RESEARCH

Pool Detection from Smart Metering Data with Convolutional Neural Networks

Cornelia Ferner*, Günther Eibl, Andreas Unterweger, Sebastian Burkhart and Stefan Wegenkittl

*Correspondence:
cornelia.ferner@fh-salzburg.ac.at
Salzburg University of Applied Sciences,Urstein Sued 1, 5412 Puch/Salzburg,Austria
Full list of author information is available at the end of the article

Abstract

The nationwide rollout of smart meters in private households raises privacy concerns: Is it possible to extract privacy-sensitive information from a household's power consumption? For a small sample of 869 Upper Austrian households, information about consumption-heavy amenities and household characteristics are available. This work studies the detection of households with swimming pools (the most common amenity in the dataset) using Convolutional Neural Networks (CNNs) applied on load heatmaps constructed from load profiles. Although only a small dataset is available, results show that by using CNNs, privacy can be broken automatically, i.e., without the time-consuming, manual feature generation. The method even slightly outperforms a previous approach that relies on a nearest neighbor classifier with engineered features.

Keywords: smart metering; convolutional neural network; privacy

Introduction

Based on a decision of the European Union, the Austrian Government aims at a 95% coverage for the smart meter rollout in private households until 2022 [1]. Smart meters automatically record the load data in a 15-minute interval and, per default, report the numbers once a day. The automated communication raises customers' privacy concerns as knowledge is still sparse about what information can be extracted from 15-minute load data [2].

A 2017 survey [3] on smart meter data privacy provides a thorough overview of the major concerns related to the collection of load data. Especially, non-intrusive load monitoring (NILM [4]) could be potentially harmful, as the techniques aim at disaggregating load data to extract load patterns incurred by different devices [5], [6]. In addition to detecting appliances, previous works by Chen et al. [7] and Eibl et al. [8] demonstrate the possibility of detecting the presence or absence of residents, with their work being framed as "occupancy" and "holiday detection", respectively.

Chen et al. [9] showed that is possible to locate solar-powered homes based on their energy production signatures, which are location-dependent. For 14 households, they achieved 500-meter accuracy with one-second resolution and 28-kilometer accuracy with one-minute resolution, respectively.

Beckel et al. [10] showed that demographic data, e.g., the number of residents in a household or whether or not there are children in the household, can be extracted from 30-minute load data over a period of one and a half years. They achieved an accuracy of 70% over all 4232 households. For specific characteristics and appliances, the accuracy even exceeded 80%.

Burkhart et al. [11] investigated a sample dataset of 869 households from Upper Austria [12]: In addition to the 15-minute load readings, the dataset offers information about whether the household possesses any consumption-heavy amenities, such as a swimming pool, home cinema, aquarium or water bed. By comparing the households' consumption patterns to the characteristic load profile of a pool filter pump, they were able to correctly detect households with pools with a precision of 68.5% using a nearest neighbor classifier on manually designed features.

This work explores how to automate the pool detection process by training a convolutional neural network (CNN) [13] directly on the load data. While traditional machine learning methods usually require careful feature generation and selection before applying the classifier, deep learning methods such as the CNN process the raw data and learn both, how to extract appropriate features and how to classify, at the same time. Thus, no prior knowledge about the domain and data is needed and consequently time and manual effort can be saved.

Since CNNs are usually applied on very large datasets[1], it is not known how the methodology performs for a small dataset like the given one, where only 64 out of 869 households have a pool. The results for pool detection show that the deep learning approach can compete with the approach in [11] that uses manually designed features. While the practical consequences seem marginal, from a privacy perspective the implications are considerable: using CNNs private information can be inferred having both, a small dataset and a small time budget.

This paper is structured as follows: In the Background Section, we provide an overview of the pool detection method from [11] and CNNs. In the Experiments Section, we describe our proposed CNN for automated pool detection and the detection results for the dataset from [11]. In the Discussion and Future Work Section, we compare and discuss these results and provide an outlook to future research directions.

Background

This section describes the existing pool detection method that we use for comparison as well as CNNs that are employed in this work.

Existing Pool Detection Method

Pool detection has already been done in [11] by exploiting the fact that a swimming pool requires a filter pump. The detection method aims at finding filter pumps which (i) run in a regular manner over the warmest months of the year and (ii) consume several hundred Watts of electrical power, thus constituting a distinct part of the load.

An important prerequisite for the method is the representation of the time series as a heatmap, where the columns represent days and the rows represent the time of day. Using this representation, the two properties of the pool pump stated above result in more (Fig. 1) or less (Fig. 2) regular rectangular shapes overlaying the remaining load.

[1]Two famous datasets are MNIST and ImageNet. MNIST (http://yann.lecun.com/exdb/mnist/) is a dataset for classifying handwritten digits and comprises 60000 images. ImageNet (http://www.image-net.org/) is a collection of images in 27 high-level cateories and about 14 million images.

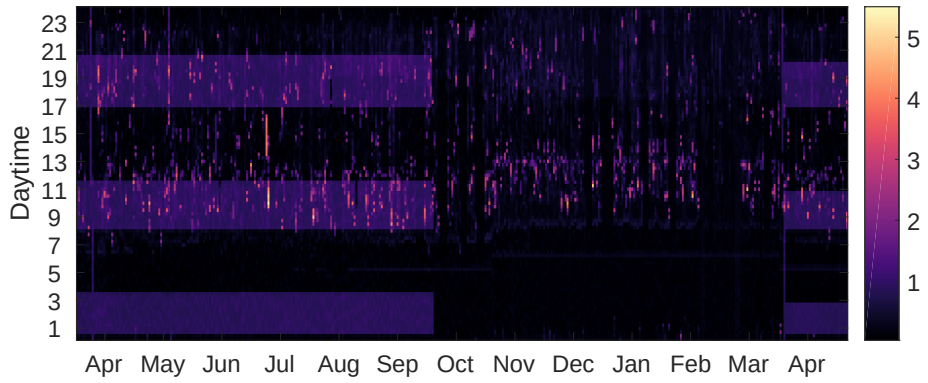

Figure 1 Heatmap showing the load [kW] of a household with a pool. This household's pool pump runs regularly so that the pool features of [11] can be extracted rather easily by detecting the rectangular shaped areas.

The algorithm detects these rectangular shapes by treating heatmaps as images and by applying image processing methods such as morphological operations. Using a carefully engineered, sophisticated sequence of steps, the authors achieve a recall of 68% (Fig. 4 (left)).

Convolutional Neural Networks (CNN)

Deep neural networks [14] differ from other machine learning algorithms in that they process the raw data directly, i.e., without an explicit feature generation step. The inherent feature generation is one of the big advantages of neural networks, as results can be obtained more quickly without the need for manually engineered features. The raw data passes through the layers of the network which interact in a non-linear fashion.

Feature Generation

In common network architectures, the first few layers extract features from the given input vectors (4 layers, denoted as *Feature Generation* in Fig. 3). In a convolutional neural network, as described in the following, the first layers are convolutional layers. Each layer consists of a number of filters that detect low level patterns in small spatial regions, e.g., in a 5×5 pixel area[2]. The output of the convolutional layer passes through the non-linear activation function. The most common activation function is the rectified linear unit (ReLU) [15] that assigns $f(x) = \max(0, x)$ to each input. The use of ReLU speeds up the training, because the function is easy to compute and to derive. Usually, a pooling layer follows the convolutional layer (see Fig. 3). Pooling reduces the feature size in the network and prevents overfitting. The most common form is max pooling which chooses the highest value in a given window [13].

Classification

The subsequent classification layers (2 layers, denoted as *Classification* in Fig. 3), follow a more traditional, dense setup, where each node of one layer layer is con-

[2]For each position of the convolutional filter, the inner product of the filter matrix and the input data is computed.

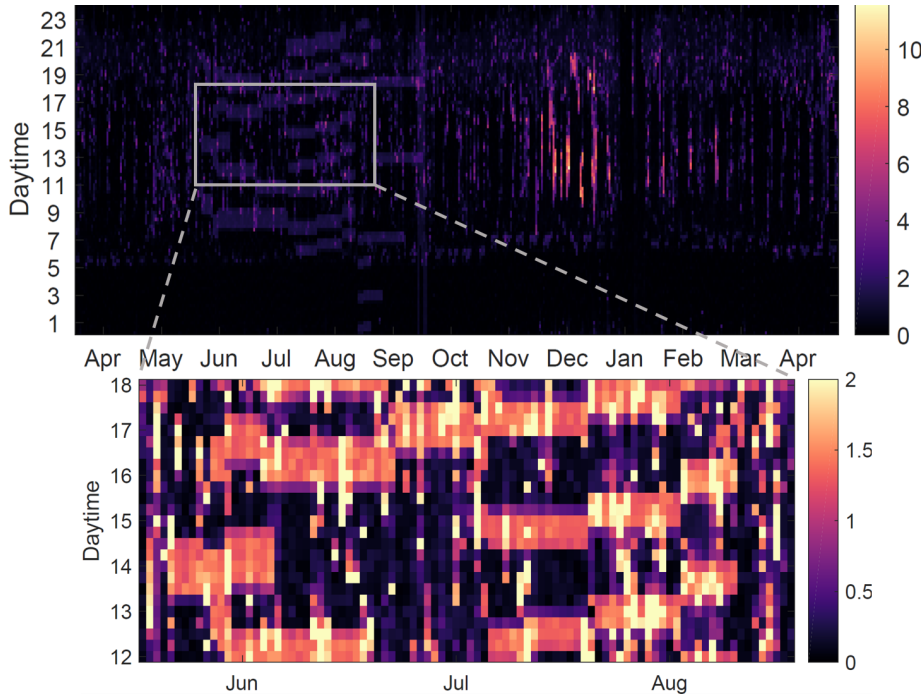

Figure 2 *top:* This household's pool pump runs less regularly such that the features are more difficult to detect from the load in the top graph. *bottom:* The maximum value of the detail below is set to 2 kW to make the rectangular pool pump pattern more visible.

nected with every node of the subsequent layer to learn accurate decision boundaries in the feature space (fully connected). In order to prevent overfitting to the training data, a dropout layer [16] is inserted between two fully connected layers. Dropout means to randomly choose a subset of nodes that is ignored when updating the node weights. The final classification layer, typically a softmax layer [14], which predicts a vector of probabilities corresponding to the output classes.

Training

The network's weights are trained by optimizing a loss function based on the error between the network's predicted output and the target classes. The optimization algorithm iteratively updates the networks' weights to minimize the classification error. If gradient-based loss optimization is used, the update is commonly known as backpropagation.

The trained network weights in the feature generation layers exploit redundant information in the input data. The output of each layer (activations) represents the discriminative information for classification of this input, thus leading to a good, discriminative representation of the original input.

Experiments

In the following, we present experiments to demonstrate the effectiveness of a CNN to generate suitable features from the input data without any prior knowledge about the problem domain. From a privacy perspective, the intriguing results are disadvantageous: The CNN is a privacy invader. While the development of the

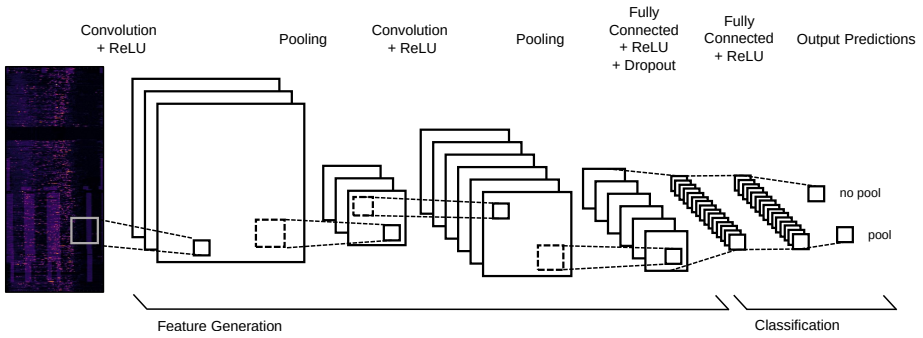

Figure 3 Sketch of the network architecture. The 96x395 heatmaps pass through 4 convolutional layers followed by two fully connected layers. For better visualization, the input data is depicted in color. Note that 4 convolutional layers plus pooling are used instead of the depicted 2.

method in [11] took several weeks, the development of the CNN is much faster, provided that the attacker has suitable skills and tools in both cases.

In order to compare our results to those in [11], we apply our method to the same dataset. It consists of load data (in kW) captured by smart meters with a time granularity of 15 minutes from 869 households in Upper Austria covering slightly more than one year (396 days). 64 of these households have a swimming pool. As in [11], each raw time series is converted into a 96 × 396 heatmap (e.g. Fig. 1). In contrast to their approach, we directly pass the heatmaps to the CNN network, so no further feature generation or preprocessing is applied.

Setup of the CNN

The setup loosely follows common basic CNN architectures, especially those of the AlexNet architecture described in [13]. We set the filter sizes to fit our input image size and to still have sufficiently many nodes left after the feature generation layers. However, as the focus lies on automating the classification process, the parameters and settings have not been further tuned.

More specifically, Figure 3 displays our network architecture[3]: We use 4 convolutional layers, each with batch normalization (see below), followed by ReLU and max pooling. The first convolution layers are based on 15 5×5 filters, the subsequent ones use 15 3×3 filters. Filters are moved each time by 1×1 pixels (=stride). At the border, values are computed by extending the image with zeros (padding). The window size of the first pooling layer is chosen to be 5×5 with a stride of 3×3, the latter ones are 3×3 with a stride of 2×2. The pooling layers reduce the feature size from originally 96·396 to 195 (= 15·13, i.e. number of filters · remaining heatmap pixels). The subsequent classification layer is a fully connected (dense) layer with 32 nodes with batch normalization and ReLU followed by a 50% dropout layer. The remaining fully connected layer reduces the size to two nodes corresponding to the two classes “pool” and “no pool”. This final layer includes the softmax function producing a distribution over the two classes.

[3]The network is implemented in Python 3 using PyTorch: https://pytorch.org/

During training, the cross entropy loss L [14] between the predicted output class $\hat{y}$ and the actual output y is computed:

$$L(y, \hat{y}) = -\frac{1}{\pi_y} y \log(\hat{y}) - \frac{1}{\pi_{1-y}} (1-y) \log(1-\hat{y}) \quad (1)$$

We account for the imbalanced relative amount of heatmaps per class in the dataset (denoted as π_y) by applying a weighting inverse to the class distribution to the cross entropy loss function. This weighting prevents the loss function from overemphasizing on the "no pool" class. Thus, using a binary output with "pool" and "no pool" coded as $y = 1$ and $y = 0$, respectively.

In order to minimize the loss function, the gradient-based Noam optimizer [18] is used that iteratively updates the weights based on an empirically estimated model size s and the current epoch p. It initially increases the learning rate[4] r for a number of warm-up steps w and finally decreases it:

$$r = s^{-0.5} \cdot \min\left(p \cdot w^{-1.5}, p^{-0.5}\right) \quad (2)$$

The Noam optimizer is parameterized with $w = 15$ and $s = 1000$. The training lasts for 100 epochs, i.e. 100 passes over the training data, and the data is split into batches of size 32. Data batches are used to reduce the input data size. As the network does not see the data at once, we apply batch normalization [17] after each layer to reduce the variance of input values to the next layer.

Results

We evaluate two settings: i) using the CNN as is for classification (CNN pure) and ii) extracting the CNN features (after the fourth convolutional layer) and classifying these features by means of a nearest neighbor classifier (CNN+k-NN). The latter version is intended to study whether the features obtained by the CNN are only suitable within a CNN setting or generalize for different classifiers. In order to compare our results with the ones of [11], the same evaluation methodology is used.

Table 1 provides the detailed classification results. As a baseline, we report the classification results from assigning all heatmaps to either the "pool" or the "no pool" class (all-positive and all-negative, respectively). Sin ce the classes are imbalanced, the accuracy in the all-negative class is already high and therefore, as discussed in [11], we aim for high precision.

The best classifier with manual features is the 5-nearest neighbors classifier, which achieves 94% accuracy and 68.5% precision [11]. When using CNN features with a k-nearest neighbor classifier, we can maintain the classification accuracy, but lose precision. The best classifier in terms of accuracy only achieves 60.6% of precision. However, the full CNN setting outperforms previous methods and yields 95.5% accuracy and 71.9% precision. Fig. 4 (right) shows the corresponding confusion matrix.

[4]Scaling value which determines to which extent weight values are changed. If the learning rate is too large, the optimizer will overshoot the minimum. If it is too slow, the training takes a long time.

Table 1 Classification results. a) Baseline classification by classifying all instances as "pool" or "no pool", respectively. b) Classification with manually designed features as in [11]. c) Nearest-neighbor classification with CNN features. d) CNN classification.

	Classification Method	Accuracy	Precision
a)	All-positive	10.5%	10.5%
	All-negative	89.5%	-
b)	SVM Gaussian	93.1%	66.7%
	5-NN	94.0%	68.5%
	1-NN	93.4%	66.7%
c)	CNN + 7-NN	93.1%	60.0%
	CNN + 5-NN	93.1%	57.7%
	CNN + 1-NN	93.4%	60.6%
d)	**CNN pure**	**95.5%**	**71.9%**

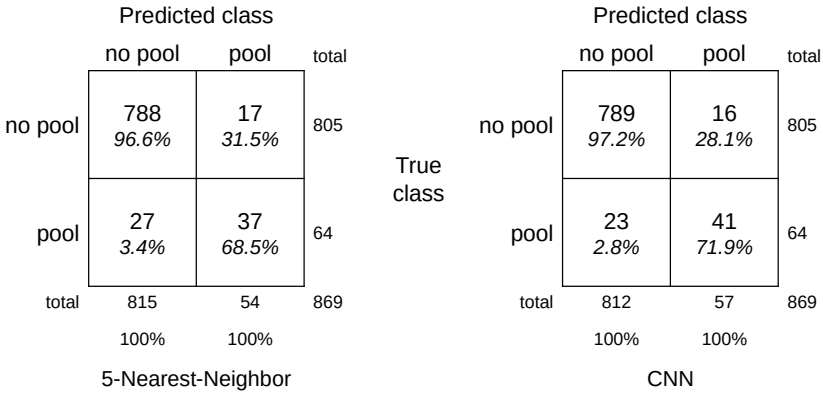

Figure 4 *left:* Confusion matrix for the 5-nearest-neighbor classification with manual feature engineering from [11]. The overall performance of the pool detection algorithm is 68.5% in terms of precision. The overall accuracy is 94.8%. *right:* Confusion matrix for the CNN classification: The overall performance is 71.9% in terms of precision. The overall accuracy is 95.5%.

In order to investigate the result more closely, we visually examined the heatmaps of the households with pools to search for the above mentioned rectangular patterns. For some cases, these patterns were not obvious and could only be seen after setting the maximum load to 2 kW (see Fig. 2 (bottom)), exploiting the fact that the load of the pump is lower than this value. From the 23 households with an undetected pool, 11 households did not show a rectangular pattern (which is true for only one of the 41 households with a detected pool). One interpretation of this insight is that the 11 households without the pattern might not be using their pools, i.e., the pool pump is not running. Households with inactive pools will of course not be detected by any method. Following this interpretation, the 11 misclassifications are label errors, not classification errors. Furthermore, the presence of such inactive pools could have the effect of label noise which significantly increases the classification difficulty.

Further experiments with other appliances (home cinema, aquarium) were conducted, but yielded poor results so far. One major issue is the low number of positive samples in the dataset, which is even lower than those of the swimming pools.

Discussion and Future Work

The results show that convolutional neural networks are able to competitively learn the presence of pools even in the case of a relatively small dataset that contains load profiles of 64 households with pools.

Deep learning is a quickly expanding field with a number of powerful methods to improve the classification process. While pool detection on the given dataset may have reached a limit, as argued above, one could try to predict more rare appliances using data augmentation methods. Generative adversarial networks (GAN) [19] are a means to produce artificial, new data but, to the best of our knowledge, have not been applied to time series data (in the form of heatmaps) so far. A siamese network architecture [20] could be used to train two subnetworks on a pair of heatmaps to decide whether they are from the same class or not. The learned features could be used for classification. The application of a siamese network drastically increases the amount of input data as potentially all 869^2 image combinations can be used.

From a privacy perspective, the good classification result is negative, since it shows that personal information can be inferred from data in an automatic way, i.e., without manual feature generation, which is typically the most time-consuming modeling step. From this perspective, it is interesting to investigate whether the features learned by the CNN (which showed worse results when combined with k-nearest-neighbor classifier) could reach the performance of the full CNN approach when used with another common classifier.

Similarly, but broadening the scope, it would be interesting to investigate whether the features generalize to other appliances, i.e., whether the features are indicative not only for pools. If so, the features could be used for transfer learning, meaning that the already trained CNN is reused by fixing the weights in the feature layers and only retraining the classification layers on the new task. Moreover, the transfer learning setting could be applied to similar datasets from other countries. A generalization across appliances or countries would have even more negative implications for privacy.

Acknowledgements
The authors would like to thank the *Energieinstitut* at the Johannes Kepler University Linz for providing the dataset.

Funding
The financial support by the Federal State of Salzburg is gratefully acknowledged.

Availability of data and materials
For privacy reasons, the data is not publicly available.

Author's contributions
This paper was written by Cornelia Ferner (40%), Günther Eibl (35%), Andreas Unterweger (10%), Sebastian Burkhart (5%) and Stefan Wegenkittl (10%).
The detailed contributions are as follows: The idea for the paper was developed by Andreas Unterweger (25%), Günther Eibl (25%), Stefan Wegenkittl (25%), Cornelia Ferner (15%) and Sebastian Burkhart (10%). The neural network architecture was developed by Cornelia Ferner (60%) and Stefan Wegenkittl (40%). The data cleaning was done by Sebastian Burkhart (90%) and Günther Eibl (10%). The Abstract was written by Cornelia Ferner (50%) and Günther Eibl (50%). The Introduction section was written by Günther Eibl (40%), Cornelia Ferner (30%) and Andreas Unterweger (30%). The Background section on the Existing Pool Detection Method was written by Günther Eibl (90%) and Cornelia Ferner (10%). The Background section on Convolutional Neural Networks was written by Cornelia Ferner (50%), Stefan Wegenkittl (30%) and Günther Eibl (20%). The Experiment section on the setup of the CNN was written by Cornelia Ferner (50%), Günther Eibl (30%) and Stefan Wegenkittl (20%). The Experiment section on the results was written by Cornelia Ferner (50%) and Günther Eibl (50%). The Discussion and Future Work section was written by Cornelia Ferner (50%) and Günther Eibl (50%).

Competing interests
The authors declare that they have no competing interests.

References

1. RIS - Rechtsinformationssystem des Bundes: Rechtsvorschrift für Intelligente Messgeräte-Einführungsverordnung. online. https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20007808 (2019)
2. Eibl, G., Engel, D.: Influence of Data Granularity on Smart Meter Privacy. IEEE Transactions on Smart Grid **6**(2), 930–939 (2015)
3. Asghar, M.R., Dán, G., Miorandi, D., Chlamtac, I.: Smart meter data privacy: A survey. IEEE Communications Surveys & Tutorials **19**(4), 2820–2835 (2017)
4. Hart, G.W.: Nonintrusive Appliance Load Monitoring. Proceedings of the IEEE **80**(12), 1870–1891 (1992)
5. Molina-Markham, A., Shenoy, P., Fu, K., Cecchet, E., Irwin, D.: Private memoirs of a smart meter. In: Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building. BuildSys '10, pp. 61–66. ACM, New York, NY, USA (2010)
6. Lisovich, M.A., Wicker, S.B.: Privacy Concerns in Upcoming Residential and Commercial Demand-Response Systems. In: Clemson Power Systems Conference (2008)
7. Chen, D., Barker, S., Subbaswamy, A., Irwin, D., Shenoy, P.: Non-intrusive occupancy monitoring using smart meters. In: Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings, pp. 1–8 (2013). ACM
8. Eibl, G., Burkhart, S., Engel, D.: Unsupervised holiday detection from low-resolution smart metering data. In: ICISSP, pp. 477–486 (2018)
9. Chen, D., Iyengar, S., Irwin, D., Shenoy, P.: Sunspot: Exposing the location of anonymous solar-powered homes. In: Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments. BuildSys '16, pp. 85–94. ACM, New York, NY, USA (2016)
10. Beckel, C., Sadamori, L., Staake, T., Santini, S.: Revealing household characteristics from smart meter data. Energy **78**, 397–410 (2014)
11. Burkhart, S., Unterweger, A., Eibl, G., Engel, D.: Detecting swimming pools in 15-minute load data. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp. 1651–1655 (2018). IEEE
12. Azarova, V., Engel, D., Ferner, C., Kollmann, A., Reichl, J.: Exploring the impact of network tariffs on household electricity expenditures using load profiles and socio-economic characteristics. Nature Energy, 1 (2018)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
14. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning, pp. 1–716. MIT press, Cambridge, MA (2016)
15. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp. 807–814 (2010)
16. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**, 1929–1958 (2014)
17. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. ICML'15, pp. 448–456 (2015)
18. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
19. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
20. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML Deep Learning Workshop, vol. 2 (2015)