Coupling of neural models for predicting indoor temperatures and heating loads in buildings

Benoit Delcroix\textsuperscript{1}, Simon Sansregret\textsuperscript{1}, Michaël Fournier\textsuperscript{1}, Ahmed Daoud\textsuperscript{1}

\textsuperscript{1}Hydro-Québec Research Institute, Laboratoire des Technologies de l’Énergie, Shawinigan, QC, Canada

Abstract

Building energy models are critical to forecast the energy use and to improve the operations of HVAC systems. However, these models are building-specific, and their development is tedious, error-prone and time-consuming. Compared to traditional white-box and grey-box models, black-box models need less development time and no information about the building properties, and only rely on collected data. In this work, a model coupling two neural networks is developed and used to simulate the building energy behaviour: both networks predict successively the indoor temperatures and heating loads of each room. The model is trained, validated and compared to experimental data obtained for seven houses in Canada heated by electric baseboards controlled by connected thermostats (on average, ten thermostats per house). For simulations with a time horizon of two days and a time-step of one hour, errors are promising, especially in winter where root mean square errors are up to 0.29 °C for indoor temperatures and 1050 Wh for heating loads. In summer, errors are higher due to the free-floating nature of simulations, with root mean square errors up to 1.09 °C for indoor temperatures and 139 Wh for heating loads.

Key Innovations

- Coupling of two neural models to mimic the techniques used in initial-value problem solvers that are applied in building energy simulation tools.
- Black-box model generalizable to all buildings without major modifications.

Practical Implications

The proposed approach is interesting for building managers dealing with many buildings, who need to implement in a timely manner models forecasting indoor temperatures and heating loads for diverse applications such as model predictive control, and fault detection and diagnosis.

Introduction

According to the IEA (International Energy Agency, 2020), buildings represent more than a third of the global energy consumption, but they also have an enormous efficiency potential. Energy efficiency measures can be assessed using building energy models. Zhao and Magoulès (2012), and Fouququier et al. (2013) presented an overview of models for building energy consumption prediction, including the three main modelling categories: physics-based (white-box), data-driven (black-box) and hybrid (grey-box) models. Coakley et al. (2014) contrasted these approaches in terms of advantages and drawbacks. White-box methods simulate buildings in details, modelling all the aspects such as the envelope, HVAC system and control. Nevertheless, the development of such models is error-prone and time-consuming, and they require detailed building information and a solid knowledge in building engineering. Crawley et al. (2008) contrasted some of the most popular tools for building energy performance simulation using a white-box approach, including software’s like TRNSYS (Beckman et al., 1994) and EnergyPlus (Crawley et al., 2001). On the other hand, black-box methods are fast to develop and can provide good precision if accurate data is available in large quantities. However, their parameters are not easily interpretable from a physical point of view. Amasyali and El-Gohary (2018) overviewed black-box approaches used for building energy consumption prediction, including machine learning techniques and other statistical methods. Grey-box methods are in-between approaches, using a simplified physics-based approach whose the unknown parameters are calibrated by minimization of differences between observed and simulated data. Typical examples are inverse grey-box models (Braun & Chaturvedi, 2002), resistance-capacitance models (Bueno et al., 2012), state-space models (Hu & Karava, 2014) and thermal network models (Xu & Wang, 2008). However, these models are generally custom-made, and they are therefore difficult to develop in an automated and timely manner, particularly when several buildings are considered. In this context, black-box approaches are more appropriate. Among them, neural networks are popular to tackle ill-defined and complex problems. Neural models are used for building energy simulation, especially for inside temperature prediction (Delcroix et al., 2020; Frausto & Pieters, 2004; Kramer et al., 2012; Mechaqrane & Zouak, 2004; Mustafaraj et al., 2011) and for energy use forecasts (Bennett et al., 2014; Ruiz et al., 2016). Unfortunately, the predictions of indoor temperatures and energy use are generally addressed separately. The current work proposes to tackle this issue with an approach coupling predictions of indoor temperatures and heating loads. Improved control of building components is a key end application of simplified (grey-box or black-box) building models. An obvious example is model predictive control,
which has the potential to increase the building energy efficiency (Oldewurtel et al., 2010) and thermal comfort (Castilla et al., 2014). Another key end application is automated fault detection and diagnosis, which is mainly achieved through data-driven (black-box) methods, as indicated by Kim and Katipamula (2018).

As suggested in this section, black-box approaches are an adequate manner to develop rapidly models for multiple buildings with mostly unknown properties but with a large quantity of accurate data. In this work, we propose an approach based on the coupling of two neural models for predicting iteratively the inside temperatures and heating loads. Both models are separately trained and validated using experimental data. They are then coupled to simulate the whole building. In the following sections, the data, their pre-processing, the modelling approach, the error indicators, and a comparison between simulations and observations are presented.

Materials and methods

This section provides a description of the data of interest, their pre-processing, the models used to predict indoor temperatures and heating loads, and the error indicators.

Data description

Most of the data comes from seven houses in Canada (province of Quebec). Table 1 presents the house properties. For all of them, the data cover almost two years. In Quebec, it is common that each room of a house is heated by one or several electric baseboard(s), each one being controlled by a thermostat.

Table 1: House properties

<table>
<thead>
<tr>
<th>ID</th>
<th>Area [m²]</th>
<th># thermostats</th>
<th>Heating capacity [kW]</th>
<th>Start date</th>
<th>End date</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>313</td>
<td>15</td>
<td>22.0</td>
<td>2016-11-17 13:00</td>
<td>2018-09-24 19:00</td>
</tr>
<tr>
<td>84</td>
<td>135</td>
<td>9</td>
<td>10.2</td>
<td>2016-11-23 16:00</td>
<td>2018-09-24 19:00</td>
</tr>
<tr>
<td>88</td>
<td>219</td>
<td>8</td>
<td>10.6</td>
<td>2016-12-16 10:00</td>
<td>2018-09-24 19:00</td>
</tr>
<tr>
<td>92</td>
<td>216</td>
<td>10</td>
<td>17.2</td>
<td>2016-12-01 10:00</td>
<td>2018-09-24 19:00</td>
</tr>
<tr>
<td>93</td>
<td>209</td>
<td>8</td>
<td>13.0</td>
<td>2016-12-15 10:00</td>
<td>2018-09-24 19:00</td>
</tr>
<tr>
<td>94</td>
<td>163</td>
<td>11</td>
<td>13.4</td>
<td>2016-12-01 15:00</td>
<td>2018-08-16 14:00</td>
</tr>
<tr>
<td>95</td>
<td>152</td>
<td>9</td>
<td>9.7</td>
<td>2016-12-13 10:00</td>
<td>2018-09-24 19:00</td>
</tr>
</tbody>
</table>

Table 2 presents the data of interest used in this work to develop the forecasting models.

Data pre-processing

Data pre-processing is necessary to yield an adequate and clean dataset, which will be used to train and validate the models. Table 3 lists seven data pre-processing techniques applied to the data presented in Table 2.

As noted in Table 2, the data have different units and value ranges, which can make the model training step less efficient. It is therefore often advisable to normalize the data, as recommended by Zhang et al. (1998). In this case, the heating loads are normalized to have values ranging from 0 to 1 (0 = no heating; 1 = maximum heating) while all other data (except one-hot encoded variables) are normalized to have a mean of zero and a standard deviation of one. Normalization can be an issue for set-point temperatures if this variable remains constant, the variance being zero. A solution consists in using the mean and standard deviation of the inside temperature measurements to normalize the set-point temperatures.

Table 2: Data description

<table>
<thead>
<tr>
<th>Name [Unit]</th>
<th>Description</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inside temperature [°C]</td>
<td>1-min sampling</td>
<td>In-house (connected thermostats)</td>
</tr>
<tr>
<td>Set-point temperature [°C]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heating load [Wh]</td>
<td>1-hour sampling</td>
<td></td>
</tr>
<tr>
<td>Outside temperature [°C]</td>
<td></td>
<td>(National Renewable Energy Laboratory, 2020)</td>
</tr>
<tr>
<td>Wind speed [km/h]</td>
<td>30-min sampling</td>
<td></td>
</tr>
<tr>
<td>Global horizontal irradiation [W/m²]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date [YYYY-MM-DD HH-MM]</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
</tbody>
</table>

Table 3: List of pre-processing steps

<table>
<thead>
<tr>
<th>Pre-processed data</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>All data, except heating load</td>
<td>Hourly sampling (mean)</td>
</tr>
<tr>
<td>Date</td>
<td>Get day of week and hour of day</td>
</tr>
<tr>
<td>All data</td>
<td>Drop samples with NaN value(s)</td>
</tr>
<tr>
<td>Day of week and hour of day</td>
<td>One-hot encoding</td>
</tr>
<tr>
<td>All data, except heating load</td>
<td>Normalization (standardization)</td>
</tr>
<tr>
<td>Heating load</td>
<td>Normalization (min-max [0-1])</td>
</tr>
<tr>
<td>All data</td>
<td>Dataset partitioning</td>
</tr>
</tbody>
</table>

To sum up, the complete dataset (almost two years like presented in Table 1) includes the following variables for each house and one-hour time-step: normalized weather data (outdoor temperature, solar radiation and wind speed), normalized inside temperatures, normalized set-point temperatures, normalized heating loads, day of the week (one-hot encoded) and hour of day (one-hot encoded). This dataset is divided into four parts:

- Two days in summer (1st and 2nd of July 2018) and winter (1st and 2nd of January 2018) are put aside to validate the simulation results;
- 80% of the remaining data is randomly allocated to the model training step. Among this 80%, 25% is randomly put aside to validate the training (early stopping technique). It means that 60% of the data is used to train the model (training dataset) and 20% is used to validate the training (validation dataset);
- The last 20% of the remaining data is used to test the model after training (test dataset).

Dataset partitioning ensures that the generated models are not subject to overfitting. Overfitting minimization through data partitioning and early stopping is well documented (Caruana et al., 2001; Prechelt, 1998).

Modelling methodology

In this work, feedforward neural networks, also called multilayer perceptrons (Zhang et al., 2019), are used to forecast inside temperatures and heating loads. A neural
network consists of a network of computing units, called 'neurons', whose purpose is to establish a mathematical relationship between input and output data. Each neuron performs a simple computational task that is formulated mathematically as follows:

\[ Y_n = f_{act} \left( b + \sum_{i=1}^{m} w_i x_i \right) \]  

(1)

Where \( Y_n \) is the neuron output; \( b \) is a constant bias associated with the neuron; \( m \) is the number of inputs; \( x_i \) is the value of input \( i \); \( w_i \) is the weight applied to input \( i \); \( f_{act} \) is the activation function, which allows neural networks to consider non-linear phenomena. In this work, the activation function is ReLU (Rectified Linear Unit). The association of neurons forms a network of neurons. The network complexity depends on the number of inputs, outputs, hidden layers and neurons in each layer, as presented in Figure 1.

The choice of the number of layers and neurons is complex and depends mainly on the number of inputs, outputs and samples (Sheela & Deepa, 2013). The neural network must be neither too small ("weak explanatory power") nor too large (increase in computing time and potential overfitting). In this work, the number of hidden layers has been set to three and the number of neurons per layer \( N_n \) is defined according to the following empirical and custom-made equation:

\[ N_n = \min(\round(\sqrt{N_i \times N_0} + \sqrt{N_3}), 200) \]  

(2)

Where \( N_i \) is the number of input variables; \( N_0 \) is the number of output variables; \( N_3 \) is the number of training samples. The \( \min \) function ensures that \( N_n \) is maximum 200. The most complex network, i.e., \( 3 \times 200 \), can be considered relatively large. Nevertheless, the risks of long computation time and overfitting remain under control. As highlighted later in the results section, the computation times remain low. On the other hand, data partitioning (training, validation and test), early stopping, and regularization (see next paragraph and equation (3)) mitigate the risk of overfitting. Kukacka et al. (2017) provided an overview of methods that aim at making the model generalize better, i.e., produce better results on the test set.

Training a neural network involves finding the weights \( w_i \) and biases \( b \) that minimize the error between observed and simulated data \( y \) and \( \hat{y} \), respectively. Starting from initial random weights and biases, the algorithm minimizes the loss function by repeatedly updating these weights and biases. In this work, the neural models use the square error loss function with a L2-regularization term \( \alpha \) (penalty) (Schmidt, 2016), which is expressed as:

\[ \text{Loss}(\hat{y}, y, W) = \frac{1}{2} (\hat{y} - y)^2 + \frac{\alpha}{2} ||W||_2^2 \]  

(3)

Where \( W \) is the matrix of weights and \( ||W||_2 \) is the L2-norm or Euclidean norm of the matrix \( W \). Using a regularization term in the loss function is an additional solution to avoid overfitting. In this work, \( \alpha \) is 0.0001.

In this work, two neural models are trained, one for indoor temperature prediction and one for heating load prediction. The first model forecasting indoor temperatures is formalized as follows:

\[ T_{t+1} = f_{\text{NN1}}(X_{\text{model1}}) \]  

(4)

Where \( T_{t+1} \) is a vector including the indoor temperatures of all rooms at time-step \( t + 1 \); \( f_{\text{NN1}} \) is the trained neural network; \( X_{\text{mode1}} \) is an input vector including:

- \( T_t \) is a vector including the inside temperatures of all rooms at time-step \( t \);
- \( E_t \) is a vector including the heating loads of all rooms at time-step \( t \);
- \( T_{\text{set}, t-t+1} \) is a vector including the set-point temperatures in all rooms at time-steps \( t \) and \( t + 1 \);
- \( T_{\text{out}, t-t+1} \) is a vector including the outdoor temperatures at time-steps \( t \) and \( t + 1 \) (known from the weather forecast);
- \( SR_{t-t+1} \) is a vector including the global solar radiation at time-steps \( t \) and \( t + 1 \) (known from the weather forecast);
- \( WS_{t-t+1} \) is a vector including the wind speed at time-steps \( t \) and \( t + 1 \) (known from the weather forecast);
- \( \text{Day}_{t+1} \) is a vector including the day of week at time \( t + 1 \) (one-hot encoded);
- \( \text{Hour}_{t+1} \) is a vector including the hour of day at time \( t + 1 \) (one-hot encoded).

The second model for forecasting heating loads can be formalized as follows:

\[ E_{t+1} = f_{\text{NN2}}(X_{\text{model2}}) \]  

(5)

Equation (5) is close to equation (4). The differences are:

- The output is the heating loads \( E_{t+1} \);
- \( f_{\text{NN2}} \) is another neural network;
- \( X_{\text{model2}} \) includes the same elements as in \( X_{\text{mode1}} \), and \( T_{t+1} \) from equation (4).

Coupling of both models

The slight difference between equations (4) and (5) allows the coupling of both models to simulate the behaviour of a building whose initial conditions at time \( t \) are known and whose the states at time-steps \( t + 1 \), \( t + 2 \), etc. are desired. Figure 2 presents a flow diagram on how the coupling, and more generally the simulation, works. In brief, the coupling takes place in two steps:
• Step 1: using the initial conditions at time \( t \), the inside temperatures \( T_{t+1} \) are computed using a first neural network, as formalized in equation (4);
• Step 2: \( T_{t+1} \) being found through step 1, step 2 consists in predicting the heating loads \( E_{t+1} \) in the building at time \( t + 1 \) using a second neural network, as formalized by equation (5).

Iteratively (by repeating steps 1 and 2), indoor temperatures and heating loads can be determined up to the desired time horizon. In theory, the most constraining factor is the quality of the weather forecast, which decreases with the time horizon. In this work, real weather data are used.

The rationale presented in Figure 2 mimics the operating principle of the vast majority of building energy simulation software, i.e. using initial values to simulate the building’s behaviour.

In this work, the simulation time-step is one hour. However, Delcroix et al. (2020) used a similar method in a case with a simulation time-step of five minutes. In their work, the inputs do not only include one previous time-step, but several (up to twelve).

![Algorithm flow diagram](image)

**Error indicators**

Two error indicators are used in this work:

- The mean bias error (MBE): it is an adequate indicator to assess whether the model overestimates or underestimates the observed values, overall. This indicator is expressed in the same unit as the results of interest, i.e., in degrees Celsius for temperatures and in Wh for heating loads. If it is positive, the model overestimates the reality. The MBE is calculated as follows:

\[
MBE = \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)}{n}
\]  

Where \( n \) is the number of samples; \( y \) is the observed value; \( \hat{y} \) is the predicted value.

- The root mean square error (RMSE): it is a measure of the average difference observed between the actual and predicted values. Like the MBE, this indicator is expressed in the same unit as the results of interest. The RMSE is defined as follows:

\[
RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}
\]  

Normalized versions of the RMSE and MBE indicators are also used for the heating loads in the following section on results to consider differences in installed baseboard capacity in the houses. The NRMSE and NMBE are calculated as follows:

\[
NRMSE = \frac{RMSE}{E_{max} - E_{min}}
\]

\[
NMBE = \frac{MBE}{E_{max} - E_{min}}
\]

Where \( E_{max} \) and \( E_{min} \) are the maximum and minimum hourly heating consumption observed in the data.

These indicators are used to define the quality of the training as well as the quality of the simulations. Between the training and simulation phase, we will observe an increase in the error. It can be explained as follows:

- The error is evaluated sample by sample (time-step by time-step) during training;
- Only the initial conditions are provided in simulation, with the models using the outputs of the other model to simulate the desired time horizon. This leads to a propagation of the error over time and thus an increase in the error in general.

**Results and discussion**

For more readability, results presented in this section are aggregated at the house level. For example, results from house 80 with 15 thermostats are presented as the averages of the 15 indoor temperatures and the sums of the 15 heating loads. However, as a reminder, the models do not simulate the aggregate behaviour of the house, but all the thermostats individually. However, hints about the models’ accuracy at the room or thermostat level are given in Tables 4 and 5.

Table 4 presents the training errors (sample by sample or time-step by time-step) between simulated and observed data in the test dataset (unseen during training) for each of the seven houses. Training errors are systematically low: \(< 0.5 \, ^\circ C\) for temperatures, and \(< 350 \text{ Wh} \), or \(< 3.5 \%\) for heating loads (absolute values). The measurement errors of the thermostats are approximately \( \pm 0.5 \, ^\circ C\).
A first observation shows that the models forecast more accurately during winter, compared to summer, especially for indoor temperature predictions (see the rows “Avg.” and “Avg.*”). A potential reason is the free-floating nature of the simulation during summer. Room set-point temperatures are inputs providing major hints for indoor temperature predictions, especially in winter when the heating system is active (vs. mostly inactive in summer). In summer, these inputs do not provide additional hints. Moreover, the behaviour of occupants is difficult to forecast (e.g., window opening), leading to additional uncertainties.

In the winter cases, the models always provide a good accuracy for indoor temperature predictions, i.e., with absolute errors (RMSE and MBE) always below 0.50 °C. For the heating loads, prediction errors are higher than 1000 Wh or 10 % for three cases: houses 84 (RMSE and NRMSE), 92 (only NRMSE) and 93 (only NRMSE).

In summer, only two cases (Houses 93 and 94) have errors lower than 0.5 °C for indoor temperature predictions. In terms of heating loads, the prediction errors are most of the time low because little heating is needed during the summer. All the cases with values “n.a.” are cases without observed heating loads, i.e., for all houses except 84. For all these cases, the models slightly overestimate the heating loads (MBE > 0 Wh). For house 84, the model underestimates the heating loads with significantly high errors in terms of NRMSE (26.83 %) and NMBE (-13.91 %). If RMSE and MBE are considered, the errors are relatively low, compared to what is observed in the winter cases.

To illustrate the simulation results in more details, we have chosen two houses, representing one of the best case (house 94 - Figure 3) and one of the worst case (house 84 - Figure 4) in terms of prediction accuracy (see Table 5).
Figure 3 presents the comparison between the reference and simulated data for house 94 during two days in winter (1\textsuperscript{st} and 2\textsuperscript{nd} of January 2018). In the graph (a), the average of all the observed indoor temperatures (from 11 thermostats) is compared to the average of all the simulated indoor temperatures. The average of all the setpoint temperatures is also given for information. We can observe that reference and prediction data remain close during the full two-day period (inside the interval of ±0.5 °C). In the graph (b), the total observed heating loads are compared to the total simulated values for the same period. The model replicates adequately the reference data.

If we look at the same period for house 84 (one of the worst case) in Figure 4, the indoor temperature predictions (average for 9 thermostats) in graph (a) are as accurate as for house 94. On the other hand, the heating load predictions in house 84 in graph (b) are less precise, compared to house 94. Inaccuracies appear particularly during the second day when the peak demand is not replicated by the model (Reference > Prediction). This inaccuracy could potentially be the consequence of the propagation of error over time during the simulation. Another source of error can be related to internal gains that are unpredictable (e.g., household appliances).

When considering neural networks, considerations about computing times always arise since these models can be resource-intensive, especially during training. For information, the work was performed on a laptop (16 GB RAM and i7 processor). Table 6 presents the computing times for training and simulation for each house. The neural networks architectures are also given for each house and model (indoor temperature and heating load neural models). As stated previously, the number of hidden layers is set to three, and the number of neurons per layer depends on the number of training samples, inputs and outputs (see equation (2)).

As presented in Table 6, the training time ranges between 18 and 42 seconds, which is relatively low if we compare to computing times generally observed in other fields like computer vision. As to the simulation step, the computing times (two days with a time-step of one hour) are low at around 0.25 second.

Finally, the main tools used in this work for data preprocessing, model development and training, and simulation are the Python language, including the following main libraries: Pandas (McKinney, 2010) and Scikit-Learn (Pedregosa et al., 2011).
Table 6: Computing times for training and simulation

<table>
<thead>
<tr>
<th>ID</th>
<th>Neural network architecture (for temperatures and heating loads)</th>
<th>Training computing time [s]</th>
<th>Simulation computing time (2-day hourly simulation) [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>3x150 and 3x138</td>
<td>41.46</td>
<td>0.25</td>
</tr>
<tr>
<td>84</td>
<td>3x138 and 3x139</td>
<td>28.59</td>
<td>0.27</td>
</tr>
<tr>
<td>88</td>
<td>3x134 and 3x135</td>
<td>27.38</td>
<td>0.24</td>
</tr>
<tr>
<td>92</td>
<td>3x139 and 3x141</td>
<td>28.30</td>
<td>0.26</td>
</tr>
<tr>
<td>93</td>
<td>3x133 and 3x135</td>
<td>18.12</td>
<td>0.26</td>
</tr>
<tr>
<td>94</td>
<td>3x138 and 3x140</td>
<td>26.67</td>
<td>0.22</td>
</tr>
<tr>
<td>95</td>
<td>3x136 and 3x137</td>
<td>32.32</td>
<td>0.26</td>
</tr>
</tbody>
</table>

In summer, the simulation errors are significantly higher for indoor temperature predictions with $RMSE$ up to 1.09 °C and $\|MBE\|$ up to 0.74 °C. It is potentially due to the free-floating nature of simulations in summer. Since there is little heating in summer, the simulation errors for heating load prediction are low: $RMSE < 140$ Wh and $\|MBE\| < 80$ Wh.

In terms of computing times, the neural models do not turn out to be resource-intensive, compared to neural models in other fields like computer vision. In this work, the training step and two-day simulations took respectively up to 42 and 0.27 seconds with a laptop (16 GB RAM and i7 processor).

The global performance (low errors and computing times) of the presented approach, i.e., the coupling of two neural networks for building energy simulation, is in agreement with potential applications like model predictive control, and automated fault detection and diagnosis. However, further work should be dedicated to improving the models’ performance in summer conditions and to develop a more formal tool for automated building model development.

Acknowledgement

The authors would like to acknowledge the high quality and open source resources made available online, especially by the Python community.

References


Castilla, M., Álvarez, J., Normey-Rico, J., & Rodríguez, F. (2014). Thermal comfort control using a non-linear...


