Forecasting residential gas consumption with machine learning algorithms on weather data

Conference Paper · May 2019

CITATIONS
0

READS
498

7 authors, including:

Brian de Keijzer
The Hague University of Applied Sciences
2 PUBLICATIONS **1** CITATION
SEE PROFILE

Pol de Visser
Hogeschool The Hague
2 PUBLICATIONS **1** CITATION
SEE PROFILE

Victor Garcia Romillo
Universidad del País Vasco / Euskal Herriko Unibertsitatea
3 PUBLICATIONS **1** CITATION
SEE PROFILE

Megan Meezen
Hogeschool The Hague
2 PUBLICATIONS **1** CITATION
SEE PROFILE

Forecasting residential gas consumption with machine learning algorithms on weather data

Brian de Keijzer[1], *Pol* de Visser[1], *Víctor* García Romillo[2], *Víctor* Gómez Muñoz[3], *Daan* Boesten[1], *Megan* Meezen[1] and *Tadeo Baldiri* Salcedo Rahola[1*]

[1]Faculty of Technology, Innovation and Society, The Hague University of Applied Sciences, Rotterdamseweg 137, 2628 CN, Delft, The Netherlands
[2]Faculty of Engineering, University of the Basque Country, Paseo Rafael Moreno 3 48013 Bilbao, Vizcaya, Spain
[3]Escuela Politécnica Superior, University Francisco de Vitoria, Carretera Pozuelo a Majadahonda, Pozuelo de Alarcón, Madrid, Spain

Abstract - Machine learning models have proven to be reliable methods in the forecasting of energy use in commercial and office buildings. However, little research has been done on energy forecasting in dwellings, mainly due to the difficulty of obtaining household level data while keeping the privacy of inhabitants in mind. Gaining insight into the energy consumption in the near future can be helpful in balancing the grid and insights in how to reduce the energy consumption can be received. In collaboration with OPSCHALER, a measurement campaign on the influence of housing characteristics on energy costs and comfort, several machine learning models were compared on forecasting performance and the computational time needed. Nine months of data containing the mean gas consumption of 52 dwellings on a one hour resolution was used for this research. The first 6 months were used for training, whereas the last 3 months were used to evaluate the models. The results showed that the Deep Neural Network (DNN) performed best with a 50.1 % Mean Absolute Percentage Error (MAPE) on a one hour resolution. When comparing daily and weekly resolutions, the Multivariate Linear Regression (MVLR) outperformed other models, with a 20.1 % and 17.0 % MAPE, respectively. The models were programmed in Python.

1 Introduction

In recent years the European Commission has set ambitious CO_2 emission reduction targets. As a consequence, the Dutch government has been increasingly regulating the energy sector. The Netherlands aims to be free of natural gas use by the end of 2050 [1]. Therefore, the importance of diminishing natural gas consumption in dwellings is growing.

Gaining insight in the prediction of gas consumption in dwellings is critical to meet the Dutch government requirements. However, a vast number of research has been conducted towards energy consumption prediction in commercial and office buildings [2, 3, 4, 5, 6]. Thus far, little research has been conducted to predict gas consumption on single household level due to the difficulty of obtaining the energy use data due to data privacy [7]. From commercial and office building energy forecasting research, Alberto Hernandez Neto [3] concluded that deep neural networks outperform physical simulation models by 3-6 %. Furthermore, according to Jurado López [8], weather conditions have the highest correlation to gas use. It is worth noting that using multiple weather parameters as features in prediction models improve the accuracy of the model [3]. This paper focusses on comparing the accuracy of widely discussed machine learning algorithms on an hourly, daily and weekly resolution and ultimately, creating a model that can predict the natural gas use of dwellings as accurate as possible. Different models were made and their results were evaluated. This was done with a dataset containing nine months of the mean hourly gas consumption data from a total of 52 dwellings in The Netherlands, together with the mean hourly weather data of a nearby weather station. In this paper the models used as less features as possible, to minimize the computation power required, which enables the models to be able to run on computers with limited computer power in dwellings.

2 Methodology

This paper focusses on the evaluation of different machine learning models. Specifically the Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) and Convolutional Neural Network (CNN). These deep learning algorithms have shown the best potential for this type of prediction [1, 5, 6, 9, 4]. In most of the previously listed references, the models were based on electricity prediction of commercial and office buildings. These algorithms are called complex models in literature. The accuracy of the models is dependent on the environment of the research and chosen features. Consequently, the

* Corresponding author: t.b.salcedorahola@hhs.nl

models need to be verified on residential households because this environment is different from commercial buildings. Commercial buildings usually have a fixed energy use time schedule, while dwellings have more variation in its energy use pattern [8].

In addition to the complex models, Multivariate Linear Regression (MVLR) and Deep Neural Network (DNN) have shown promising results in comparison to their simplicity [5, 10]. These models are called simple models in literature. The different models are discussed in their respective subchapters.

In this section the way the data was collected and processed is discussed. Insights are given on what machine learning methods were applied and which metrics were used for the evaluation of the models. The same metrics were used to evaluate the different machine learning models. Furthermore, this section clarifies the commonly used environmental settings for all the models. All the models were programmed in Python.

2.1 Data collecting and processing

The data used in this research was gathered by the OPSCHALER project from 2017-02 to 2017-12. It contained data concerning gas and electricity consumption. This information was obtained from the smart meters of 52 different dwellings, everything was anonymized and therefore the dwelling locations were unknown. The data acquisition period from the dwellings varied from one to nine months. The electricity data was sampled with a ten second resolution, whereas the gas consumption was sampled with a one hour resolution. These sampling periods are visualized in *Figure 1*.

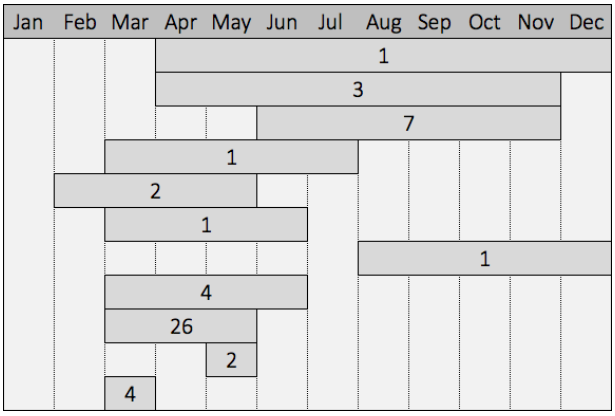

Fig. 1. Distribution of the data acquisition from all different dwellings. The number in each grey selected area is the number of dwellings of which data was gathered from during this period.

Due to an uneven and scarce distribution of the data, the mean of all dwellings at each timestamp was taken along with its standard deviation. This represented the mean gas use on the aggregated level, e.g. a block of 52 dwellings.

In addition to this, weather information obtained from the Royal Netherlands Meteorological Institute (KNMI) station in Rotterdam was used. This was the weather station most nearby to all dwellings. However, the distance to the most distant dwelling was 103 km. The weather information was sampled with a 15 minute resolution.

One of the goals of this research was to use as less features as possible, therefore a selection of the used parameters was made. Parameters referred to all the available variables from the original dataset, whereas features referred to the variables that were selected to forecast the target 'gasUse'. According to [12, 13] air temperature and calendar related features were the most relevant to use when predicting gas consumption in the residential sector. In this research the target was called 'gasUse', being this the gas consumption in an interval of an hour and measured in $[m^3]$. As a baseline, the parameter which had the highest Pearson correlation coefficient P with 'gasUse' was selected as a feature. In this case this parameter was the temperature T. The other parameters that had a Pearson correlation coefficient with the temperature that suffixes $|P| < 0.1$, were selected. Setting this threshold to $|P| < 0.1$ minimized the influence of the features on each other. All the used parameters are visible in *Table 1*.

Table 1. Features extracted from the available parameters. *Values have been calculated.

Features	Unit	Description
FF	m/s	Wind speed at 10 m
RG	mm/h	Rain intensity
T	°C	Temperature at 1,5 m (1 minute mean)
hour of day	-	Hour of day at given the timestamp*.
day of week	-	Day of week at given the timestamp*.
season	-	Season of the year at given the timestamp*.

The weather data was sampled with a 15 minutes resolution, whereas the 'gasUse' was sampled with a one hour resolution. To combine these two datasets into one, the weather data was down sampled to one hour by mean and combined with the 'gasUse'. Not a Number (NaN) values appeared in the intervals of time where the data acquisition system stopped gathering information, this was the case for the features and target. All NaNs were removed from the dataset.

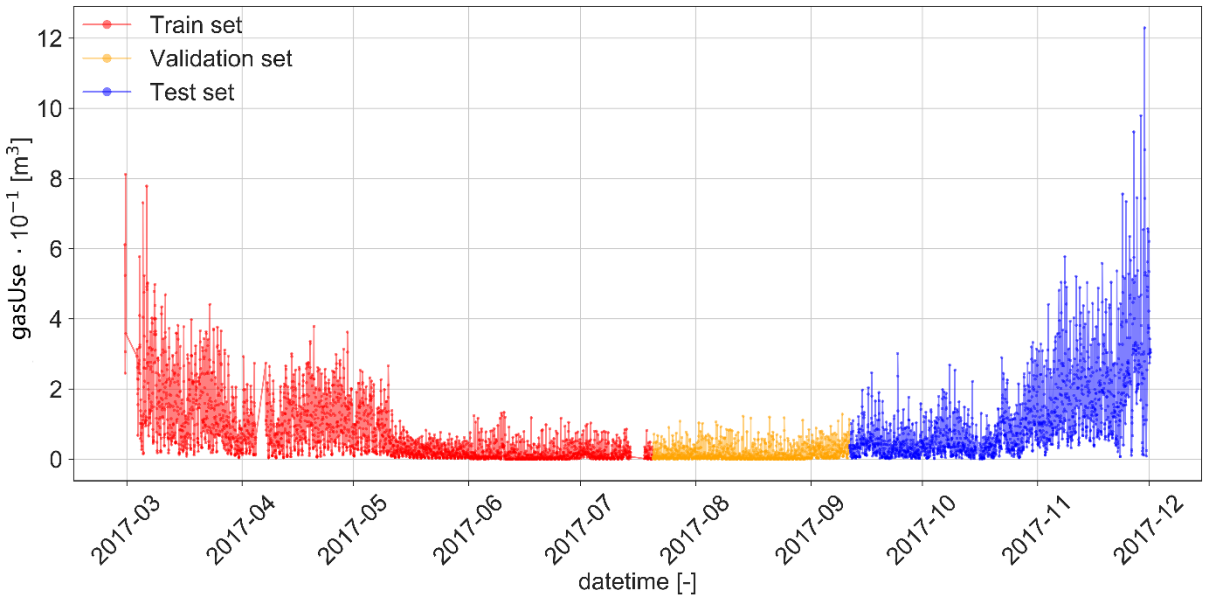

Fig. 2. A visualization of the train, validation and test dataset distribution on a daily resolution.

2.2 Train, test & validation dataset

All models used the same train, validation and test dataset. A visualisation of the distribution of the train, validation and test dataset is shown in *Figure 2*. The first 70 % of the dataset was used as the train set. The test set containing the remaining 30 % was used for cross-validation of each model. During training of the neural networks, the last 20 % of the train set was taken as the validation set.

2.3 Model evaluation

Neural networks use an optimizer to minimize the loss function. This can be interpreted as the least squares method for linear regression which minimizes the squared residuals. The loss function used for the neural network models in this paper was the Mean Squared Error (MSE) and is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2. \qquad (1)$$

Where $\hat{Y}_i$ is the i-th ground truth value, Y_i is the i-th predicted value and n is the total number of samples.

Two positive properties of the MSE is that in general it is relatively computationally inexpensive and is sensitive to outliers because the difference between Y_i and $\hat{Y}_i$ is squared. This sensitivity to outliers had a positive influence on the used dataset because the outliers in the gas consumption represent valid data points, e.g. they were not corrupt data due to malfunctioning of the measurement devices.

Two other available loss functions are the Mean Absolute Percentage (MAPE) and the Symmetric Mean Absolute Percentage Error (SMAPE). MSE was chosen over MAPE and SMAPE because they are less sensitive for outliers. Where MAPE is defined as:

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|\hat{Y}_i - Y_i|}{|Y_i|} \qquad (2)$$

, and SMAPE as:

$$\text{SMAPE} = \frac{100\%}{2n}\sum_{i=1}^{n}\frac{|Y_i - \hat{Y}_i|}{|\hat{Y}_i| + |\hat{Y}_i|}. \qquad (3)$$

Notice how the MSE is scale dependent whereas MAPE and SMAPE are in percentages. The MAPE and SMAPE were used as evaluation metrics, together with MSE to determine the performance of each model.

2.4 Optimizers and learning rate scheduler

Like stated in chapter 2.3, neural networks use an optimizer to minimize the loss function. In this research Adam [15] and Nadam [16] were used for the models. The used optimizer is specified per model in their respective subchapters. The main difference between Adam and Nadam is that Adam is essentially RMSprop with momentum whereas Nadam is Adam RMSprop with Nesterov momentum. This is simplified by imagining a curved plane in $\mathbb{R}^3$. When trying to get to the lowest point of this plane, Nadam will jump over hills more quickly than Adam would by default. Adam and Nadam were chosen instead of the commonly used Stochastic Gradient Descent (SGD) method, because they converge quicker than SGD does [17].

To improve the rate of convergence of the loss function, a cosine annealing learning rate scheduler with periodic restarts was applied to the optimizer. Together with improving the rate of convergence, this also gives the ability to find a lower and wider minimum [18].

Regularly, without the learning rate scheduler, the learning rate η_t is set to a fixed value for each batch within the total amount of epochs T_i. Whereas with the learning

rate scheduler, the learning rate is changed per batch within the i-th run as follows [19]:

$$\eta_t = \eta_{min}^i + \frac{1}{2}(\eta_{max}^i - \eta_{min}^i)(1 + \cos\left(\frac{T_{cur}}{T_i}\pi\right). \quad (4)$$

Where η_{max}^i and η_{min}^i are the ranges for the learning rate and T_{cur} are the number of epochs since the last restart.

2.5 Feature and batch normalization

The features X were standardized by removing the mean and scaling to unit variance using the 'StandardScaler' function from scikit-learn. This function scaled all feature samples by removing the mean and scaling to unit variance. This prevents one or more features dominating the others. The models also converge less quick and had a likelihood to have a lower accuracy when the features are not scaled. Each feature from the features train dataset X_{train} were scaled independently. The standard score Z of the feature sample was calculated as [20]:

$$Z = \frac{X_{train} - u}{s} \quad (5)$$

Where u is the mean value of the sample and s is the standard deviation from the sample. The standard score was stored and used to also transform the features X_{test} from the test dataset. The standard score was not being recalculated for the test set. This prevents having data leakage from the distribution of the feature samples from the test- to the train-set.

2.6 One hot encoding

The hour of the day, day of the week and current season values were extracted as features from the timestamp of each row in the dataset. Where hour of day ranged from 0 to 23, day of the week from 0 to 6 and season from 1 to 4. These features were one hot encoded. This transformed the data so each value of the feature had a separate column in the dataset, as for example this matrix:

$$\begin{bmatrix}0\\1\\2\end{bmatrix} \rightarrow \begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix} \quad (6)$$

Where the day of the week is represented as a number in the column vector. After transforming this column vector to the 3×3 matrix, each column represents a day of the week. With 1 representing that at given row is currently that day of the week. Where column one represented that row being on a Monday, column two being on a Tuesday and column three being on a Wednesday. Representing the features extracted from the timestamp in this way allows the models to assign different weights to for example 07:00 AM on a Monday and 09:00 AM on a Saturday.

2.7 Architecture evaluations

Hyperas was used to evaluate different architectures of each neural network model. A commonly used distribution of nodes and layers was set as the available parameter space. Hyperas was set to evaluate a fixed amount of possibilities from the parameter space. The amount of evaluations was chosen so the total evaluation time per model took 24 hours. During each evaluation Hyperas trained a different architecture for a specific number of epochs. In the end the best performing architecture was used for the models in this paper. The amount of evaluations per model was defined in *Table 2*.

2.8 Initial neural network architecture setup

All neural network models were programmed in Python using the Keras library with a TensorFlow backend and were trained on a NVIDIA GeForce 960m GPU. The weights of each layer were initialized by a truncated normal distribution. The bias from each dense layer was turned off because each layer was followed up by a batch normalization layer, apart from the output layer. This batch normalization layer normalized the weights like described in chapter 2.5 and was also applied after recurrent and convolutional layers. Finally, each layer apart from the output layer was followed up by the Leaky version of a Rectified Linear Unit (LeakyReLu) activation function, which is defined as [21]:

$$h^{(i)} = \max\left(\boldsymbol{w}^{(i)T}, 0\right) = \begin{cases} \boldsymbol{w}^{(i)T}x & \boldsymbol{w}^{(i)T}x > 0 \\ 0.01\boldsymbol{w}^{(i)T}x & \text{else} \end{cases} \quad (7)$$

Where $\boldsymbol{w}^{(i)}$ is the weight vector for the i-th hidden node and x is the node input.

2.9 MVLR

MVLR was the simplest model used in this research. Despite being a simple model, it has been often used for the energy forecasting and it is known to make relatively accurate predictions [22, 8]. The combination of the performance and simplicity made the evaluation metrics from MVLR the baseline to compare other models results with.

$$y = b_0 + b_1X_0 + b_2X_1 + b_3X_2 + \sum_{i=0}^{23} b_{4+i}X_{3+i} + \sum_{j=0}^{6} b_{28+j}X_{27+j} + \sum_{k=0}^{3} b_{34+k}X_{33+k} \quad (8)$$

where

y	gas use	[m^3]
b_0	offset	[-]
X_0	temperature	[°C]
X_1	Wind speed	[m/s]
X_2	rain intensity	[mm/h]
X_{3+i}	hour of the day	[-]
X_{27+j}	day of the week	[-]
X_{33+k}	season	[-]

2.10 DNN

The next model used was a feed-forward DNN, this is one of the basic types of neural networks due to all connections going in one direction without cycles or loops. The data from the input layers are passed on to the next layers of nodes (hidden layers) and based on the weights, offset and activation function they compute a value per node, which is passed on to the next layer until the output node is reached. The value of the output node will then output a prediction y_{pred}. One of the main benefits of feed-forward DNN is the ability to adapt to non-linear relationships, in contrast to MVLR.

Figure 3 contains a schematic of the used DNN architecture. The input shape of this model was a matrix of shape (features), containing the features of the current hour, to predict the next target value of shape (target). Where (features) when using five features would be (5), e.g. a five-dimensional row vector. Simplified, the weather information from the current hour was used to predict the gas consumption of the next hour. Nadam was used as the optimizer while training this model.

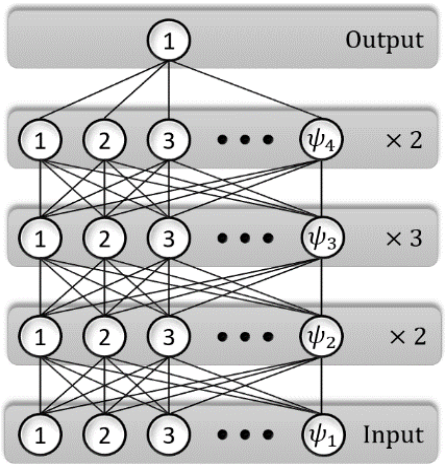

Fig. 3. Where $\psi_1, \psi_2, \psi_3, \psi_4$ represent the number of nodes of respective layer and are equal to 64, 256, 64, 1024, 8 respectively. The × 2 represents this layer configuration being repeated two times behind each other.

2.11 LSTM and GRU

LSTM and GRU are based on the Recurrent Neural Network (RNN), which are often used for natural language and text processing [23]. LSTM networks are different from RNNs by the ability to store historical information which has been processed in its internal memory units, which can be an advantage when using time series data [6]. Compared to LSTM, GRU networks use less parameters per node and thus can be interpreted as a simplified version of the LSTM model [24].

The input shape of this model was a matrix of shape (timesteps, features), containing the features of all the timesteps, to predict the next target value of shape (target). Timesteps can be interpreted as the model being able to look back a specific number of hours. For both the LSTM and GRU models this was set to 120 hours. Simplified and summarising, the weather information from the past 120 hours was used to predict the gas consumption of the next hour. The architectures used can be seen in *Figure 4*. Adam was used as the optimizer during training of the LSTM model, whereas Nadam was used for GRU.

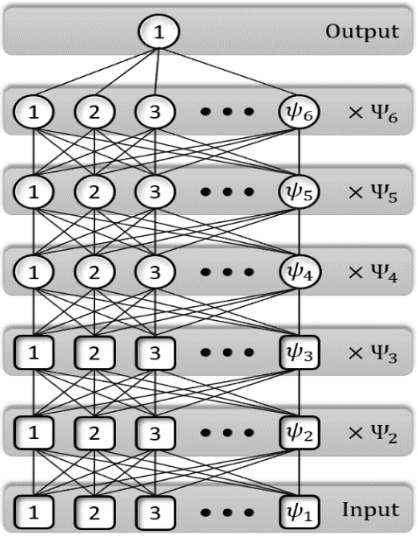

Figure 4. Where $\psi_1, \ldots, \psi_6$ represent the number of nodes of respective layer. For LSTM these are equal to 8, 0, 16, 128, 8, 16 and for GRU are equal to 16, 8, 4, 0, 8, 8 respectively. Each layer configuration being repeated Ψ_i times behind each other is represented by $\times \Psi_i$, where i is the layer number. For LSTM $\Psi_2, \ldots, \Psi_6$ are equal to 0, 1, 3, 2, 1 and for GRU are equal to 1, 1, 0, 4, 1 respectively.

2.12 CNN

A CNN is a type of deep neural network, most commonly used for image recognition. Partly due to the development of autonomous cars, image classification, facial recognition and more, CNNs are one of the most advanced neural networks currently being developed in computer science [25, 26, 27]. Compared to the previously discussed networks, a requirement to apply CNNs was the addition of a channel dimension to the feature matrix used in the RNN model. This changed the matrix from shape [timesteps, features] to [height, width, channel]. Where channel is the colour dimension, three for RGB images and one for grey-scaled images. Timesteps and features were interpreted as the height and width of the input image. The architecture used can be seen in *Figure 5*. The model was trained with the Nadam optimizer.

2.13 Time distributed CNN + RNN + DNN

The Time Distributed model consisted of a time distributed CNN layer, being followed up by an LSTM and finally a DNN. This model combined the power of all three models. An image with (timesteps, columns) of (120, 39) was fed into the input layer. This image was then reshaped to 24 smaller images of (5, 39, 1). Where 1 is the channel dimension required by the CNN.

Each smaller image was fed to the CNN and the flattened CNN output was saved in memory. These 5

flattened outputs made up the sequence that was fed into the RNN of shape (5, flattened CNN output). From here on the RNN and DNN were applied as explained in their respective subchapters. Nadam was used as the optimizer to train this model.

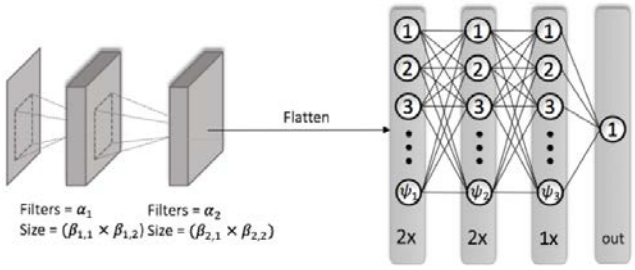

Figure 5. Where α_1 is equal to 5, α_2 is equal to 8, $(\beta_{1,1} \times \beta_{1,2})$ equals (8 x 4) and $(\beta_{2,1} \times \beta_{2,2})$ equals (10 x 8). The final output of the CNN is flatted and fed into a DNN where $\psi_1 \dots \psi_3$ equals 64, 128, 256 and $\Psi_1 \dots \Psi_3$ are equal to 2, 2, 1 respectively.

3 Results

Table 2 shows that on one hour resolution, the DNN model performed best with a 50.1 % MAPE, while LSTM had the lowest performance with a MAPE of 139 %. In comparison to MVLR, DNN outperformed MVLR because of its ability to adapt to non-linearities. DNN outperformed the other deep neural network models when comparing it to the other deep neural networks. Due to the 24 hour limitation on the architecture evaluations and number of epochs, DNN outperformed the other models thanks to its simplicity. A probable reason for LSTM and GRU performing worse than expected is the presence of NaNs in the dataset. When NaNs are removed, missing timestamps affect to the periodicity of data and therefore could have an influence on the accuracy of the model. Furthermore, this could explain why one hot encoded features such as hour of the day, day of the week and season leads to performance gains. This is explained by the LSTM and GRU models adapting to a pattern of a fixed periodicity between the time steps.

When comparing one day and one week resolutions, the results indicated that MVLR model outperformed the other models. MVLR had a MAPE of 20.2 % and 17.0 % , whereas LSTM was the lowest performing model with a MAPE of 99.7 % and 95.0 % on a one day and one week resolution respectively. When down sampling the data from one hour to lower resolutions, the cumulative error gets reduced because of the surface area of the errors getting smaller.

Figure 6 shows that the models tended to forecast systematically below the real values. Furthermore, during summer MVLR and DNN outperformed LSTM, GRU, CNN and Time Distributed. During winter, the difference between the real and the forecasted values became larger.

Table 2. The values of cross-validation evaluation metrics, amount of architecture evaluations and the amount of epochs done per model.

Model [-]	Resolution	MSE [-]	MAPE [%]	SMAPE [%]	Architecture evaluations [-]	Time per epoch [s]	Epochs [-]
MVLR	Hour	0.62	78.3	193	n.a.	n.a.	n.a.
	Day	99.0	20.2	920			
	Week	$2.44 \cdot 10^3$	17.0	7.80			
DNN	Hour	0.67	50.1	16.6	$1.00 \cdot 10^3$	$4.00 \cdot 10^{-6}$	$3.50 \cdot 10^4$
	Day	104	25.1	10.5			
	Week	$2.96 \cdot 10^3$	20.1	8.70			
LSTM	Hour	1.00	139	33.9	50.0	$4.62 \cdot 10^{-3}$	$4.00 \cdot 10^3$
	Day	206	99.7	30.1			
	Week	$7.06 \cdot 10^3$	95.0	31.1			
GRU	Hour	1.19	78.6	30.5	100	0.11	$4.00 \cdot 10^3$
	Day	264	59.8	19.4			
	Week	$9.38 \cdot 10^3$	45.3	16.9			
CNN	Hour	0.84	84.3	28.3	50.0	0.76	$8.00 \cdot 10^3$
	Day	115	33.3	13.5			
	Week	$3.51 \cdot 10^3$	32.3	13.6			
Time Dist.	Hour	0.91	74.0	26.8	100	$2.88 \cdot 10^{-3}$	$4.00 \cdot 10^3$
	Day	184	42.7	16.4			
	Week	$5.93 \cdot 10^3$	41.5	16.3			

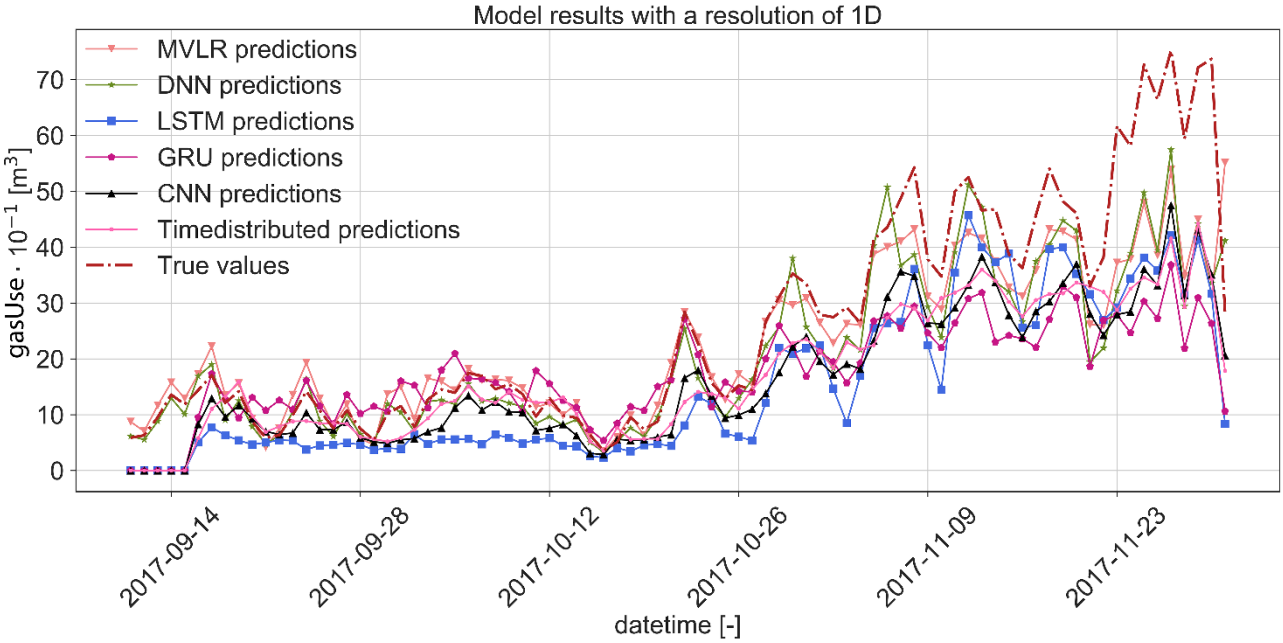

Fig. 6. The forecasted gasUse consumption of the different models on a daily resolution.

4 Conclusion

This paper compares several machine learning algorithms to forecast the mean gas consumption of 52 dwellings, representing a block of dwellings on the aggregated level. Forecasts were done with an hourly resolution by using the wind speed, rain intensity, temperature, season, hour of day and day of the week as features. The feature selection was made by using as few features as possible, with the objective of keeping the computation power required as low as possible. The choice of models was based on previous research studies which were mainly focused on forecasting the gas and electricity consumption in commercial and office buildings. Furthermore, three types of deep learning models were combined into a single model called Time Distributed. Time Distributed combined the potential of CNNs and RNNs, with the goal of getting a better performance regarding the outcome of the results. More specifically, this was a time distributed CNN followed up by a LSTM and DNN.

To validate the applicability of each model, the models were compared on performance and computational time required. The gas consumption data of the mean of 52 dwellings was split into a training and test dataset of 70 % and 30 %, respectively. Predictions were cross validated on the test set with an hourly resolution. To evaluate the performance on multiple resolutions, the hourly predictions were down sampled to one day and one week resolution by summation.

As seen in *Table 2*, DNN performed best on an hourly resolution when looking at the MAPE. On a daily and weekly resolution, MVLR outperformed the other models. In all resolutions, LSTM had the lowest performance.

Further studies should focus on exploring the possibilities of getting more accurate results and applying the models on individual dwellings. One way the evaluation metrics could be improved is by using more data, e.g. a sampling period of full-year or more. This is substantiated on the variance between the validation and train loss. Improving the amount of training data could help with the recognition of human patterns and dependency on outside weather conditions. Alongside this, more features like the electricity consumption can be used to improve the accuracy of the deep learning models. However, this results in an increase in computational power needed, which can be a drawback in certain situations, e.g. when the hardware used has insufficient computational power. In the case of this research, as less features as possible were used. In addition to the previously stated recommendations, increasing the amount of architecture evaluations, along with the number of epochs of the final model architecture, can lead to a better performance.

The models programmed in Python can be found at GitHub: https://github.com/deKeijzer/KB-74-OPSCHALER

References

1 Rijksoverheid, *Energieagenda: naar een CO_2-arme energievoorziening*, (2016)

2 C. Xu, H. Chen, J. Wang, Y. Guo, Y. Yuan, Building and Environment, **148**, 128 (2019)

3 A. Hernandez Neto, F. A. Sanzovo Fiorelli, Energy and Buildings, **40-12**, 2169 (2008)

4 J. Woo, A. E. Fenner, A. Asutosh, D. Kim, M. Razkenari, C.J. Kibert, *Proceedings IISE Annual Conference* (2018)

5 S. Bouktif, A. Fiaz, A. Ouni, M.A. Serhani, Energies, **11**, 1636 (2018)

6 W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu,Y. Zhang, Member, IEEE Transactions on Smart Grid, **10-1** (2017)

7 A. Özmen, Y. Yılmaz, G.W. Weber, Energy Economics, **70-C**, 357, (2018)

8 C. Jurado López, *Data-driven Predictive Control for Heating Demand in Buildings*, (TU Delft, 2017)

9 N. G. Paterakis, E. Mocanu, M. Gibescu, B. Stappers, W. van Alst, *Proceedings ISGT-Europe*, (2017)

10 K. Sabo, R. Scitovski, I. Vazler, M. Zekić-Sušac,Energy Conversion and Management, **52-3**, 1721 (2011)

11 KNMI, *Daggegevens KNMI Rotterdam*, (2018) http://projects.knmi.nl/klimatologie/daggegevens/selectie.cgi.

12 M. Brabec, O. Konár, E. Pelikán, M. Malý, International Journal of Forecasting, **24-4**, 659 (2008)

13 J. Szoplik, Energy, **85**, 208(2015)

14 D. M. Allen, Technometrics, **13-3**, 469 (1971)

15 D. P. Kingma, J.L. Ba, *Proceedings ICLR* (2015)

16 T. Dozat, *Proceedings ICLR* (2016)

17 S. Ruder, arXiv:1609.04747 (2017)

18 B. Y. Hsueh, W. Li, I. Wu, arXiv:1806.01593v2 (2018)

19 I. Loshchilov, F. Hutter, arXiv:1608.03983 (2017)

20 T.O. Adeyemi, Research Journal of Mathematics and Statistics, **3-3**, 91 (2011)

21 A.L. Maas, A.Y. Hannun, A.Y. Ng, *Proceedings ICML*, 30-1, 3 (2013)

22 V. Bianco, O. Manca, S. Nardini, Energy, **34-9**, 1413 (2009)

23 M. Morchid, Neurocomputing, **314**, 48 (2018)

24 J. Chung, C. Gulcehre, K. Cho, Y. Bengio, arXiv:1412.3555 (2014)

25 B.B.Traore, B. Kamsu-Foguem, F. Tangara, Ecological Informatics, **48**, 257 (2018)

26 L. Caltagirone, M. Bellone, L. Svensson, M. Wahde, Robotics and Autonomous Systems, **111**, 125 (2019)

27 X. Zhu, M. Zhu, H. Ren, Cognitive Systems Research, 52, 223 (2018)