs for NILM classification tasks are presented by the authors in [3], where they introduce a Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) based model and show that it outperforms the considered baselines. In [7] they approach a similar problem by proposing a convolutional neural network based model that allows simultaneous detection and classification of events without having to perform double processing. In [8] authors train a temporal convolutional neural network to automatically extract high-level load signatures for individual appliances while in [9] a feature extraction method is presented using multiple parallel convolutional layers as well as an LSTM recurrent neural network based model is proposed.

3 Problem formulation

Our goal was to design a classifier that when given an input time series T, it is able to accurately map this data to the appropriate class C, as shown in equation 1.

$$C = \Phi(T) \tag{1}$$

where Φ represents the mapping function from time series to target classes and C is a set of these classes, where each class corresponds to one of the following household appliances: computer monitor, laptop computer, television, washer dryer, microwave, boiler, toaster, kettle and fridge. The appliances and measured data illustrated in Figure 1 available in the public UK-Dale dataset are used. The UK DALE (Domestic Appliance-level Electricity) contains the power demand from 5 different houses in the United Kingdom. The dataset was build at a sample-rate of 16 Hz for the whole-house and 0.1667 Hz for each individual appliance. Data is spread into 1 hour long segments, each dataset sample contains a time series with 600 datapoints as depicted in Figure 1.

For realizing Φ, we perform first a feature selection task followed by a model selection one. For selecting the best feature set, we perform feature selection in Section 4. For model selection, we go beyond the work in [5] and consider deep learning architectures enabled by Tensorflow and advanced decision trees that use on optimized distributed gradient boosting technique available in the XGBoost open source library as detailed in Section 5.

4 Feature selection

As can be seen in Figure 1, the time-series corresponding to each device has unique shape and patterns, therefore an intuitive approach to feature selection is to extract statistical properties of the time series that would capture the unique properties of the signals. For instance, a summary such as the peak-to-peak value is able to capture the difference between the maximum and minimum value in a time series signal while one such as skewness is able to describe the asymmetry in the distribution of datapoints in a particular sample. A good combination of such feature would be able to inform the model with relevant information about the power consumption of each appliance, making it easier to find patterns in the data and perform classification task more accurately. Recently, standard tools for computing a large range of such summaries

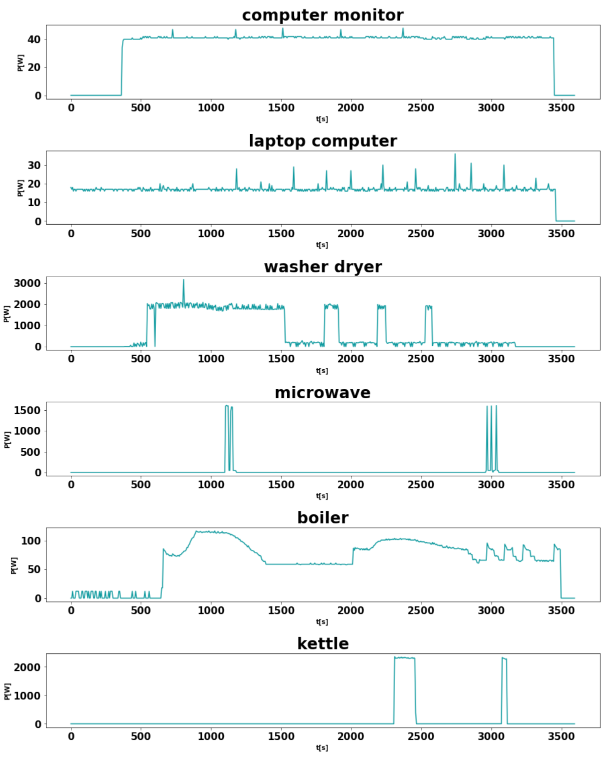

Figure 1: Selected appliances, showing power in relation to time over a 1 hour interval.

are provided by dedicated time series feature engineering tools such as tsfresh [1].

Following an extensive evaluation of combinations of time-series, we report the results for a representative selection of three feature sets as follows:

FeatureSet1 - This feature set consists of the raw time series, containing 2517 time series samples, each with 600 datapoints. It is used as a baseline to see the performance achieves with the available data.

FeatureSet2 - This feature set consists of: *mean value*, *maximum*, *minimum*, *standard deviation*, *variance*, *peak − to − peak*, *count above mean*, *count below mean*, *mean change*, *absolute mean change*, *absolute energy*. The count above and below mean counts the numbers of values in each sample that are higher or lower than the mean value of that same sample and helps quantifying the width of a pulse such as the ones for the toaster and microwave from Figure 1. The mean absolute change gives the mean over the absolute differences between subsequent time series values. The absolute energy represents the sum of squared values, calculated using formula shown in equation 2 and provides the information on whether a specific appliance has large consumption profile or not.

$$E = \sum_{i=0}^{n-1} x_i^2 \tag{2}$$

FeatureSet3 - After taking a deeper look into the features from FeatureSet2, we noticed that minimum is re-

[1]*https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html*

dundant as it is usually zero in every sample and peak-to-peak is in most cases equal to maximum value due to the lowest value mostly being zero. This feature set consists of: $maximum$, $standard\ deviation$, $mean\ absolute\ change$, $mean\ change$, $longest\ strike\ above\ mean$, $longest\ strike\ below\ mean$, $absolute\ energy$, $kurtosis$, $number\ of\ peaks\ in\ each\ signal$. The longest strike above and below mean returns the length of the the longest consecutive subsequence that is higher or lower than the mean value of that specific sample. The kurtosis is another metric of describing the probability distribution and measures how heavily the tails of a distribution differ from the tails of a normal distribution.

Table 1: Feature comparison using the best models.

Model	Feature set	Precision	Recall	f1
DNN3	FeatureSet1	0.638	0.595	0.573
XGB3	FeatureSet1	0.799	0.769	0.779
DNN3	FeatureSet2	0.918	0.885	0.889
XGB3	FeatureSet2	0.869	0.864	0.867
DNN3	FeatureSet3	**0.931**	**0.898**	**0.902**
XGB3	FeatureSet3	**0.888**	**0.889**	**0.889**
DNN3	best[5]	0.893	0.887	0.888
XGB3	best[5]	0.861	0.860	0.861
SVM[5]	best[5]	0.851	0.835	0.834

4.1 Results

The results of the feature selection process are listed in Table 1 for the two techniques considered in this paper. As can be seen from the second column of the table entitles instances, the dataset is balanced. From columns 3-5 it can be seen that for the baseline FeatureSet1, the f1 score is 0.57 for the CNN and 0.77 for XGB. By using features that better capture the shape of the time series such as in the case of FeatureSet2, an improvement of up to 20% can be seen as follows: the f1 of the CNN model increasing to 0.89, the precision 0.92 and recall to 0.88. The XGBoost model also performed better with an f1 of 0.87, precision of 0.87 and recall of 0.86. Finally, it can be seen from the table that FeatureSet3 performs the best with the f1 of 0.90, precision of 0.93 and recall of 0.90 for the CNN model and f1 of 0.89, precision of 0.89 and recall of 0.89 for the XGB model. FeatureSet3 performed better than FeatureSet2 because its features had much less correlation between each other as well as all of the redundant features from FeatureSet2 were removed. For FeatureSet3, a variety of different feature orderings were also tested but the results remained more within 1% accuracy variance.

To gain insights into the per class performance of FeatureSet3 with the two techniques, we present per device f1 score breakdown in Table 2. It can be seen that computer monitor, microwave and kettle are classified worst by all three models, as their similar consumption profiles make it difficult for the models to distinguish between them. Nevertheless, the CNN classifies all three the best due to its superior pattern recognition ability.

Table 2: Per class performance, FeatureSet3 vs best [5]

Class	Inst.	CNN f1	XGB f1	[5] f1
monitor	300	0.827	0.833	0.780
laptop	276	0.983	0.932	0.838
television	300	0.992	0.976	0.941
washer/dryer	226	0.941	0.912	0.804
microwave	300	0.688	0.620	0.687
boiler	300	1.000	0.968	0.940
toaster	215	0.949	0.940	0.806
kettle	300	0.756	0.722	0.739
fridge	300	1.000	0.983	0.970

5 Model selection

For analyzing the performance of DNN and XGBoost for our problem we conducted extensive performance evaluations. We started by developing a deep learning sequential model, which at first consisted of three dense layers, each with an arbitrarily chosen number of neurons. By trying different combinations of hyperparameters such as number of neurons, loss functions, optimizers, batch size, number of epochs, number of layers and learning rate, we came closer to finding the best suited model for our problem. For optimizing certain hyperparameters we took advantage of the automatic hyperparameter optimization framework Optuna [2]. We then applied similar optimization techniques on the XGB model, although it's default parameter configuration already gave good results. All the experiments were ran on Google Colab using an instance with Nvidia Tesla K80 GPU and 12.69 GB of RAM.

In this section we present and analyze three representative models from each class, DNN and XGboost respectively.

5.1 Deep neural network

DNN1 - This model consisted of three fully connected dense layers. The first two had 32 neurons each as well as ReLU (rectified linear unit) activation function, while the output layer had nine neurons, each corresponding to one of the nine possible appliances and Softmax activation function.

DNN2 - For this model we took the DNN1 model and added an additional dense layer with 64 neurons as well as changed the activation function to linear in the penultimate layer. With this additional complexity we expected to see better results.

DNN3 - For this model we introduced two 1D convolution layers, first with 128 filters and second with 64. Then we used a flatten layer to reduce the dimensionality of the output space, and make the data compatible with the following dense layer, followed by another (output) dense layer.

[2] $https://optuna.org$

5.2 XGBoost

XGB1 - This is the model with standard configuration, i.e. maximum depth of 3, 100 estimators and learning rate of 0.1.

XGB2 - In this model we increased the maximum depth to 4 as well as first reduced learning rate by 50% (to 0.05) and then increased the number of estimators by 50% (to 200). Doing this gave slightly better results.

XGB3 - For this model we decreased the maximum depth to 2, increased number of estimators to 500 and learning rate to 0.25.

Table 3: Model performance on FeatureSet3.

Model	Precision	Recall	f1	Comp. time
DNN1	0.866	0.851	0.846	10.972s
DNN2	0.900	0.887	0.889	21.026s
DNN3	**0.931**	**0.898**	**0.902**	21.124s
XGB1	0.876	0.863	0.864	1.126s
XGB2	0.884	0.881	0.882	2.518s
XGB3	**0.888**	**0.889**	**0.889**	3.225s
SVM [5]	0.878	0.852	0.852	0.301s

5.3 Results

5.3.1 Classification performance

The classification performance of the models is provided in Table 3. It can be seen that the best performing models are DNN3 with an f1 score of 0.90 and XGB3 with an f1 of 0.88. However, the computation time of XGB3 is only 3.23s while for DNN3 it is 21.12s. The XGB classifier using classical machine learning performed only about 1 percentage point worse than the CNN model, while at the same time being much less complex and able to complete the entire training process about 18 seconds faster than the CNN. In addition, the XGB model is much easier to optimize since it has no hidden layers and a prearranged hyperparameter configuration that usually requires no further optimization at all. From the last line of the table it can be seen that the SVM-based model from [5] performs 5 percentage points less than DNN3 on FeatureSet3.

5.3.2 Computation time

The superior performance of the DNN model comes at a cost of increased algorithm complexity and hence longer computation time. As depicted in Table 3 the first DNN model took 10.97 seconds to complete the training process and the best (most complex one) took 21.12 seconds. XGBoost, on the other hand, was much faster with XGB1 taking only 1.12 seconds. The added depth for the XGB2 caused a slight increase in computation time to 2.52 seconds, which further increased to 3.23 seconds due to the high number of estimators used in XGB3. Finally, the state of the art was the fastest to complete the training process taking only 0.3 seconds but scored the worst in terms of performance.

6 Conclusions

In this paper we investigated the design trade-offs during the feature and model selection steps of the development of the ML-based classifier for ILM. After formulating our problem, we first show that by extracting various statistical features from raw time series data and then training our models with these features, we were able to improve f1 score by up to 20 percentage points.

Second, we propose two different ML techniques and our process of developing the proposed models using these. We show that optimizing hyperparameters to better suit our specific problem can improve their respective performance by around 4 percentage points. However, choosing the right features that better capture the shape of the data has a much greater impact on the end results than optimizing the models. We also show that classical machine learning model does not perform significantly worse than the deep neural network based one, while at the same time being less computationally expensive.

References

[1] L. Shorrock, J. Utley *et al.*, *Domestic energy fact file 2003*. Citeseer, 2003.

[2] A. Zoha, A. Gluhak, M. A. Imran, and S. Rajasegarar, "Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey," *Sensors*, vol. 12, no. 12, pp. 16 838–16 866, 2012.

[3] J. Kim, T.-T.-H. Le, and H. Kim, "Nonintrusive Load Monitoring Based on Advanced Deep Learning and Novel Signature," *Computational Intelligence and Neuroscience*, vol. 2017, p. e4216281, Oct. 2017, publisher: Hindawi. [Online]. Available: https://www.hindawi.com/journals/cin/2017/4216281/

[4] E. Aladesanmi and K. Folly, "Overview of non-intrusive load monitoring and identification techniques," *IFAC-PapersOnLine*, vol. 48, no. 30, pp. 415–420, 2015, 9th IFAC Symposium on Control of Power and Energy Systems CPES 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896315030566

[5] L. Ogrizek, B. Bertalanic, G. Cerar, M. Meza, and C. Fortuna, "Designing a machine learning based non-intrusive load monitoring classifier," in *2021 IEEE ERK*, 2021, pp. 1–4.

[6] M. Devlin and B. P. Hayes, "Non-intrusive load monitoring using electricity smart meter data: A deep learning approach," in *2019 IEEE Power Energy Society General Meeting (PESGM)*, 2019, pp. 1–5.

[7] F. Ciancetta, G. Bucci, E. Fiorucci, S. Mari, and A. Fioravanti, "A new convolutional neural network-based system for nilm applications," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–12, 2021.

[8] Y. Yang, J. Zhong, W. Li, T. A. Gulliver, and S. Li, "Semisupervised multilabel deep learning based nonintrusive load monitoring in smart grids," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 11, pp. 6892–6902, 2020.

[9] W. He and Y. Chai, "An empirical study on energy disaggregation via deep learning," *Advances in Intelligent Systems Research*, vol. 133, pp. 338–342, 2016.