Article

Artificial Neural Network Simulation of Energetic Performance for Sorption Thermal Energy Storage Reactors

Carla Delmarre ^{1,†}, Marie-Anne Resmond ^{1,†}, Frédéric Kuznik ¹ , Christian Obrecht ^{1,*}, Bao Chen ^{1,2}
and Kévy Johannes ¹

¹ Université de Lyon, CNRS, INSA-Lyon, Université Claude Bernard Lyon1, CETHIL UMR5008, 69621 Villeurbanne, France; carla.delmarre@insa-lyon.fr (C.D.); marie-anne.resmond@insa-lyon.fr (M.-A.R.); frederic.kuznik@insa-lyon.fr (F.K.); bao.chen@insa-lyon.fr (B.C.); kevy.johannes@insa-lyon.fr (K.J.)

² LafargeHolcim Innovation Center, 95 rue du Montmurier BP15, 38291 Saint-Quentin-Fallavier, France;

* Correspondence: christian.obrecht@insa-lyon.fr

† These authors contributed equally to this work.

Abstract: Sorption thermal heat storage is a promising solution to improve the development of renewable energies and to promote a rational use of energy both for industry and households. These systems store thermal energy through physico-chemical sorption/desorption reactions that are also termed hydration/dehydration. Their introduction to the market requires to assess their energy performances, usually analysed by numerical simulation of the overall system. To address this, physical models are commonly developed and used. However, simulation based on such models are time-consuming which does not allow their use for yearly simulations. Artificial neural network (ANN)-based models, which are known for their computational efficiency, may overcome this issue. Therefore, the main objective of this study is to investigate the use of an ANN model to simulate a sorption heat storage system, instead of using a physical model. The neural network is trained using experimental results in order to evaluate this approach on actual systems. By using a recurrent neural network (RNN) and the Deep Learning Toolbox in MATLAB, a good accuracy is reached, and the predicted results are close to the experimental results. The root mean squared error for the prediction of the temperature difference during the thermal energy storage process is less than 3 K for both hydration and dehydration, the maximal temperature difference being, respectively, about 90 K and 40 K.

Keywords: sorption thermal energy storage; zeolite; artificial neural network; recurrent neural network

Citation: Delmarre, C.; Resmond, M.-A.; Kuznik, F.; Obrecht, C.; Chen, B.; Johannes, K. Artificial Neural Network Simulation of Energetic Performance for Sorption Thermal Energy Storage Reactors. *Energies* **2021**, *14*, 3294. <http://doi.org/10.3390/en14113294>

Academic Editor: Simona Bennici and Mohamed Zbair

Received: 29 April 2021

Accepted: 31 May 2021

Published: 4 June 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Renewable energy deployment is often applied in conjunction with Thermal Energy Storage (TES) to balance the energy between production and demand, e.g., storing summer heat for winter heating. Chemical and sorption TES have been identified as promising technologies to solve the seasonal mismatch of solar energy storage [1] offering high energy densities around $600 \text{ kW} \cdot \text{h} \cdot \text{m}^{-3}$ and $200 \text{ kW} \cdot \text{h} \cdot \text{m}^{-3}$, respectively, [2]. Despite their high energy densities, chemical and sorption TES suffer from their low technology readiness level (typically two to three), which justifies the intensive research on this topic that has occurred in the last ten years [1–8]. The main scientific bottlenecks are in the improvement of the heat and mass transfer in the TES during hydration and to provide a better integration of the TES within the system to increase the overall efficiency. Recently, the analysis of the energy chain of a zeolite TES has shown that approximately 70% of absorbed energy is converted into useful heat released on discharge. However, approximately half of this heat is then directly lost at the outlet of the adsorbent bed. The overall system efficiency is 36% [9]. Numerical simulation is usually used to address such problems with tools like TRNSYS, ENERGY+... that require robust and time efficient TES numerical models compatible with yearly simulation.

Detailed physical models of TES have been developed and reported in the literature [9,10]. Usually based on the analysis of heat and mass transfer in the reactor, they are complex and require large computational efforts, making it difficult to use them for yearly simulations. One possible strategy to overcome this difficulty is the use of regression methods based on machine learning which are considerably less demanding in terms of computation. Support-vector regression (SVR) is one classic option [11] to reproduce complex time series but with the advances in machine learning, SVR approaches are nowadays generally outperformed by artificial neural network (ANN)-based techniques. Basically, an ANN is composed of different layers: an input layer with the information given to the network, hidden layers showing the influence among different elements, and an output layer which gives the desired results [12].

Recently, Scapino et al. [5] have developed a potassium carbonate (K_2CO_3) TES model based on an artificial neural network (ANN). The network was trained with a set of simulated data produced by a physics-based model. It showed a significant reduction in time of simulation and acceptable discrepancies with respect to the physics-based model. Taking into consideration these promising results, the objective of our research work is to develop a neural network that will assess the performance of a zeolite TES. In our approach, however, experimental data instead of simulation data are used for the training and the validation of the neural network. We chose to use a recurrent neural network (RNN) as this category of ANN was designed for sequence modelling and therefore naturally applies to time series forecasting [13]. The main reason for the effectiveness of RNN is its ability to capture past information from data and use it for upcoming sequence steps.

The studied sorption thermal energy storage system is presented in Figure 1. The main part of the system consists of two vertical packed-bed reactors (top and bottom, Figure 1b,d). Each cylinder reactor (72 cm in diameter) can load 40 kg of zeolite materials (Alfa Aesar, 13X, beads of 1.6 mm to 2.5 mm, see Figure 1a). An air treatment system (Figure 1c) drives the sorbate flow at various temperature and humidity levels into the reactors according to different dehydration and hydration tests. At the inlet and outlet of reactors, temperature sensors, hydrometers and flow meters are installed to measure the properties of airflow. More details about the reactor system are available in [2]. The performance of the system could be explored with the variation of charging temperature, airflow rate, relative humidity in desorption mode, bed thickness and serial or parallel reactor test configurations [2]. Since we aimed to approach daily use cases, several sets of results were chosen (see Section 2.1) for ANN training and validation.

Figure 1. Cont.

(c) Experimental setup without insulation

(d) Experimental setup with loaded and opened reactors

Figure 1. Pictures of the experimental setup and zeolite material, adapted from [14].

2. Methodology

2.1. Recurrent Neural Network

Recurrent neural networks (RNNs) form a category of ANNs which may be seen as general sequence processors. Contrary to more widespread ANNs, such as multilayered perceptrons which propagate data forwards only, RNNs propagate data both forwards and backwards. This feature allows RNNs to process sequences of data, and notably to handle time-dependent situations. In this study, we choose to use long short-term memory (LSTM) recurrent neural networks [13]. LSTM has the ability to overcome the vanishing and exploding gradient problems [15] often encountered in recurrent networks. Basically, this is achieved via special gates which determine which past information should be passed and which should be forgotten.

In this work, the data were organised to fit the created RNN model. The relevant parameters (temperature, air flow, air humidity, temperature difference) were selected based on a study of the material properties. Firstly, the same experimental data covering both cycles (hydration/dehydration) were used to train and validate the RNN model in order to verify the validity of this approach. Then, the data were separated into training and validation sets and used on the second RNN in order to provide a reliable evaluation of the accuracy of the model.

Training a neural network requires both a learning dataset and a validation dataset, with inputs and corresponding outputs. A large amount of learning data are recommended to gain a well-performing neural network. The chosen amount of validation data is usually significantly smaller. The root mean square error (RMSE, Equation (1)) was used to evaluate the accuracy of the model during the validation phase:

$$\text{RMSE} = \sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}}, \quad (1)$$

where $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n$ are the predicted values, $y_1, y_2, \dots, y_n$ are the observed values, and n is the number of observations.

It should be noted that the RMSE is the square root of the variance of the residuals, which is an absolute measure of fit. The lower its value, the more accurate the model is. However, the RMSE depending on the scale of the used data implies that there is no normalised value assessing the reliability of the model.

In order to construct a model which is as accurate as possible, the different parameters of a RNN (e.g., number of hidden layers, number of epochs and the number of batches) are generally tested to achieve the best combination. The number of epochs corresponds to the number of times that all the training sets will undergo processing. By increasing the number of epochs, the output of the neural network goes from underfitting, optimum

and finally to overfitting. The batch divides the total amount of training data. Its size corresponds to the number of training data that will be used in one pass. Therefore, the smaller this number is, the longer the training of the neural network will take, since a smaller amount of data will be used for each propagation. The number of hidden layers corresponds to the size of the network and the number of neurons that are used for learning. The larger this number is, the larger the neural network will be and the longer it will take to train it.

2.2. Data Processing

The data sets originating from [2] must be processed before training the neural network. The parameters selected from the data as inputs are: the relative humidity measured at the entry of the reactor (RH_{in}), the temperature of air flow at the entry of the reactor (T_{in}), the rate of air flow (Q_v) inside the reactor. The parameter selected as an output is the variation of the temperature (ΔT) between the entry of the reactor and the output temperature. Those parameters are used for both hydration and dehydration in order to construct a neural network that can predict both phases. The MATLAB code used for training and validation is given in Appendix A.

Six different experiments of hydration and three experiments of dehydration are used for the training. This corresponds to a total of 15,700 measurements. For the validation of the model, 3 experiments of hydration and 1 of dehydration, 5500 measurements in total, are used.

All experiments used during the training process have to be in the same database. Yet, for proper training of the neural network, the different experiments have to be segregated. Therefore, during data processing, all experiments are numbered. All the data from the first experiment will be preceded by one, and so on. Thanks to a data processing code (see Appendix B), the neural network can separate each experiment as a cycle. The experiments were organised in a random way.

3. Results and Discussion

3.1. Determination of the Optimal Rnn Parameters for Training

A parametric study on three RNN parameters (mini batch size, number of max epoch, number of hidden layers) was performed to analyse the RMSE of the outlet temperature. The lower the RMSE is, the better the result. One parameter is modified at a time whereas the two others are set to a constant value of 200. Figure 2 summarises the influence of each parameter and gives the RMSE as a function of time. Figure 2a shows the influence of the batch size. It indicates that a small mini batch size will give better results than a bigger one. Indeed, a smaller number of data go through one pass, and therefore the RNN will update itself more efficiently because it processes less data. Figure 2b shows the influence of the max epoch number. The curve representing the number of epochs has an erratically decreasing shape. It shows that there is an optimum number of epochs (800 or 1300) that is not necessarily the largest one. Moreover, the curve has a repetitive shape showing some determinism in the number of epochs for which training is efficient. Finally, Figure 2c shows an optimum number (400) of the hidden Layers that is not the highest one. It also suggests that an even number of hidden layers works better than an uneven number.

From Figure 2, an optimal solution can be extracted with a minimum batch size to use between 100 and 300, the number of hidden layers should be 400 and the number of epochs should be 800 or 1300. Indeed, the results gave the lowest RMSE for each simulation. Note that the calculation time increases considerably when the number of epoch or the number of hidden layers increases. For our data set, the calculation time remained lower than 3 h.

Figure 2. Graphics showing the evolution of the chosen RNN parameter when the other two have a value of 200, (a) shows the influence of the MiniBatchSize, (b) the influence of the MaxEpoch and (c) the influence of hidden layers.

3.2. Validation of the Model with the Same Training and Verification Dataset

According to the previous section the parameters of the RNN were set to (200, 800, 400). First, the model was trained with the data sets referenced as *training* in Tables 1 and 2, in total nine training simulations. Then with the same data sets, the results were compared in order to understand if the model is able to reproduce the results of the training data. The RMSEs are shown in Table 3 for the nine trainings. They range from 0.9 to 2.31, which is relatively low, and the mean value for these 9 simulations is 1.60. This result indicates that the network predicted correctly the outcome of the data set that was used to create it.

Table 1. Hydration data [2,14].

Experiment	T_{in} min (°C)	T_{in} max (°C)	RH_{in} min	RH_{in} max	Q_v min (m ³ · h ^{−1})	Q_v max (m ³ · h ^{−1})	ΔT min (°C)	ΔT max (°C)	Volume of Data	Training or Validation
1 series	19.71	20.39	0.0067	0.0073	127.9	189.8	−0.16	38.42	784	training
2 top	19.38	20.48	0.0074	0.0089	49.09	90.22	−0.40	37.69	2671	validation
2 bottom	19.45	20.59	0.0074	0.0089	56.21	95.85	−0.46	37.80	2671	training
4 top	17.49	20.21	0.0046	0.0050	91.49	99.83	−5.80	37.18	2272	validation
4 bottom	17.31	20.26	0.0046	0.0050	85.28	92.56	−7.61	37.18	2272	training
5 top	16.92	20.12	0.0038	0.0047	72.45	99.15	−0.48	26.52	2695	training
5 bottom	17.09	20.18	0.0038	0.0047	17.24	92.65	−0.53	26.60	2695	training
8 top	18.80	19.97	0.0025	0.0030	55.10	65.82	−0.12	36.68	2083	training
8 bottom	18.85	20.18	0.0025	0.0030	52.50	62.37	−0.52	36.60	2083	validation

Table 2. Dehydration data [2,14].

Experiment	T_{in} min (°C)	T_{in} max (°C)	RH_{in} min	RH_{in} max	Q_v min (m ³ · h ^{−1})	Q_v max (m ³ · h ^{−1})	ΔT min (°C)	ΔT max (°C)	Volume of Data	Training or Validation
4 top	19.27	121.97	0.0047	0.0111	76.69	99.82	−96.25	−0.20	544	validation
4 bottom	19.41	124.28	0.0047	0.0111	69.01	87.65	−102.99	−0.39	544	training
5 top	18.92	123.13	0.0048	0.0052	75.68	101.53	−100.04	0.60	980	training
5 bottom	18.88	126.60	0.0048	0.0052	66.82	86.10	−107.33	−0.80	980	training

Table 3. RMSE values of the prediction (same dataset).

RMSE1	RMSE2	RMSE3	RMSE4	RMSE5
1.66	1.32	0.90	2.10	2.02
RMSE6	RMSE7	RMSE8	RMSE9	Average
2.02	1.11	0.99	2.31	1.60

Figures 3 and 4 compare the results of the temperature difference as a function of the volume of data (the time step is one of 30 s for all the experiments) for both hydration and dehydration. As seen in Figures 3 and 4, the prediction curve correctly follows the shape of the experimental curve but is slightly offset.

Figure 3. Outcome of the prediction — best (left) and worst (right) predictions for hydration (Number of data on the x -axis, ΔT on the y -axis). **Figure 4.** Outcome of the prediction — best (left) and worst (right) predictions for dehydration (Number of data on the x -axis, ΔT on the y -axis).

The average RMSE is around 1.60, which validates the above method using RNNs. The model would still be considered as accurate when considering this precision.

3.3. Validation of the Model with Specific Dataset

Many simulations computing different sets of parameters were run and can be found in Appendix C. The (200, 1300, 400) set of parameters gives the most accurate results for our dataset (Figure 5).

Figure 5. Outcome of the prediction with the neural network of parameters (200, 1300, 400)—(number of data on the x -axis, ΔT on the y -axis)—(a) is dehydration, (b–d) are hydration.

The RMSE values for the specific dataset are shown in Table 4. The average RMSE of the simulation is 2.37. The predictions given by the RNN are accurate but not perfect. For example, on diagram (d), the prediction curve follows the experimental curve well, but is offset during all the drops in temperature. The other prediction curves, (a), (b), and (c), are more acceptable. Comparing Figure 5 to Figure 3, it can be assessed that the prediction is less satisfactory when the neural network has never met the data. The RMSE is not the same for all the hydration simulations. This result could perhaps still be improved using better-suited parameters.

Table 4. RMSE values of the prediction (specific dataset).

RMSE1	RMSE2	RMSE3	RMSE4	Average
2.91	2.11	2.59	1.87	2.37

4. Conclusions

An RNN model suited for zeolite-based thermal energy storage was constructed. For quoted experimental data, the RNN model gave accurate results on the prediction of the temperature of air flow coming out of the reactor on complete cycles of both dehydration and hydration of zeolite. The calculation time of RNN model is lower than that of physics-based models (less than 2 or 3 h to compute more than 15,700 estimations). Moreover, once the RNN model is trained, the calculation time to predict any given data set is almost instantaneous (less than a minute for more than 7500 estimations).

It should however be mentioned that in the predictions given by the RNN a gap between the experimental results and the predicted values remains. The accuracy of the predictions could still be improved, especially when significant changes in values occur over a short period of time. A possible cure to this issue could be to use more appropriate metrics such as dynamic time warping (DTW) [16] which is less sensitive to time shifts although more computationally demanding.

The results from the RNN are nevertheless promising. It should be noted that the predictions for the dehydration cycles are often more accurate than for the hydration cycles,

which may be due to the fact that the variation of the output parameter is twice as large in the dehydration cases than in the hydration cases. However, there are less experimental data available for the dehydration cycles. Furthermore, the experiments covered only full hydration and dehydration cycles. It would be more realistic to model the reactors' behaviour using cycles including both hydration and dehydration in an experiment to see if an RNN is able to model accurately such cases.

Alternative machine learning based approaches for time series prediction could also be considered in future work. We plan to experiment with the gated recurrent unit (GRU) approach [17] which can be seen as a simplified version of LSTM and has been shown to often perform better than LSTM on small datasets as in our case. We may also evaluate the use of convolutional neural network (CNN)-based models. CNNs were originally devised for pattern recognition but have been successfully applied to time series forecasting in recent years [18]. They are known to be resistant to noise which might be beneficial when using a experimental datasets.

Author Contributions: Conceptualization, F.K. and K.J.; Data curation, B.C.; Investigation, C.D. and M.-A.R.; Methodology, C.O.; Software, C.D. and M.-A.R.; Supervision, F.K. and K.J.; Visualization, C.D. and M.-A.R.; Writing—original draft, F.K.; Writing—review and editing, C.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data sets used in this study originate from [2].

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Training Code (Matlab)

```
% Create and train a Recurrent Neural Network (RNN)
%
% A set of data is used to train the network, where inputs
% and outputs are known. Another set of data is used to test
% (validate) the accuracy of the RNN.
%
% Graphs for each cycles will be plotted at the end of the
% program to compare the predicted values with the
% experimental ones.

%% Initialisation of the data
% Load the data from the files
filenamePredictorsX = fullfile("TrainingX.txt");
filenamePredictorsY = fullfile("TrainingY.txt");

% Put the data in matrix XTrain for the inputs and YTrain
% for the outputs using the prepareData() function
[XTrain] = prepareData(filenamePredictorsX);
[YTrain] = prepareData(filenamePredictorsY);

% Normalise matrix XTrain
mu = mean([XTrain{:}],2);
sig = std([XTrain{:}],0,2);

for i = 1:numel(XTrain)
    XTrain{i} = (XTrain{i} - mu) ./ sig;
end

%% Create the Neural Network

% Neural Network parameters
miniBatchSize = 100;
maxEpochs = 200;
numHiddenUnits = 200;

% Neural network initialisation
numResponses = size(YTrain{1},1);
```

```

featureDimension = size(XTrain{1},1);
layers = [ ...
    sequenceInputLayer(featureDimension)
    lstmLayer(numHiddenUnits,'OutputMode','sequence')
    fullyConnectedLayer(50)
    dropoutLayer(0.5)
    fullyConnectedLayer(numResponses)
    regressionLayer];

options = trainingOptions('adam', ...
    'MaxEpochs',maxEpochs, ...
    'MiniBatchSize',miniBatchSize, ...
    'InitialLearnRate',0.01, ...
    'GradientThreshold',1, ...
    'Shuffle','never', ...
    'Plots','training-progress',...
    'Verbose',0);

%% Train the Neural Network

net = trainNetwork(XTrain, YTrain, layers, options);

%% Save the Neural Network to a file

save ("netsave4","net");

%% Test the Neural Network

% Load the text files used to test the neural network and
% put them in a matrix

filenameResponsesX = fullfile("HDXValidation.txt");
filenameResponsesY = fullfile("HDYValidation.txt");
[XTest] = prepareData(filenameResponsesX);
[YVerif] = prepareData(filenameResponsesY);

% Normalise the test matrix

for i = 1:numel(XTest)
    XTest{i} = (XTest{i} - mu) ./ sig;
end

% Predict the outputs of the testing inputs through the
% neural network

YTest = predict(net,XTest);

%% Plot the results predicted with the neural network along
% with the experimental results using the plotResults()
% function

plotResults(XTest,YTest,YVerif)

%% Validation of the model

% Compute the RMSE at each cycle
% RMSE is an indicator of the discrepancy between
% experimental values and predicted values

RMSE = zeros(numel(XTest),1);

for i = 1:numel(XTest)
    RMSE(i)= sqrt(mean((YTest{i} - YVerif{i}).^2));
end

RMSE

%% plotResults() returns a figure for each test set plotting
%% the predicted values along with the experimental results.

```

```

function [] = plotResults(XTest,YTest,YVerif)
    for i = 1:numel(XTest)
        YTestcurrent=YTest{i}';
        YVerifcurrent=YVerif{i}';
        figure(i)
        plot(YVerifcurrent,'--')
        hold on
        plot(YTestcurrent,'.-')
        hold off
    end
end

```

Appendix B. Data Processing Code (Matlab)

*%% prepareData() returns a cell composed of one matrix per
%% experimental test data.*

```

function [XTrain] = prepareData(filenamePredictors)

    dataTrain = dlmread(filenamePredictors);
    numObservations = max(dataTrain(:,1));
    XTrain = cell(numObservations,1);

    for i = 1:numObservations
        idx = dataTrain(:,1) == i;

        X = dataTrain(idx,2:end)';
        XTrain{i} = X;
    end
end

```

Appendix C. Determination of Hyperparameters

Try	Mini Batchsize	Max Epoch	Hidden Layers	RMSE1	RMSE2	RMSE3	RMSE4	Mean RMSE
1	200	100	200	18.55	16.08	24.34	14.73	18.43
2	200	200	200	16.34	15.00	4.68	14.44	12.62
3	200	300	200	16.11	14.95	8.23	13.44	13.18
4	200	400	200	16.28	15.21	2.86	26.21	15.14
5	200	500	200	16.73	5.80	3.90	17.56	11.00
6	200	600	200	16.18	15.83	2.82	13.18	12.00
7	200	700	200	23.64	21.49	4.26	12.28	15.42
8	200	800	200	2.56	1.96	2.43	2.25	3.30
9	200	900	200	2.42	4.38	2.39	7.78	4.24
10	200	1000	200	4.94	2.71	5.49	13.67	6.70
11	200	1100	200	22.92	3.17	4.37	3.68	8.54
12	200	1200	200	24.30	16.60	6.60	14.80	15.58
13	200	1300	200	3.30	2.10	4.90	2.60	3.23
14	200	1400	200	9.00	12.00	2.80	10.70	8.63
15	50	200	200	16.15	15.07	9.14	13.61	13.49
16	100	200	200	17.41	15.63	4.76	14.33	13.03
17	200	200	200	16.50	15.10	4.40	13.50	12.38
18	300	200	200	15.60	15.10	3.50	14.20	12.10
19	400	200	200	16.60	15.10	3.40	14.10	12.30
20	500	200	200	17.00	15.40	5.30	14.10	12.95
21	600	200	200	16.20	16.00	8.50	14.20	13.73
22	700	200	200	16.20	15.70	4.60	14.10	12.65
23	800	200	200	20.30	15.20	9.70	23.00	17.05
24	900	200	200	16.30	15.20	5.30	13.90	12.68
25	200	200	100	16.10	15.60	6.20	14.20	13.03
26	200	200	200	16.80	14.80	5.40	13.80	12.70
27	200	200	300	16.50	15.20	24.50	14.10	17.58

28	200	200	400	7.80	4.60	4.10	3.90	5.10
29	200	200	500	14.00	14.50	3.50	14.10	11.53
30	200	200	600	16.28	15.68	7.87	13.86	13.42
31	200	200	700	13.44	15.37	7.19	14.97	12.74
32	50	800	400	6.76	14.82	2.89	14.25	9.68
33	200	1300	200	3.26	2.68	3.25	15.99	6.30
34	200	800	200	17.19	12.46	5.29	15.64	12.65
35	200	900	200	24.11	7.18	3.88	12.98	12.04
36	200	1300	400	2.91	2.11	2.59	1.87	2.37

References

1. Mahon, D.; Claudio, G.; Eames, P. A study of novel high performance and energy dense zeolite composite materials for domestic interseasonal thermochemical energy storage. *Energy Procedia* **2019**, *158*, 4489–4494. [\[CrossRef\]](#)
2. Johannes, K.; Kuznik, F.; Hubert, J.L.; Durier, F.; Obrecht, C. Design and characterisation of a high powered energy dense zeolite thermal energy storage system for buildings. *Appl. Energy* **2015**, *159*, 80–86. doi:10.1016/j.apenergy.2015.08.109. [\[CrossRef\]](#)
3. Chen, B.; Kuznik, F.; Horgnies, M.; Johannes, K.; Morin, V.; Gengembre, E. Physicochemical properties of ettringite/meta-ettringite for thermal energy storage: Review. *Sol. Energy Mater. Sol. Cells* **2019**, *193*, 320–334. [\[CrossRef\]](#)
4. Hongois, S.; Kuznik, F.; Stevens, P.; Roux, J.J. Development and characterisation of a new MgSO₄-zeolite composite for long-term thermal energy storage. *Sol. Energy Mater. Sol. Cells* **2011**, *95*, 1831–1837. doi:10.1016/j.solmat.2011.01.050. [\[CrossRef\]](#)
5. Scapino, L.; Zondag, H.A.; Diriken, J.; Rindt, C.C.; Van Bael, J.; Sciacovelli, A. Modeling the performance of a sorption thermal energy storage reactor using artificial neural networks. *Appl. Energy* **2019**, *253*, 113525. [\[CrossRef\]](#)
6. Korhammer, K.; Druske, M.M.; Fopah-Lele, A.; Rammelberg, H.U.; Wegscheider, N.; Opel, O.; Osterland, T.; Ruck, W. Sorption and thermal characterization of composite materials based on chlorides for thermal energy storage. *Appl. Energy* **2016**, *162*, 1462–1472. doi:10.1016/j.apenergy.2015.08.037. [\[CrossRef\]](#)
7. Fumey, B.; Weber, R.; Baldini, L. Sorption based long-term thermal energy storage—Process classification and analysis of performance limitations: A review. *Renew. Sustain. Energy Rev.* **2019**, *111*, 57–74. [\[CrossRef\]](#)
8. Cabeza, L.; Martorell, I.; Miró, L.; Fernández, A.; Barreneche, C. 1-Introduction to thermal energy storage (TES) systems. In *Advances in Thermal Energy Storage Systems*; Cabeza, L.F., Ed.; Woodhead Publishing Series in Energy; Woodhead Publishing: Cambridge, UK, 2015; pp. 1–28. [\[CrossRef\]](#)
9. Kuznik, F.; Gondre, D.; Johannes, K.; Obrecht, C.; David, D. Numerical modelling and investigations on a full-scale zeolite 13X open heat storage for buildings. *Renew. Energy* **2019**, *132*, 761–772. [\[CrossRef\]](#)
10. Tatsidjodoung, P.; Le Pierrès, N.; Heintz, J.; Lagre, D.; Luo, L.; Durier, F. Experimental and numerical investigations of a zeolite 13X/water reactor for solar heat storage in buildings. *Energy Convers. Manag.* **2016**, *108*, 488–500. doi:10.1016/j.enconman.2015.11.011. [\[CrossRef\]](#)
11. Drucker, H.; Burges, C.J.; Kaufman, L.; Smola, A.; Vapnik, V. Support vector regression machines. *Adv. Neural Inf. Process. Syst.* **1997**, *9*, 155–161.
12. Lawrence, J. Introduction to Neural Networks: Design. In *Theory, and Applications*; California Scientific Software: Nevada City, CA, USA, 1994.
13. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [\[CrossRef\]](#) [\[PubMed\]](#)
14. Kuznik, F.; Gondre, D.; Johannes, K.; Obrecht, C.; David, D. Sensitivity analysis of a zeolite energy storage model: Impact of parameters on heat storage density and discharge power density. *Renew. Energy* **2020**, *149*, 468–478. doi:10.1016/j.renene.2019.12.035. [\[CrossRef\]](#)
15. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **1994**, *5*, 157–166. [\[CrossRef\]](#) [\[PubMed\]](#)
16. Gold, O.; Sharir, M. Dynamic Time Warping and Geometric Edit Distance: Breaking the Quadratic Barrier. *ACM Trans. Algorithms* **2018**, *14*. [\[CrossRef\]](#)
17. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
18. Koprinska, I.; Wu, D.; Wang, Z. Convolutional neural networks for energy time series forecasting. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.