Domestic hot water forecasting for individual housing with deep learning

Paul Compagnon, Aurore Lomet, Marina Reyboz, Martial Mermillod

Domestic Hot Water Forecasting for Individual Housing with Deep Learning

Paul Compagnon[1], Aurore Lomet[2], Marina Reyboz[1], and Martial Mermillod[3]

[1] Univ. Grenoble Alpes, CEA, List, F-38000 Grenoble, France
[2] Université Paris-Saclay, CEA, List, Palaiseau, France,
{paul.compagnon, aurore.lomet, marina.reyboz}@cea.fr
[3] LPNC Univ. Grenoble Alpes & CNRS F-38000 Grenoble, France
martial.mermillod@univ-grenoble-alpes.fr

Abstract. The energy sharing used to heat water represents around 15% in European houses. To improve energy efficiency, smart heating systems could benefit from accurate domestic hot water consumption forecasting in order to adapt their heating profile. However, forecasting the hot water consumption for a single accommodation can be difficult since the data are generally highly non smooth and present large variations from day to day. We propose to tackle this issue with three deep learning approaches, Recurrent Neural Networks, 1-Dimensional Convolutional Neural Networks and Multi-Head Attention to perform one day ahead prediction of hot water consumption for an individual residence. Moreover, similarly as in the transformer architecture, we experiment enriching the last two approaches with various forms of position encoding to include the order of the sequence in the data. The experimented models achieved satisfying performances in term of MSE on an individual residence dataset, showing that this approach is promising to conceive building energy management systems based on deep forecasting models.

Keywords: Time Series Forecasting · Domestic Hot Water · Deep Learning · Convolutional Neural Networks · Multi-Head Attention

1 Introduction

In 2020, the average energy sharing used in an European house to heat water was around 15% [1]. Most of the time, water is heated before being used by a water heating system which priority is to insure user comfort by providing enough hot water at any time. The system can heat too much water in advance leading to energy waste [2]. This is even more true if the system is alimented by a heat pump which performance depends on a large number of external factors such as the weather. These considerations highlight the need for accurate forecasting models of hot water consumption, that would allow to heat up just the right amount of water at the best moment improving the building energy efficiency.

However, Domestic Hot Water (DHW) forecast for individual residences can be difficult due to the consumption being very sporadic and highly variable from

days to days [3]. To solve this problem, we explore the use of deep learning models. Indeed, we propose to compare three deep learning approaches adapted to sequence processing to forecast individual hot water consumption: Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN) and Multi-Head Attention (MHA). Unlike RNN, CNN and MHA do not explicitly take into account the temporal dependencies in the data (they both consider a sequence as an unordered matrix). One way to correct this is to use position encoding as in the whole transformer architecture [4]. A position encoding will be added to a representation in order to drag it in a specific direction in the feature space, the same for elements at the same position in different sequences. We compare CNN and MHA equipped with 3 different position encoding: fixed as in [4], learned as in [5] and produced by a RNN. In a similar way as in [6], we also propose a version where the days and hours are embedded the same way instead of being one hot encoded. Another aspect of the proposed models is that they are constituted of a few layers unlike recent approaches proposed for time series forecasting [7] and this has two upsides. Firstly, they can be trained on a small number of samples and therefore be put into production quickly with data coming from a single accommodation. Secondly, they can also more easily be embedded inside a heating system with small computation power preserving the privacy of data. We test those architectures on a dataset recorded on a real system installed in an individual accommodation.

The remaining of the paper is organized as follows. Section 2 overviews the state of the art of water consumption forecasting with a focus on deep learning approaches. Section 3 describes the components of the employed deep architectures and specifically the different position encodings experimented. Section 4 presents the individual consumption dataset we used, the experimental setup and the achieved results. Finally, Section 5 concludes this paper and presents some perspectives of this work.

2 Related work

In this section, we give an overview of the different approaches used in the literature to forecast water consumption. First, a large number of papers seek to forecast water consumption at a large scale, for neighborhood, cities or even entire regions [8–10]. Candelieri et al. [11] proposed to use several Support Vector Machine (SVM) models to forecast the water demand in a distribution network with a 24h delay. Each model takes as input the first six hours of the day and forecast one different time slot of one hour of consumption. Moreover, the authors employ time series clustering to determine several consumption profiles and train different SVM models accordingly. Mu et al. [12] proposed to use a Long-Short term memory network (LSTM) [13] to forecast the water demand of a city at a one-hour or 24h delay. They found that their approach got better results than AutoRegressive Integrated Moving Average modeling (ARIMA), which combines an autoregressive part consisting of a weighted sum of the previous times steps and a moving average part to model the error, SVM and random

forests models. This was particularly true when performing high resolution predictions (e.g. a prediction every few minutes) on data with abrupt changes. They also highlighted the upside of LSTM to allow to output a sequence of predictions. Finally, in a recent paper, Karamaziotis et al. [14] compared ARIMA models, Optimized Theta (a form of exponential smoothing model), Ensemble methods and neural networks to forecast the water consumption of in European capital cities with a forecast horizon of several months. They concluded, based on various metrics, that ARIMA models seem the best approach in several prediction scenarii. From these papers, we can conclude that various machine learning models can be considered to forecast the water consumption with ARIMA models seemingly achieving the best performances.

However, individual residences present an additional difficulty since the consumption is generally non smooth and irregular compared to apartment building. Moreover, large variations across days could happen more frequently (for examples, all the inhabitants of the house leave for several days). This assumption was observed experimentally by Maltais et al. [3] who proposed to use neural networks to predict the DHW consumption of residential buildings with different sizes. They observed that the prediction performances were better when increasing size of the systems. They attributed this difference to the smallest ones presenting too much variations in their consumption and they were consequently the worst predicted. Overall, the problem of small or individual residences has been less tackled by the literature and the employed approaches generally fall into two categories. The first one is once again ARMA modeling [15]. This approach have been used by Lomet et al. [16] to perform DHW load prediction for a single family accommodation. After studying the data, notably the autocorrelograms, they decided to take into account the weekly periodicity (the seasonality of the time series) and the consumption of the two previous days to build their model. Gelažanskas et al. [17] also analyzed the effect of seasonality to improve regression performances. They likewise found that a SARIMA (Seasonal ARIMA) model taking into account daily and weekly consumption patterns performed better. The second type of approaches is deep learning. Barteczko-Hibbert et al. [18] experimented using Multi-Layer Perceptron (MLP) to predict the temperature of the drawn hot water to optimize the heating system. They trained the network on a particular heating profile and tested on another one and found that the model generalized better if it was trained on two different heating profiles instead of one. In another publication, Gelažanskas et al. [19] tackled the issue of hot water forecasting with an auto regressive neural network taking as input data from the previous hours, data from seven days before and external variables, in an autoregressive manner. They showed the importance of external variables to improve predictions. Regarding the type of approaches, the last two papers did not employed specific deep learning models for sequences such as LSTM mentioned above. Gelažanskas et al. took into account the seasonality of the data but with a MLP architecture.

If ARIMA models have proven their efficiency to model hot water consumption, more results seem needed with neural networks to conclude. However, neu-

ral networks have also been found to perform better than ARIMA in various conditions which are realized for individual housing profiles: when the needed resolution of the prediction is high and when the data present gaps [12] or high variability [20]. Moreover, ARIMA models are linear unlike neural networks than can model non-linear relationships which has proven to be useful to forecast hot water consumption [21]. As a consequence, ARIMA models seem less adapted to this task. Additionally, pure deep learning models present several upsides compared to classical machine learning or hybrid models: firstly they do not require feature engineering, secondly they can be trained end to end with a single objective and allow to simply combine different types of layers and differentiable processings, finally they have a very low inference time making them suited for real time systems. Based on these observations, in this paper, we propose to employ deep learning architectures to build lightweight forecast models for individual homes that can be easily embedded inside heat pump control systems. We focus on architectures specifically adapted for time series processing and we will now present them.

3 Proposed Architectures

In this section, we describe the different neural network components that will constitute the tested deep architectures for DHW forecasting.

3.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are the primary component to deal with sequences and time series. A simple RNN's equation is as follows:

$$h_t = f(W_i x_t + b_i + W_h h_{t-1} + b_h), \tag{1}$$

where $x_t \in \mathrm{R}^n$ defines a sequential input, f is a non linear activation function, most of the time hyperbolic tangent or sigmoid and W_i, W_h, b_i and b_h are parameters of the models to be learned. The output $h \in \mathrm{R}^m$ defines a sequence with the same length as the input sequence and is reinjected each time with along the new input. This way, the RNN is able to learn temporal correlations inside the sequence.

However, vanilla RNN as the one described above actually do not work when sequence become too long. Due to a phenomenon called vanishing gradient, they are in fact not able to learn long-term dependencies in the sequence [22]. To work around this issue, gated RNN such as LSTM [13] and later Gated Recurrent Units (GRU) [23] have been developed. These neural networks emulate a memory thanks to a system of gates that let pass new information and forget old one.

3.2 One-dimensional Convolutional Neural Networks

One-dimensional Convolutional Neural Networks (1DCNN) are a variant of CNN that has been employed to perform signal processing tasks and time series forecasting [24–26]. They are actually close to the old Time-Delay neural networks

[27]. One great upside of 1DCNN compared to their 2D counterparts is their lower time complexity: they are therefore well suited to build compact embedded architectures with few layers [28].

1DCNN uses convolutional filters that move only in the time direction on all the features at the same time, with a kernel size determining the number of convoluted timesteps. This way, they can detect local temporal correlations in the data. The locality can be extended by adding more layers. Similarly as with 2D CNN, after one or several convolution layers, a pooling layer is called. In our experiments, we used average pooling instead of the classical maximum pooling as we found it achieved better results.

3.3 Multi-Head Attention

Multi-Head attention (MHA) [4] is the central component of the transformer neural network architecture that is primarily used for natural language processing tasks. Transformer architectures can also be used for time series forecasting [29, 7], by they are very deep and would require much more data to be properly trained as well as a lot of computation power. Therefore, in this paper, we will not make use of a whole transformer model but only some of its components to build an *ad hoc* architecture: position encodings (see below) and thus Multi-Head Attention.

Attention is a differentiable mechanism that allows to make a query on a discrete set to get a result as a weighted sum of the elements of the set. Formally, consider three matrices Q, K and V , respectively the query, the keys and the values, attention is computed the following way:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{m}}\right)V, \tag{2}$$

where m is the number of features of K used here to scale the softmax. In practice, self-attention is used meaning that Q, K and V are the same matrix, in our case, a sequence. To make Multi-Head Attention, for the desired number of heads, Q, K and V are each projected in a different (learnable) space before Equation 2 is applied. Then, the results are concatenated and projected again.

The output of the MHA being a sequence, we need a mechanism to obtain a vector to be then projected into a single value. We explored three way to perform this operation: with a RNN, using a average pooling similarly as with CNN and simply taking the last value of the sequence. After several tests, we finally decided to use an average pooling as with 1DCNN.

3.4 Position encodings

Along MHA, position encoding (or embeddings) in another component of the transformer architecture [4]. It is used to introduce information about the order in the data since the transformer will see the sequence as a whole and not timesteps by timesteps as for a RNN. A similar idea was introduced in [5] for

CNN with learned embedding. For both architectures, we start by projecting the input a time t inside a first representation space for dimensionality reduction before adding the position encodings associated with time t:

$$r_t = Wx_t + e_t, \tag{3}$$

where $r_t, e_t \in \mathrm{R}^m$ and W are learnable parameters. We detailed below the two types of positional encoding previously mentioned along with two others.

Fixed position embedding This approach is used in [4] and relies on fixed predefined codes to be added to the representations. Those codes are computed the following way for timestep t and feature index i:

$$e_t(2i) = \sin\frac{t}{10000^{2i/m}}, \tag{4}$$

$$e_t(2i+1) = \cos\frac{t}{10000^{2i/m}}. \tag{5}$$

With this position embedding, each position is uniquely represented by trigonometric functions with different frequencies.

Learned position embedding This approach was proposed by Gehring et al. [5] for CNN. Here, each timestep t is embedded in a feature space of the same dimensionality as:

$$e_t = W_t t, \tag{6}$$

where W_t are learnable parameters.

Generated by a RNN Recurrent networks take into account the temporal dependencies of the data by design since the timesteps of the sequence are input one after the other. We explore the possibility of including the temporal information of the data inside the MHA and CNN architectures by directly adding the output sequence of a RNN to the input embedding:

$$e_t = \mathrm{RNN}(x)_t. \tag{7}$$

In this paper, the RNN is implemented by a single layer of GRU.

Enriched learned position embedding In a similar way as [6], one can add multiple different embeddings corresponding to different categories. Along with position, we propose to use learned embedding for the day of the week and the hour of the day instead of one-hot encoded features:

$$e_t = W_t t + W_d d + W_h h, \tag{8}$$

$$r_t = \frac{Wx_t + e_t}{||Wx_t + e_t||_2}, \tag{9}$$

where W_d and W_h are learnable parameters and d and h are respectively the day of week and the hour of the day. The representation r_t is normalized after the sum to preserve its scale. We shall compare experimentally those four approaches in the next section.

4 Experiments

4.1 Dataset

This dataset was recorded by the Fraunhofer Institute for Solar Energy Systems ISE on a sensor equipped heating system of an individual residence. The system contains an heat pump to supply energy and a water tank to heat up water. In addition to drinking water, the hot water is used in the floor heating of the house. The dataset contains several months of recorded data sampled at one minute. Various measurement types for each component of the heating system are available: supply an return temperatures of the water, water flow, power and energy. The system is alimented by a heat pump for which we also have the energy and power measurements. Finally, ambient temperature has also been recorded.

The raw dataset contains more than 100 features. After removing the columns that were missing to much values, there are 93 features left. We also add 24 features for the hour of the day one-hot encoded, 7 features for the day of the week one-hot encoded, 1 feature indicating if it is the weekend and another one for the holidays for a total of 126 features. A simple interpolation was realized to complete the remaining few holes in the dataset.

4.2 Experimental Setup

We used sequences of length 72, sampled at one value every twenty minutes (equivalent to a day of data). We tested different downsampling values and found that this value was sufficient. The input sequence used to predict the DHW consumption at time t is constituted of the sequence from time t-48h to t-24h concatenated on the feature dimension with the sequence from time t-7 days to t-6 days. This allows to take into account potential weekly periodicity in the data, as most approaches from the state of the art. The water flow to predict is accumulated during a day to smooth the values to output since the hot water consumption in the dataset is very sporadic, as expected for an individual house. We give here more details about the training process and the hyperparameters. The experiments were conducted with 2 months of data from which 3 disjoint sets were created: 1 month for the training set, 10 days for validation and 20 days for test. One value every 30 minutes was predicted. We used a starting learning rate of 0.001 divided by 10 every 25 epochs without improvement. The training is stopped once the loss on the validation set has not decreased during 500 epochs. The batch size is 128. Finally, the features were scaled between 0 and 1 to ease the training for the neural networks.

The position encoding approaches are compared with constant architectures. Those architectures were found after hyperparameter search. The dimensions for each architecture are reported in Table 1, in addition the kernel size used for the CNN is 3. The dimension of the position encoding thus corresponds to the dense layer sizes.

Architecture type	Dense layer size	Specific layer size/filters	Output
GRU/LSTM	80	20	1
1DCNN	100	[8, 16]	1
MHA	64	32	1

Table 1: Architecture dimension summary

The models are trained with the Mean Squared Error loss (MSE) and regularized with weight decay (factor 10^{-4} and dropout (probability 0.5). We trained each version of each model 10 times and saved the best trained model regarding validation to perform a test.

We compare the results of the deep learning models with an ARIMA model. We generally followed the results of the study by Lomet et al. [16] to select the coefficients but adapted it to our data since the sampling is notably different by using the validation set. The autoregressive and moving average orders were set to 336 (one week of data sampled at 30 minutes to have the same rate of prediction) with all coefficients set to zero except for $\{p_{46}, ..., p_{50}, p_{332}, ..., p_{336}, q_{47}, q_{48}, q_{49}\}$. Similarly as for deep learning models and [16], the prediction only depends on data from 24h and one week before the prediction. We restrained the coefficients in order to avoid overfitting, especially for the moving average ones. No differentiation was performed following a study using the augmented Dickey–Fuller test and the seasonal orders were as well set to zero. The remaining exogenous variables were embedded through a principal components analysis in order to reduce their dimension to 6. The results are obtained by sliding along the validation set and making a one-step ahead prediction each time.

4.3 Results

We present in this section our results on the ISE dataset. The standard regression scores MSE and Mean Absolute Error (MAE) are reported. It is sometimes advised in the literature [30] to report Mean Average Percentage Error. However, this score explodes when the value to predict is zero which is often the case here, during the night for example. That is why we reported MAE instead.

The validation results for the version without position encodings are presented on Table 2. The best average MSE and MAE are achieved by GRU with 0.0033 and 0.0329 of MSE and MAE respectively. However, the best models overall where produced with the LSTM with 0.0028 and 0.0288 of MSE and MAE respectively. The 1DCNN and MHA architectures achieved results around ten

Algorithms	Position encoding	average MSE	average MAE	best MSE	best MAE
GRU	None	**0.0033** ± 0.0002	**0.0329** ± 0.0015	0.0030	0.0310
LSTM	None	0.0037 ± 0.0006	0.0337 ± 0.0026	**0.0028**	**0.0288**
1DCNN	None	0.0336 ± 0.0074	0.1413 ± 0.0169	0.0221	0.1145
MHA	None	0.0353 ± 0.0085	0.1323 ± 0.0295	0.0245	0.1051
ARIMA	None	-	-	0.1044	0.2741

Table 2: Validation results on ISE dataset for models without position encoding, average of 10 runs and best values

Algorithms	Position encoding	average MSE	average MAE	best MSE	best MAE
1DCNN	Fixed	**0.0084** ± 0.0023	**0.0632** ± 0.0120	**0.0052**	**0.0463**
1DCNN	Learned	0.0185 ± 0.0029	0.1002 ± 0.0114	0.0104	0.0720
1DCNN	RNN	0.0272 ± 0.0067	0.1213 ± 0.0183	0.0175	0.0938
1DCNN	Enriched	0.0277 ± 0.0087	0.1269 ± 0.0201	0.0175	0.0996

Table 3: Validation results on ISE dataset for CNN with position encoding, average of 10 runs and best values

Algorithms	Position encoding	average MSE	average MAE	best MSE	best MAE
MHA	Fixed	**0.0067** ± 0.0015	**0.0533** ± 0.0072	**0.0045**	0.0420
MHA	Learned	0.0123± 0.0009	0.0723 ± 0.0024	0.0111	0.0697
MHA	RNN	0.0111 ± 0.0041	0.0692 ± 0.0148	0.0060	0.0490
MHA	Enriched	0.0132 ± 0.0074	0.0708 ± 0.0255	0.0047	**0.0407**

Table 4: Validation results on ISE dataset for MHA with position encoding, average of 10 runs and best values

Algorithms	PE	MSE	MAE
GRU	None	0.0032	0.0352
LSTM	None	0.0046	0.0409
1DCNN	Fixed	0.0074	0.0917
MHA	Fixed	0.0030	0.0328

Table 5: Test results on ISE dataset

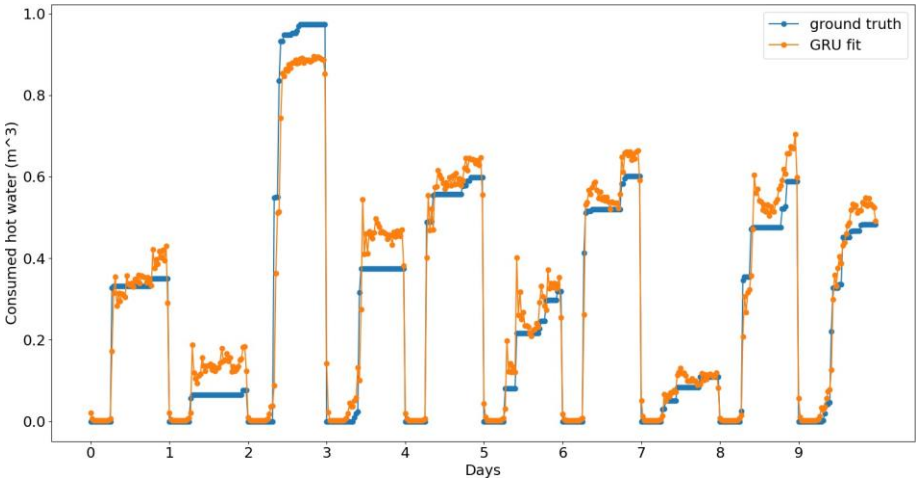
times inferior to the RNN architectures showing that information about the temporal order of the vectors seems necessary for this task on this dataset. ARIMA also seem to achieve lower results on this dataset. This observation is further confirmed in Table 3 and Table 4 which respectively present the validation results for 1DCNN and MHA with position encodings. Indeed, both approaches achieved better results with all forms of position encodings than without. However, the performance improvement is clearly better for MHA than for the 1DCNN: the best MSE scores achieved, 0.0045, is the closest we could achieve from the RNNs performances. In both cases also, fixed position encoding led to the best results with the enriched version being close second for MHA for the best iterations. We suppose that the low quantity of data favors the fixed version since nothing more needs to be learn. Finally, for MHA and 1DCNN, we remark than standard deviations are higher than for GRU and LSTM. We make the assumption that for these architectures, on this dataset, the initialization is a crucial factor to achieve the best performances.

We now present test results on Table 5, obtained each time from the best trained model from the validation phase. We observe that GRU and MHA with fixed position encoding achieved the best results with the later being slightly better with 0.0030 and 0.0328 of MSE and MAE respectively. The model based on LSTM seems to generalize a bit less well than GRU, with an MSE of 0.0046 even though it achieved a better validation MSE than GRU. Figure 1 shows the comparison between the obtained fit for both models and the groundtruth for the 10 first days of the test set. We see that the consumption of days 1 and 3 is overestimated by both models whereas it is underestimated for day 2, though less by MHA than GRU. An interesting day is day 5 whose gaps are correctly predicted by MHA unlike GRU, the inverse is observed for day 7. Those gaps, that constitute the major difficulties of individual housing datasets, are therefore possible to predict with neural networks, though not always easily.

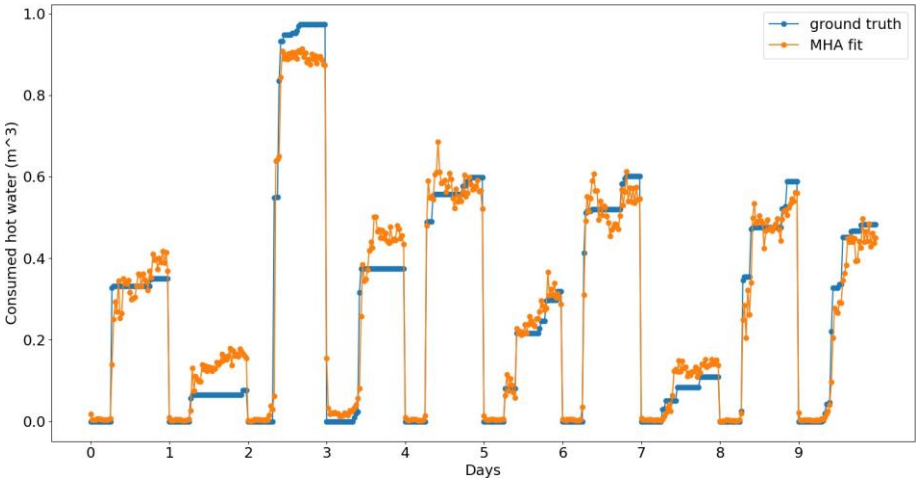
The 1DCNN architecture achieves lower results as expected from the validation. We make the assumption that the local temporal correlations in the sequence bring too less information to the model due notably to the presence of large plateau observed in the groundtruth curves (see blue curves on Figure 1). Extending the range of possible correlations would require to increase the depth of the network and thus to train it with more data, threatening the use of the model in an embedded environment.

5 Conclusion and Perspectives

We proposed in this paper to use deep learning to tackle the issue of hot water consumption for individual housing. Neural networks are especially recommended when the data are variable and present abrupt variations [20, 12] which is the case when dealing with individual house consumption profiles. We compared three deep learning approaches adapted to time series forecasting: RNN, 1DCNN and MHA, the last two being equipped with various form of position encodings [4] to improve their sequence processing ability. One objective was



(a) GRU



(b) MHA

Fig. 1: Comparison between the fits and the true water consumption

to conceive lightweight models able to be embedded in an energy management system to preserve the privacy of the data and to avoid wasting energy with computationally greedy models. We experimented the three approaches on a dataset recorded in a real individual housing. We achieved the best results with GRU and MHA with fixed position encoding architectures and demonstrated the ability of the models to correctly predict in some cases the gaps in the consumption on this dataset.

In the future, we plan to test those architectures on more datasets, notably on datasets containing less features since the inclusion of multiple sensors in a heating system also costs energy resource. We plan as well to integrate those models in real energy management system to see if the prediction they produce can effectively reduce the energy consumption.

Acknowledgment

This paper has been written as part of the Artificial Intelligence for Heat Pumps project financed by the French national Agency for Research (ANR). This work has been partially supported by MIAI@Grenoble Alpes, (ANR-19-P3IA-0003). The authors would like to thank Fraunhofer ISE and especially Lilli Frison and Simon Gölzhaeuser for providing their dataset. They also would like to thank Olivier Antoni, Romain Bailly, Yanis Basso-Bert and Marielle Malfante for constructive discussions on Transformers.

References

1. Eurostat. *Energy, transport and environment statistics*. European Commission, 2020.
2. Bo Lin, Shuhui Li, and Yang Xiao. Optimal and learning-based demand response mechanism for electric water heater system. *Energies*, 10(11):1722, 2017.
3. Louis-Gabriel Maltais and Louis Gosselin. Predictability analysis of domestic hot water consumption with neural networks: From single units to large residential buildings. *Energy*, 229:120658, 2021.
4. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
5. Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252. PMLR, 2017.
6. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
7. Bryan Lim, Sercan O Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *arXiv preprint arXiv:1912.09363*, 2019.
8. E Pacchin, F Gagliardi, S Alvisi, and M Franchini. A comparison of short-term water demand forecasting models. *Water resources management*, 33(4):1481–1497, 2019.

9. Shan Wu, Hongquan Han, Benwei Hou, and Kegong Diao. Hybrid model for short-term water demand forecasting based on error correction using chaotic time series. *Water*, 12(6):1683, 2020.
10. Matheus Henrique Dal Molin Ribeiro, Ramon Gomes Da Silva, José Henrique Kleinübing Larcher, José Donizetti De Lima, Viviana Cocco Mariani, and Leandro Dos Santos Coelho. Seasonal-trend and multiobjective ensemble learning model for water consumption forecasting. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.
11. Antonio Candelieri, Ilaria Giordani, Francesco Archetti, Konstantin Barkalov, Iosif Meyerov, Alexey Polovinkin, Alexander Sysoyev, and Nikolai Zolotykh. Tuning hyperparameters of a svm-based water demand forecasting system through parallel global optimization. *Computers & Operations Research*, 106:202–209, 2019.
12. Li Mu, Feifei Zheng, Ruoling Tao, Qingzhou Zhang, and Zoran Kapelan. Hourly and daily urban water demand predictions using a long short-term memory based model. *Journal of Water Resources Planning and Management*, 146(9):05020017, 2020.
13. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
14. Panagiotis I Karamaziotis, Achilleas Raptis, Konstantinos Nikolopoulos, Konstantia Litsiou, and Vassilis Assimakopoulos. An empirical investigation of water consumption forecasting methods. *International Journal of Forecasting*, 36(2):588–606, 2020.
15. George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
16. Aurore Lomet, Frédéric Suard, and David Chèze. Statistical modeling for real domestic hot water consumption forecasting. *Energy Procedia*, 70:379–387, 2015.
17. Linas Gelažanskas and Kelum AA Gamage. Forecasting hot water consumption in residential houses. *Energies*, 8(11):12702–12717, 2015.
18. Christian Barteczko-Hibbert, Mark Gillott, and Graham Kendall. An artificial neural network for predicting domestic hot water characteristics. *International Journal of Low-Carbon Technologies*, 4(2):112–119, 2009.
19. Linas Gelažanskas and Kelum AA Gamage. Forecasting hot water consumption in dwellings using artificial neural networks. In *2015 IEEE 5th International Conference on Power Engineering, Energy and Electrical Drives (POWERENG)*, pages 410–415. IEEE, 2015.
20. Guancheng Guo, Shuming Liu, Yipeng Wu, Junyu Li, Ren Zhou, and Xiaoyun Zhu. Short-term water demand forecast based on deep learning method. *Journal of Water Resources Planning and Management*, 144(12):04018076, 2018.
21. Mathias Braun, Thomas Bernard, Olivier Piller, and Fereshte Sedehizade. 24-hours demand forecasting based on sarima and support vector machines. *Procedia Engineering*, 89:926–933, 2014.
22. Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*,5(2):157–166, 1994.
23. Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
24. Shengdong Du, Tianrui Li, Yan Yang, and Shi-Jinn Horng. Deep air quality forecasting using hybrid deep learning framework. *IEEE Transactions on Knowledge and Data Engineering*, 33(6):2412–2424, 2019.

25. Christian Lang, Florian Steinborn, Oliver Steffens, and Elmar W Lang. Applying a 1d-cnn network to electricity load forecasting. In *International Conference on Time Series and Forecasting*, pages 205–218. Springer, 2019.
26. Romain Bailly, Marielle Malfante, Cédric Allier, Lamya Ghenim, and Jérôme Mars. Deep anomaly detection using self-supervised learning: application to time series of cellular data. In *3rd International Conference on Advances in Signal Processing and Artificial Intelligence*, 2021.
27. Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.
28. Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021.
29. Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems*, 32, 2019.
30. Nan Wei, Changjun Li, Xiaolong Peng, Fanhua Zeng, and Xinqian Lu. Conventional models and artificial intelligence-based models for energy consumption forecasting: A review. *Journal of Petroleum Science and Engineering*, 181:106187, 2019.