Development of a surrogate model for interactive early-stage net-zero building design

David Rulff1, Ralph Evins1, Kevin Cant1
1Energy in Cities group, Department of Civil Engineering, University of Victoria, Canada

Abstract

This paper outlines the process of developing and testing a large-scale surrogate model of building energy use. A surrogate model is a machine-learning model, in this case a deep neural network, used to approximate the behaviour of a detailed simulation, in this case an EnergyPlus model. The fast but approximate model can be used for interactive, exploratory decision-making, which is particularly useful in early-stage design. The surrogate model developed in this paper underpins the Net-Zero Navigator platform, which makes the model accessible via the web using interactive visual controls and outputs. The surrogate model has 39 inputs spanning all areas of building thermal performance (window and shading geometry, insulation, set points, ventilation, thermal mass and internal gains). This paper explores some of the key aspects of surrogate model development that were encountered at this scale. Key issues include the number of samples needed to train the model, tuning the hyperparameters of the machine learning model, the importance of the scaling functions used to normalise the training data, and the different metrics that can be used to judge the accuracy of the model.

Highlights

- We present a large-scale surrogate model (38 input parameters, 3 output parameters)
- We perform a comparison of a surrogate model using a deep neural network to a simple multiple linear regression baseline.
- We explore the value of examining the coefficients of a simple linear model to determine the importance of each parameter to each output
- We document the importance of hyper-parameter optimization (HPO) in the development of surrogates
- We investigate the impact of sample set size, including a very large set (100,000 samples).
- We compare and contrast various error metrics for quantifying model accuracy.
- We discuss the importance of standardization of model inputs and in particular model outputs, and compare alternative methods including a Box-Cox transformation.

Introduction

Surrogate modelling in building design

Surrogate modelling is emerging as a key means of delivering value to users in the field of building energy modelling. It provides an avenue to build upon the underlying detailed simulations to provide rapid design space exploration and sensitivity analysis (Westermann & Evins, 2019). Detailed simulations will always be required in the detailed stages of design, but the early-stage decision-making process values flexibility and speed over absolute accuracy. There is something intuitive about exploring a problem by making rapid incremental changes to a design and seeing the various metrics of interest change in real-time.

Brief overview of surrogate modelling

The process of surrogate modelling involves performing a large batch of simulation runs of a parametric building energy model. The parameters define the inputs to the surrogate model; they are varied randomly to sample the design space of the model. All other model inputs are left unchanged; these become the ‘context’ in which the surrogate model is valid. The inputs and outputs of the set of parametric simulation runs becomes the training and testing data for fitting a machine learning model.

Often a neural network is used as the machine learning model as they are well-suited to regression problems. Neural networks learn to represent arbitrary functional relationships by mimicking the way neurons in the brain process information. They consist of layers of nodes, densely connected by links to the preceding layer. Each node applies an activation function to the sum of all incoming links multiplied by their respective weights. Non-linear activation functions allow the network to represent non-linear relationships; traditionally a sigmoid function was used. The input nodes correspond to the input data of the regression problem; the output node(s) correspond to the output variable of the regression problem. The weights of the connections in the ‘hidden’ layers are trained to minimise a loss function using the gradient-descent method assisted by ‘back-propagation’; weights are incrementally updated each epoch in the direction that reduces the overall error. The loss function minimises the difference between the model output and the target values over the training set; often mean squared error is used.
Issues with surrogate modelling

We will briefly address some common criticisms of surrogate modelling, to set the scene for some of the investigations undertaken in this paper.

*Is this just a simple regression model?* A surrogate model is a regression model, but not a simple one. The neural network architecture can learn very nuanced, detailed trends in the dataset. Sometimes it is true that this approach is overly complicated for a given problem, but it is good to use a method that works in both simple and challenging scenarios. For comparison in this paper we provide an analysis of a multiple linear regression model.

*If speed is the aim, why not use a simple physics-based model?* There are indeed many useful, simple models based on monthly calculations, R-C network formulations etc. However:

- There may be aspects of building performance that are important in early-stage design that require a detailed model to capture them.
- There may be issues achieving consistency with subsequent phases of design, when a detailed model will be required.

*How do we know if the surrogate is accurate enough?* Generally, statistical metrics show that surrogate models can achieve high levels of accuracy in representing the behaviour of the underlying detailed model. However, evaluating surrogate model accuracy is a key area where further research is needed. Commonly-used metrics like goodness-of-fit ($R^2$) values can give very good results while masking poor performance in other metrics (Chakraborty & Elzarka, 2018). This paper examines various accuracy metrics alongside $R^2$. We are not suggesting that the surrogate model presented here is sufficiently accurate to be useful in all contexts and for all outputs; as we will see in the results section, further work is necessary particularly around ‘intermediate’ outputs like cooling energy demand intensity. However, this paper shows that reasonable accuracy according to various metrics can be obtained for very large problem spaces.

The Net-Zero Navigator platform

The surrogate model developed in this paper is part of the set of models that will underpin the Net-Zero Navigator platform (netzeronavigator.ca). The platform was first introduced in (Paul Westermann et al., 2021), which includes details on the purpose and structure of the platform. Net-Nero Navigator is a continuation of the BESOS (Building and energy simulation, optimization and surrogate modelling) platform (besos.uvic.ca) (Westermann et al., 2021). The BESOS platform was used to undertake the experiments in this paper.

Methodology

Surrogate modelling process

The process of surrogate modelling involves the steps outlined below. We provide more detail in subsequent sections on specific steps where more investigation was conducted.

1. Prepare a building simulation model representing the base case design.
2. Determine which aspects of the model to vary, and set it up to parametrically control these. These will define the input parameter space of the surrogate model.
3. Determine which model outputs are of interest. These define the surrogate model output space.
4. Generate a large number of random samples in the input parameter space.
5. Run the batch of detailed simulations corresponding to these inputs, and collate the results relating to each output. This gives two large data tables, each with one row per sample, one with columns for each input parameter and one with columns for each output.
6. Before fitting the model, normalise or standardise the input and output data to make training easier. Also convert categorical parameters into separate parameters using one-hot encoding; each category choice becomes a parameter that takes 1 if that category is selected and 0 otherwise.
7. Using the tables of inputs and outputs, fit a machine learning model that predicts the output values based on the inputs. Typically 80% of the data is used for training, with 20% reserved for testing in the next step.
8. Evaluate the accuracy of the surrogate model by comparing the results of the detailed simulation and the surrogate for separate testing data.
9. Perform Hyper-parameter optimization (HPO), in which different configuration parameters of the machine learning algorithm are trialled to find values that give good model accuracy. Typically 60% of the data is used for training each model variant and 20% for validating performance of those hyper-parameters, with the final 20% remaining for testing after the optimization is complete (this portion being analogous to Step 8).

Case study details

This paper uses as the base model (step 1) a variation of the Department of Energy Medium Office archetype model (Deru et al., 2011). The weather file used was for Toronto, Canada. The 39 model parameters (step 2) are given in Table 1.

Table 1: Parameters of the surrogate model with ranges.

<table>
<thead>
<tr>
<th>Parameter (Unit)</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>North Window to Wall Ratio (%)</td>
<td>0.001-0.99</td>
</tr>
<tr>
<td>East Window to Wall Ratio (%)</td>
<td>0.001-0.99</td>
</tr>
<tr>
<td>South Window to Wall Ratio (%)</td>
<td>0.001-0.99</td>
</tr>
<tr>
<td>West Window to Wall Ratio (%)</td>
<td>0.001-0.99</td>
</tr>
<tr>
<td>North Shading (m)</td>
<td>0.001-2</td>
</tr>
<tr>
<td>East Shading (m)</td>
<td>0.001-2</td>
</tr>
<tr>
<td>South Shading (m)</td>
<td>0.001-2</td>
</tr>
<tr>
<td>West Shading (m)</td>
<td>0.001-2</td>
</tr>
<tr>
<td>Orientation (degrees from north)</td>
<td>-90-90</td>
</tr>
<tr>
<td>Occupancy (m2/occ)</td>
<td>1-250</td>
</tr>
<tr>
<td>Heating Setpoint (C)</td>
<td>5-28</td>
</tr>
<tr>
<td>Cooling Setpoint (C)</td>
<td>13-50</td>
</tr>
<tr>
<td>RH-Low Setpoint (%RH)</td>
<td>0-80</td>
</tr>
<tr>
<td>RH-High Setpoint (%RH)</td>
<td>30-100</td>
</tr>
<tr>
<td>Ventilation Per Occupancy (L/s-occ)</td>
<td>0-25</td>
</tr>
</tbody>
</table>
The outputs of the model (step 3) are:

- Thermal Energy Demand Intensity for Heating (TEDI-H), kWh/m²/a
- Thermal Energy Demand Intensity for Cooling (TEDI-C), kWh/m²/a
- Total Energy Use Intensity (TEUI), kWh/m²/a

The model was fitted to 10,000 samples (step 4). The impact of this value was investigated, see section on sample set size.

**Multiple linear regression models**

Multiple linear regression (MLR) can be considered a basic form of a surrogate model and is therefore used in this study as the preliminary baseline for comparison for more complex implementations that follow. Providing a meaningful frame of reference for the performance of surrogate models is important. For example, the R² value of a specific surrogate model reflects a combination of the model performance and the complexity of the problem space it is trying to unravel. The linear baseline models can help evaluate the potential benefits of using more complex models.

The baseline model is a linear regression model using the ordinary least squares method for training. Even linear models can be susceptible to over-fitting when many parameters are included. Over-fitting is especially a problem in machine learning when there may be parameters that are not significant drivers of the output, but may exhibit unintended correlations and influence the MLR coefficients. Parameter selection for surrogate modelling benefits from being expert-driven; however, modellers may not have a perfect understanding of which parameters might be the most influential, nor how influential they are to each output. Overfitting can also occur from sampling limitations of the design space. Regularization is a technique which often used to improve the performance of machine learning models. It adds additional terms to the objective function in an attempt to smooth the output to prevent overfitting. Common methods include LASSO (or L1) regularization, which penalizes the magnitudes of coefficients (or weights), and Ridge (or L2) regularization, which penalizes the square of coefficient magnitudes. Both encourage lower-order (smoother) relationships, and ElasticNet (Zou & Hastie, 2005) is a method that combines both of these approaches.

Various regularization hyper-parameter combinations were tested for linear models, but only a subset are selected for presentation and comparison. In addition to the ordinary linear model (L-A) we have included the highest (L-B) and lowest performing (L-C) regularized linear models.

**Neural network formulations**

Three neural networks are employed for this study. A summary of their characteristics is given in Table 2. The first neural network (NN-A) is based on the proven architecture for surrogate modelling and is borrowed from (Paul Westermann et al., 2021). This neural network was used to develop the first iteration of neural networks for the preliminary Net-Zero Navigator platform, and with 35 inputs and 12 outputs had an average R² score of 0.986 in that study. A second proven surrogate model (NN-B) is based on (Cant & Evins, 2022), which had 23 inputs, 108 outputs, and an average R² score of 0.98 across the outputs in that study. The third neural network (NN-C) is an optimal configuration found using hyper-parameter optimization (see next section).

All of the neural networks were trained using the Adam optimizer (Kingma & Ba, 2017) with a learning rate of 0.01, a decay rate of 0.96, and a batch size of 256. The NNs were trained for 400 epochs on standardized inputs and box-cox transformed outputs. The hyper-parameters, including NN architecture, associated with each of the three neural networks is presented in Table 2. All three of the neural networks used overfitting conditioning, some combination of L1 or L2 regularization, batch normalization or dropout during training. All of the networks were relatively shallow, however the width varied by an order of magnitude. In addition, the non-linear activation functions varied across the neural networks. Further research is required to determine if there are any correlations between combinations of hyper-parameters and model performance, or correlations between problem space characteristics and optimal hyper-parameter selections.

**Table 2: Hyper-parameters of the neural networks.**

<table>
<thead>
<tr>
<th>Hidden Layers</th>
<th>NN-A</th>
<th>NN-B</th>
<th>NN-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes per Layer</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Input and Output Transformation

Both the inputs and the outputs of the sample set data should be scaled before the machine learning model is fitted (step 6 in the process above). The same transformation is applied when user-provided input values are transformed into model inputs for new predictions, which are model outputs transformed back into real units for display. Standardization refers to transformations that centre a distribution about the mean with a unit standard deviation. Normalization involves scaling the data to a fixed range, typically between 0 and 1. Scaling should be determined for the training data set only, with this transformation applied to the testing data, to avoid data leakage in which the testing set influences scaling and hence training.

Scaling of inputs can be important where different units and ranges are present that do not reflect differences in relative importance. Scaling of outputs is important for the model to provide appropriate accuracy at all output levels; if left unscaled, errors for large values would be disproportionately penalised. This is particularly relevant for our surrogate modelling application, where we are interested in predicting energy use for very high-performance designs. Both standardization and normalization are typically linear operations that preserve scedasticity in the data – regression techniques often benefit from (or assume) normal distributions and homoscedasticity. Figure 1(a) shows the output distributions for the dataset, indicating significant positive skew (where the median is less than the mean), and a long tail of outliers. This also aligns with intuition about building energy demands, which must be strictly positive. These characteristics may compromise both the effectiveness of the statistical methods employed and the regression accuracy in the low-energy regions of interest.

In this work we explore other transformation methods that we have found to have a big impact on model performance (see results section). Specifically, we applied the Box-Cox transformation (Box & Cox, 1964), which transforms each variable into an approximate normal distribution (Equation 1). This requires an appropriate parameter \( \lambda \) to be determined. Based on initial tests, \( \lambda \) values for TEDI-H, TEDI-C and TEUI of 0.16, 0.08 and 0.28 were selected (where \( \lambda = 0 \) would approach a logarithmic function and \( \lambda = 0.5 \) a square-root function). The results of this transformation are illustrated in Figure 1(b). Comparing the absolute values for Pearson’s Coefficient of Skewness and kurtosis for the real output data and the transformed data shows the latter much more closely aligns with a normal distribution (reduction in positive skew and frequency of outliers).

\[
y(\lambda) = \begin{cases} 
  \frac{y^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\
  \log y & \text{if } \lambda = 0 
\end{cases} 
\]

Hyper-parameter optimization

The hyper-parameter optimization (HPO) process (step 9 in the process outlined above) can be very important in training a high-performing neural network. HPO was conducted using the Keras Tuner library. This allows hyper-parameter values to be specified using placeholder variables that are then assigned values as part of an optimization process to find high-performing combinations. For each set of hyper-parameters being trialled, the model is trained and evaluated on a validation set; this is distinct from the testing set used for final evaluation of the optimized model. An optimization algorithm is needed to find high-performing hyper-parameters more rapidly than exhaustive search; here we use the Bayesian Optimization tuner from the Keras Tuner library, which implements an optimizer based on Gaussian processes.

Hyper-parameters to be optimized are given in Table 3; we will briefly explain their function. Network depth and width of each hidden layer define the dimensions of the neural network. Activation function selects the way each node processes values; ReLU (Rectified Linear Unit) returns 0 for input values below 0 and linearly returns the input for values above 0, while tanh (hyperbolic tangent) returns -1 for input values below 0 and +1 for values above 0. Regularization type and value defines the additional objective function terms used to prevent overfitting as discussed above. Dropout and dropout rate determine whether random portions of the network are removed at each epoch; this has been shown to improve the robustness of the model. Learning rate determines how much the weights are updated in each epoch; we use the Adam optimizer which uses an adaptive learning rate.
determined by the initial learning rate, decay rate and decay step size (Kingma & Ba, 2017). Batch size determines how the training data is subdivided for training, with each randomly chosen batch used for one epoch; it should be noted that the choice of batch size is strongly related to the sample set size used.

Results

Figure 2 shows the overall performance of the NN-C model in predicting TEDI-Heating for the test data set.

![Figure 2: Scatter plot of actual values from EnergyPlus against predicted values from the surrogate model for TEDI-Heating. The insert shows the accuracy and histogram profile of actual data for the first quartile.](image)

These results illustrate the skewed nature of the dataset, with sparse outliers of large magnitude and a large number of samples in the low-energy region. Performance is weaker in the lower region; however, this has been improved from previous approaches through the refinements described in the following sections.

Comparison of linear models and neural networks

The performance of the three linear models and the three neural networks are presented in Figure 3 using $R^2$ and root mean square error (RMSE) as the metrics of consideration.

There is wide variation between the highest and lowest performing regularized linear models, whereas there is a minimal difference between the highest performing linear model and the ordinary linear model. The highest performing regularized MLR model employed minimal weighting on the regularization coefficients and so performance similar to the ordinary MLR is expected.

The significant weighting on the regularization coefficients associated with the lowest performing MLR model compared to the others indicates that the problem is likely not associated with significant overfitting in the training set, and that including only the most significant factors can capture approximately 50% of the variation within the dataset. Furthermore, the $R^2$ values above 0.6 for TEDI and CEDI from the ordinary MLR indicate a somewhat linear problem set, with 40% of the variation not able to be captured by a linear correlation assumption between inputs and outputs.

The three neural networks all perform better than any of the linear models, which can be expected due to the complex nature of heating and cooling loads in building energy systems. Some notable complex and interactive effects include the impacts associated with solar heat gain coefficient and window to wall ratio; the characteristics of a window can have an inverted impact on the TEDI as the relative amount of window increases. In addition, there are cases where a minima or maxima is experienced with certain combinations, from which both increasing or decreasing a quantity will increase the TEDI.

The optimized neural network (NN-C) appears to perform very well in both TEDI-H and TEDI-C predictions, with $R^2$ values above 0.94 for both outputs. The un-tuned networks (NN-A and NN-B) from previous surrogate modelling projects both performed significantly better than the best performing linear models and reduced RMSE metrics by approximately 50%. The results indicate that off-the-shelf neural network architectures for surrogate models are likely to result in acceptable performance, assuming similar problem spaces, such as the continuous and/or discrete design input variables and continuous aggregated (monthly or annual) outputs. Hyper-parameter optimization does, however, provide an accuracy boost and is generally a minor additional computational burden compared to that of generating a suitable training and testing dataset.

![Figure 3: $R^2$ and RMSE metrics for TEDI and TEUI for three linear models (ordinary L-A, best performing L-B and worst performing L-C) and three neural network formulations (NN-A, NN-B, NN-C; see Table 2).](image)
Examining linear model weights

It is straightforward to fit a linear model using the surrogate modelling workflow by simply switching the model form to be fitted. (A multiple-linear model is effectively a neural network with one layer and a linear activation function). Figure 4 shows the parameter weights of some of the linear models. These weights can provide a form of sensitivity analysis, in that they represent the change in the output variable for a given change in that input.

Parameters are sorted by the weights for model L-B (the best-performing model that used elastic net regularization) for the output TEDI-Heating. In general this order makes sense, with set points, infiltration, ventilation, energy recovery, internal gains and glazing U-value all having weights above 0.1. For comparison, the sum of all weights is 5.81, so all parameters with weights lower than 0.05 can only have a 1% impact on the output.

Other parameter weights are shown for model L-A (ordinary linear) for TEDI-Heating and L-B for TEDI-Cooling and TEUI. Usually the weights are very similar, with a small number of exceptions. There are minor differences in weights for model L-A, showing where regularization had an impact (especially towards low-value weights). Some parameters have much higher weights (shading and glazing U-value for TEDI-Cooling; ventilation and daylighting for TEUI), while others were much lower (Lower RH and heating set points for cooling; most of the important parameters for TEDI-Heating for TEUI).

It should be noted that using a linear model to judge the importance of parameters has significant limitations. Parameters which have combined behaviours resulting in non-linear responses will not be identified. There is therefore a danger in missing important variables if the linear importance is used to cull parameters. For this study, we continue to use the full set of 39 parameters.

Impact of sample set size

To investigate the impact of sample set size on model accuracy, we began with a very large sample set of 100,000 runs, then sub-sampled this to get progressively smaller sample sets. Figure 5 shows model accuracy for TEDI and TEUI, showing the error metrics (a) $R^2$ and (b) RMSE.

![Figure 5: Sample set size comparison for model outputs TEDI and TEUI, showing the error metrics (a) $R^2$ and (b) RMSE](https://doi.org/10.26868/25222708.2023.1394)
performance than 5,000 this is likely an artefact of the random selection of the sub-sample sets. Further work should investigate the robustness of model accuracy to random sampling, i.e. whether there are certain sets of 5,000 samples that perform notably worse. In this work we use 10,000 samples for all other models investigated.

Impact of Standardization

The impact of applying different transformations to the input and output data prior to training the surrogates was explored in stages. Figure 6 shows the model accuracy with different transformation applied to the output (the inputs maintain basic standardization). The Box-Cox implementation applies $\lambda$ values determined through automatic tuning (0.16, 0.08 and 0.28 for TEDI-H, TEDI-C and TEUI respectively). The ‘custom’ transformation is an implementation of Box-Cox with $\lambda$ equal to -0.2, to test the impact of further transformation beyond normality. Surrogate performance was greatest with the Box-Cox transformation, with good performance for basic Standardization techniques and no scaling at all. The latter could be attributed to the similarity among the scale and type of outputs, but further investigation is warranted.

It was found that the transformations that preserve the characteristics of the distributions (e.g. normalization ‘MinMax’, or Standardization) have similar performance calculated using both scaled data (as used for training the surrogate) and corresponding unscaled data (real units); however, Box-Cox shows a loss of accuracy due to the inverse transformation back into real data.

Conversely, the surrogates trained using Box-Cox show stronger performance for accuracy metrics that are proportional to the magnitude of the outputs (e.g. Mean Absolute Percentage Error), which warrants further investigation.

While some standardization or normalization of input variables (which were sampled as uniform distributions) was found to be necessary for good surrogate accuracy, similar trends with power transformations were not found as seen with the outputs.

Hyper-parameter optimization

Figure 7 shows the results of the hyper-parameter optimization (HPO) process. The optimizer evaluated 100 combinations of the hyper-parameters in Table 3; the worst-performing three are not shown (MSE scores of 4, 6 and 36). There are some notable trends: depth is 2 for high-performing networks; width of layers 1 and 2 is 90 for the best-performing networks. However, perhaps the most notable aspect of the results is that there are exceptions to all other trends. Dropout is rarely used in high-performing networks, but there are good combinations that do use it. There is no clear trend between ReLU and tanh activation functions, or between L1 and L2 regularization. Other hyper-parameters not shown (initial learning rate, learning decay rate, learning decay step size, batch size, regularization value) similarly show possible trends but with many exceptions. Widths for deeper layers are rarely used (only 11 out of 97 have more than 2 layers) also tend to be wide (average 56 nodes).

It is worth examining the changes in parameter values among the best-performing combinations. The small change in score of the top two (MSE = 0.06, 0.07) differs only in the width parameters of layers which are not active, and comes from the stochastic nature of the training process. In addition to the parameters shown, they have learning rate 0.1, decay rate 0.99, decay step size 10 and batch size 400. The third network (MSE = 0.09) has batch size 200 and activation function ReLU. The next (MSE = 0.12) uses dropout with a rate of 20%. While no firm conclusions can be drawn without further investigation of the robustness of the HPO process, these provide interesting insights into the hyper-parameter choices that are present in all high-performing combinations and those for which the choice appears to be less critical.

Conclusions

This paper has demonstrated the importance of characterizing domain-specific inputs and outputs, using basic multiple linear regression to establish a useful frame of reference for the more sophisticated surrogate model training. The study also shows the role transformation of those variables has in supporting more effective regression analysis, better alignment with domain-specific regions of interest (in this case, buildings approaching net zero energy). Power transformation like Box-Cox can both improve surrogate accuracy, while better reflecting these objectives.
The value of Hyper-Parameter Optimization (HPO) in applied surrogate training is shown, especially as part of an iterative refinement process including consideration of sample sizes. This work suggests further investigation is warranted into expanding the HPO experimentation, including the use of alternative accuracy metrics and loss functions that better reflect the domain-specific objectives.

**Future Work**

Model interpretability could be extended from the pre-training stages (with simple linear models) to model-agnostic methods that directly use the trained surrogates. This work suggests that multiple techniques for refining the hyperparameter space can be compatible and improve overall optimization of the surrogate configuration for training. This benefit could be further explored, such as in comparison to more traditional methods.

**Acknowledgements**

The Net-Zero Navigator project was funded by Natural Resources Canada. The BESOS platform was funded by CANARIE. Related work has been funded by The Natural Sciences and Engineering Research Council of Canada.

**References**


