Predicting the hygrothermal behaviour of building components using neural networks

Astrid Tijskens1,*, Staf Roels1, and Hans Janssen1

1 Building Physics Section, Department of Civil Engineering, KU Leuven, Kasteelpark Arenberg 40 Bus 2447, 3001 Heverlee, Belgium

Abstract. Increasing the energy efficiency of the existing building stock can be accomplished by adding thermal insulation to the building envelope. In case of historic buildings with massive walls, internal insulation is often the only feasible post-insulation technique. Drawback of internal insulation is the modified hygrothermal response of the wall, which can result in moisture damage. Hence, it is crucial to assess the risk of damage accurately beforehand. Given the many uncertainties involved, a probabilistic assessment is advisable. This, however, would require thousands of simulations, which easily becomes computationally inhibitive. To overcome this time-efficiency issue, this paper proposes the use of neural networks to replace the original hygrothermal model. The neural network is trained on a small data set obtained from the hygrothermal model and can subsequently be used to predict the hygrothermal behaviour of building components with different boundary conditions and geometry. The transient nature of the hygrothermal behaviour requires a neural network type which can handle long-range time-dependencies. In the past, recurrent neural networks were often used for this type of data. Recently however, results indicate that convolutional neural networks can outperform recurrent neural networks on such tasks. This paper compares the prediction accuracy and training time of both neural network types for the prediction of the hygrothermal behaviour of building components.

1 Introduction

In recent years, traditional deterministic assessments in building physics have been replaced by probabilistic approaches [1, 2]. When evaluating the hygrothermal behaviour of a building component, there are many inherently uncertain parameters, such as the exterior climate, component geometry, material properties. A probabilistic approach takes these uncertainties into account, which allows evaluating the hygrothermal behaviour and the related damage risks more dependably. However, the commonly applied Monte Carlo method often involves thousands of simulations, which easily becomes computationally inhibitive, especially when analysing more-dimensional component connection details. To overcome this time-expense issue, the hygrothermal model can be replaced by a

* Corresponding author: astrid.tijskens@kuleuven.be

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).
metamodel, a much simpler mathematical model which aims at imitating the original model with a strongly reduced calculation time. In recent years, researchers in building physics have tried to develop metamodels to predict single response values, such as the cumulative heat demand and the number of hours with temperatures above 25°C [3], or the maximum relative humidity or maximum mould index [4], with varying success. However, to our knowledge, no metamodelling strategy exists which aims at predicting the hygrothermal response time series, such as the temperature and relative humidity inside a building component. This approach allows a much more flexible application of the metamodel: the user can choose whichever damage prediction model or performance evaluator to post-process the hygrothermal data. However, the hygrothermal response of a building component is highly non-linear and non-stationary, which means that traditional metamodelling strategies, such as multivariate adaptive regression splines, kriging and many others, will not be able to capture this complex pattern. Hence, this paper focusses on neural networks, and more specifically on recurrent and convolutional neural networks, as they have proven successful in other non-linear and non-stationary research applications.

These two types of neural networks can handle time series modelling – found to be crucial in [5] – each in their own way. Recurrent neural networks (RNN) have a cyclic structure that feeds the output from previous time steps into the current time step as input. Hence, RNNs can model temporal contextual information along time series data. The simple RNN easily suffers from the vanishing or exploding gradient problem though, which makes it difficult for the network to learn correlations between temporally distant events. In response, the long-short memory network (LSTM) was proposed by Hochreiter [6]. The LSTM model changes the structure of the hidden units to memory cells with gates. Via these gates, the LSTM unit decides what information to keep from its existing memory, while the simple recurrent unit overwrites its content at each time-step. Hence, if the LSTM unit detects an important feature from an input sequence at early stage, it easily carries this information over a long distance, thus capturing potential long-term dependencies. Consequently, the LSTM has been widely used for time-series forecasting and sequence-to-sequence modelling tasks. Convolutional neural networks (CNN) are a class of deep neural networks most commonly used to analyse images, but recently also for time series and sequence modelling. The CNN does not have memory the way the LSTM does. Instead, it has direct access to the input time steps. The filter slides over the input time steps, performs a matrix multiplication at every position and sums up the results to produce new feature maps, which are the input for the next convolutional layer. By stacking multiple convolutional layers, the final layer has access to information from a large number of input time steps, thus enabling the CNN to look back in history. The advantage of CNN’s over RNN’s is that all output time steps can be calculated in parallel, whereas the RNN needs the output at the previous time step to be able to calculate the next one, thus calculating sequentially.

In this paper, the LSTM and CNN are applied to replace the hygrothermal model in the framework of a probabilistic assessment of a massive masonry wall. Both network types are compared based on their prediction performance and training time.

2 Data and methods

2.1 Data description

Typically, when evaluating the hygrothermal performance of a massive wall, one is interested in frost damage at the exterior surface, wood decay of embedded wooden beam ends and mould growth on the interior surface. To estimate whether these damage patterns
will occur, the temperature (T), relative humidity (RH) and moisture content (MC) are monitored at the associated positions for a period of 4 years (see Table 1). The probabilistic aspect of the influencing parameters is taken into account by sampling the input parameter distributions, shown in table 2. Variability in boundary conditions is taken into account by using climate data of four different Belgian cities, as well as using different starting years in the sets of 30 years of climate data, and taking into account varying wall orientations, rain exposures, solar absorptions, exterior heat exchange coefficients and interior humidity loads. Additionally, different brick wall thicknesses are considered. The original data, used to train and evaluate the metamodels, is obtained via hygrothermal simulations in Delphin 5.8. In total, 320 samples are simulated, of which 256 samples are used for training and 64 for testing. The metamodels are trained to predict the T, RH and MC time series, given the time series of the external air temperature, relative humidity, wind-driven-rain load and short-wave radiation, and the internal temperature and relative humidity, as well as the scalar values of the sampled input parameters.

**Table 1.** The monitored quantities for the damage patterns at different positions in the wall.

| Damage pattern       | Position                                      | Quantity       |
|----------------------|-----------------------------------------------|----------------|
| Frost damage         | 0.5 cm from exterior surface                  | T, RH, MC      |
| Decay of wooden beam ends | 5 cm from interior brick surface*              | T, RH          |
| Mould growth         | Interior surface                              | T, RH          |

* no beam is modelled in the 1D configuration that is studied, these conditions are considered representative

**Table 2.** Probability distributions of probabilistic input parameters.

| Input parameter                        | Value                                           |
|----------------------------------------|-------------------------------------------------|
| External climate                       | D(Ghent; Gaasbeek; Ostend; StHubert)            |
| Internal climate                       | D(load A; load B) cfr. EN 15026                  |
| Ext. climate start year                | D(1,24)                                         |
| Wall orientation                       | U(0,360)                                        |
| Rain exposure factor                   | U(0; 2)                                         |
| Solar absorption                       | U(0.4; 0.8)                                     |
| Ext. heat transfer coeff. slope        | U(1, 8)                                         |
| Brick wall thickness [m]               | U(0.15; 0.5)                                    |

* U(a,b): uniform distribution between a and b

### 2.2 Metamodels

The LSTM network in this paper exists of four LSTM layers and ends with a fully connected output layer. Each LSTM layer has 128 memory cells. The architecture of the CNN network was based on the Wavenet architecture [7] and consists of stacked residual blocks with causal dilated convolutions and gated activation units. Causal convolutions ensure that the output of the filter does not depend on future inputs. In a dilated convolution, the filter is applied over an area larger than its length by skipping input values with a certain step. Stacked dilated convolutions enable networks to receive information over a large time span – the receptive field – with just a few layers, while preserving the input sequence length throughout the network as well as maintaining computational efficiency. The dilation factor is exponentially increased for every layer, allowing the receptive field to grow exponentially with network depth. Each residual block has a residual connection to the next block and a parameterised skip connection to the last two layers of the network, both of which speed up convergence and help training deep networks. The final layer of the network is a fully connected layer. In this paper, two stacks of 14 residual blocks are used. In each convolutional layer, 128 filters are used with a
length of 4. Both networks are trained using the Adam optimizer with a learning rate of 0.02. Before presenting data to the network, both the input and output data are standardised (zero mean, unit variance) to overcome scale differences.

The networks’ accuracy is evaluated by three indicators: the root mean square error (RMSE), mean absolute error (MAE), and coefficient of determination (R2), formulated as follows:

\[
RMSE = \sqrt{\frac{1}{n}\sum(y - y')^2} \quad MAE = \frac{1}{n}\sum|y - y'| \quad R^2 = 1 - \frac{\Sigma(y-y')^2}{\Sigma(y-\bar{y})^2}
\]  

where \(y\) is the true output, \(y'\) is the predicted output, \(\bar{y}\) is the mean of the true output and \(n\) is the total number of time steps. To overcome scale differences between the different outputs, these indicators are calculated from the standardised outputs. Although the networks are trained to predict all outputs simultaneously, these performance indicators are calculated for each output separately, as well as over all outputs simultaneously. The first allows assessing which outputs are more easy or difficult to predict, the latter which model performs best overall.

### 3 Results and discussion

Fig. 1 and Fig. 2 show the true output, the output predicted by the LSTM and CNN model respectively and the residuals of one sample. Only the temperature and moisture content for frost damage and the relative humidity for wood decay are shown, as these were found to be more difficult to predict. The RMSE and MAE performance indicators for these outputs are given at the top. It is apparent that both models perform equally well on predicting the temperature and moisture content for frost damage. However, only the CNN model is able to capture the pattern of the relative humidity at the wooden beam ends. When comparing the overall performance of both models, shown in Fig. 3, the CNN model clearly outperforms the LSTM model. The LSTM model, although specifically designed to carry long-term data, obviously struggles to remember the right information over very long time series. The CNN does not suffer this problem, as it has direct access to the past, due to its architecture with stacked causal dilated layers. On the other hand, the CNN prediction of relative humidity at the wooden beam ends clearly shows more variance between adjacent time steps than the LSTM prediction. Since the CNN calculates output time steps in parallel, it cannot use information about previous output time steps, whereas the LSTM calculates sequentially and thus starts from the previous output time step to predict the next.
Fig. 1. The true output (blue), the output predicted by the LSTM model (orange) and the residuals (grey) of one sample, shown for a period of two years.

Fig. 2. The true output (blue), the output predicted by the CNN model (orange) and the residuals (grey) of one sample, shown for a period of two years.

Fig. 3. The performance indicators show that the CNN model performs better than the LSTM model.

Finally, using two NVIDIA RTX2080 GPU’s, the total training time for the CNN and LSTM model is 45 min and three hours respectively. When employing the models, the CNN takes 6 sec to predict 64 samples, while the LSTM model needs 15 sec. As a reference, the original hygrothermal model needs 4 min to calculate a single sample. Hence, due to its parallel nature, the CNN is both faster to train and to predict.
4 Conclusion

In this paper, the hygrothermal simulation model for a massive masonry wall is replaced by a much faster metamodel. Two different types of neural networks were considered as metamodel: a long-short term memory network (LSTM) and a convolutional neural network (CNN). The LSTM calculates the output time steps sequentially; each output time step is based on the previous one but past input time steps cannot be accessed directly. The CNN, on the other hand, calculates the output time steps in parallel and has direct access to a large range of previous input time steps but not to previous output time steps. It was found that both network types were capable to predict the outputs that do not require a lot of information from the past, such as temperature profiles and the moisture content at the exterior surface. On the other hand, the CNN model performed better when predicting outputs that respond much slower to a change in boundary conditions and thus requires more detailed information from the past, such as the relative humidity at the wooden beam ends. Additionally, due to its parallel nature, the CNN requires much less training time and can predict faster than the LSTM model.

This research is part of the EU project RIBuild, which has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreement No 637268. This financial support is gratefully acknowledged.

References

1. H. Janssen, S. Roels, L. Van Gelder, Annex 55 Reliability of Energy Efficient Building Retrofitting - Probability Assessment of Performance & Cost (2013).
2. E. Vereecken, L. Van Gelder, H. Janssen, S. Roels, Energy Build. 89, 231–244 (2015).
3. L. VanGelder, P. Das, H. Janssen, S. Roels, Simul. Model. Pract. Th. 49, 245–257 (2014).
4. V. Marincioni, G. Marra, H. Altamirano-medina, Build. Environ. 137, 257–267 (2018).
5. A. Tijskens, S. Roels, H. Janssen, Conf. Proc. IBPC (2018).
6. S. Hochreiter, J. Urgen Schmidhuber, Neural Comput. 9, 1735–1780 (1997).
7. A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, arXiv:1609.03499 [cs.SD] (2016).