End-to-end deep metamodeling to calibrate and optimize energy loads

Max Cohen, Maurice Charbit, Sylvain Le Corff, Marius Preda, Gilles Nozière

End-to-end deep metamodeling to calibrate and optimize energy loads

Max Cohen[†], Maurice Charbit[‡], Sylvain Le Corff[†], Marius Preda[⋆], and Gilles Nozière[‡]

[†]Samovar, Télécom SudParis, Département CITI, TIPIC, Institut Polytechnique de Paris.
[⋆]Samovar, Télécom SudParis, Département ARTEMIS, ARMEDIA, Institut Polytechnique de Paris.
[‡]Oze-Énergies.

Abstract

In this paper, we propose a new end-to-end methodology to optimize the energy performance and the comfort, air quality and hygiene of large buildings. A metamodel based on a Transformer network is introduced and trained using a dataset sampled with a simulation program. Then, a few physical parameters and the building management system settings of this metamodel are calibrated using the CMA-ES optimization algorithm and real data obtained from sensors. Finally, the optimal settings to minimize the energy loads while maintaining a target thermal comfort and air quality are obtained using a multi-objective optimization procedure. The numerical experiments illustrate how this metamodel ensures a significant gain in energy efficiency while being computationally much more appealing than models requiring a huge number of physical parameters to be estimated.

1 Introduction

Global energy demand for heating, ventilation and air-conditioning in commercial or public buildings has been increasing rapidly for the past few decades along with population and economic growth. This rising demand is at the root of the complex problem of simultaneously ensuring a better environmental impact, as higher consumption of fossil fuels implies higher greenhouse gas emissions, while maintaining a satisfactory comfort in buildings (air and indoor temperature quality). In that respect, building designing and management have to integrate thermal performance and comfort criteria, and to assess the environmental consequences of any chosen policy. This makes the analysis of building energy performance a challenging multi-criteria problem, as detailed for instance in [Bre et al., 2016]. Our paper sets the focus on analyzing and optimizing cooling, heating and air conditioning loads by tuning the building management system in given buildings without costly, invasive or time consuming renovation works. The aim is to provide the optimal building management settings, governing Heating, Ventilation and Air-Conditioning (HVAC) and Air Handling Units (AHU), in order to improve thermal comfort, energy loads as well as environmental impact. This objective is decomposed into three steps: (i) provide a model to predict future energy loads and temperature in a building based on the HVAC system and the weather forecast,

(ii) calibrate the parameters of this model based on real data obtained in real time in each building and (iii) optimize the HVAC equipment to minimize the total energy load in future periods while maintaining a given thermal comfort.

The first category of approaches to model the energy performance of a building are based on physical equations that describe heat transfer between the building and its environment. Thanks to their increasing reliability, simulation based methods such as EnergyPlus, TRNSYS or DOE-2 are commonly used to simulate the system behavior based on a schematic view of the building. EnergyPlus was used for instance in [Shabunko et al., 2018] to build three types of typical designs and to benchmark the energy performance of 400 residential buildings. In [Zhao et al., 2016], the authors proposed a predictive control framework based on Matlab and EnergyPlus in order to optimize energy consumptions while meeting the individual thermal comfort preference. In these papers, a schematic building is used in the simulation program and considered as a baseline for energy loads. These approaches rely on a huge number of parameters, such as window to wall ratio, window leakage, or wall construction. Instead of costly campaigns to measure these parameters, that would have to be reiterated for each new building, they may be estimated using an automatic calibration procedure by minimizing a cost function which associates, with each set of parameters, the discrepancy between the true energy loads and temperatures, and the simulated ones, see [Coakley et al., 2014, Le Corff et al., 2018]. As shown in [Nagpal et al., 2018], calibration yields sufficiently accurate results for a variety of different buildings, thus ensuring limited additional costs to generalize a given model. Once calibrated, the optimisation task consists in determining a set of building management settings that will result in lower energy consumption, while preserving comfort. Following numerous works such as [Bre et al., 2020], the multi-objective Non-dominated Sorting Genetic Algorithm-II (NSGA-II), see [Deb et al., 2000], is the most widespread method to solve the optimization task. However, when no prior knowledge is available on the thousands of specific parameters required to specify each building, calibrating and optimizing such simulation programs is computationally prohibitive, see [Westermann and Evins, 2019]. This shortcoming is particularly severe in cases where many data are available from numerous wireless sensors installed in a building but no intrusive and resource consuming in-site campaigns are deployed to fix the values of the physical parameters.

Metamodeling approaches aim at overcoming this computational cost by proposing surrogate models that replace the physical simulator during calibration and optimization tasks. The parameters of such metamodels are estimated during a training phase using simulations conducted by a physical-based model, that aims at exhaustively capturing the building behavior for various building management settings. In [Bre et al., 2020, Reynolds et al., 2018], statistical models are trained on a dataset sampled from EnergyPlus, allowing significant computational savings during optimization. In [Bre et al., 2020], the authors proposed to combine NSGA-II with an artificial neural network metamodel to obtain a Pareto front of optimal HVAC parameters with the trained metamodel, in order to optimize the consumption of a $83\,\mathrm{m}^2$ house. To fit this dataset, instead of standard statistical models, this paper uses a Feed Forward Neural Network (FFN) as a metamodel. Although they often yield very accurate predictions, these neural networks are not adapted to time series problems, and are usually substituted for there sequential counter parts, such as recurrent or convolutional based approaches. This FFN was only validated with EnergyPlus simulations, and the calibration step was not performed, as no real historic data from the targeted building were discussed. Similarly, [Reynolds et al., 2018] proposed a FFN based metamodeling approach to reduce up to 25% the energy consumption in a small office building. EnergyPlus was used to sample a dataset for various zones of the building, in order to train zone level metamodels. These

simulation spread over 24 hours, and approximated the building behavior from January to March. Once again, in the absence of real historic data, no calibration step was implemented. The first optimization method is similar to previous works, and consists in optimizing consumption by feeding NSGA-II each metamodel. In a novel approach, optimization can also be updated every hour with the newly collected data from the building, in hope to avoid the error drift of simulating 24 hours of building behavior without any feedback. The study presented in [Magnier and Haghighat, 2010] focused on the optimization of a $210\,\mathrm{m}^2$ two storey house. Despite measured data being available, no automatic calibration step was discussed ; instead, TRNSYS was calibrated by hand to match the real building, resulting in relative errors of 3.7%, 3.4%, and 7.3% for heating, cooling, and fan monthly energy consumption, respectively. A FFN surrogate model was fit on a dataset of 450 samples, before being optimized using once again NSGA-II.

In this paper, we propose a end-to-end methodology, from dataset sampling to metamodel calibration and optimization using data obtained from wireless sensors set in a large building. The proposed metamodels involve classical recurrent neural networks and an approach based on a Transformer architecture ([Vaswani et al., 2017]) which has recently proven both an accurate and computing efficient alternative to traditional sequences to sequences models, such as Long Short-Term Memory (LSTM, [Hochreiter and Schmidhuber, 1997]) and Gated Recurrent Unit (GRU, [Cho et al., 2014]). Transformers combine an encoder-decoder architecture, see for instance [Cho et al., 2014] or [Bahdanau et al., 2015], allowing the model to learn semantic information from the observations using attention mechanisms ([Parikh et al., 2016, Zhu et al., 2019]) that could be interpreted as the day to day patterns of our problem. Once the metadomel is trained using a dataset built using TRNSYS, all the parameters of a real building and of its Building Management System (BMS) are estimated using real measurements with the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [Igel et al., 2007] which provides a derivative free optimization procedure. A multi-objective methodology to improve energy efficiency and maintain thermal comfort is then implemented by acting only on the BMS. The NSGA-II approach is used to obtain the Pareto optimal parameters. The performance of this metamodel are compared at each step with the usual FFN alternatives, LSTM, and GRU metamodels.

The paper is organized as follows. Section 2 provides all the deep learning architectures used in this paper to build a metamodel and describes the data and variables used in our metamodel. Section 3 illustrates the performance of our metamodel in the calibration and optimization process of a real building. The numerical experiments illustrate how this metamodel ensures a significant gain in energy in comparison to the considered alternatives.

2 Metamodeling

2.1 Notations

Let $(X_k)_{k\geqslant 0}$ be the state of the building i.e. the inside temperatures and the consumptions of the building management system. The index k denotes time and, in the setting of this paper, data are collected each hour. The aim of the metamodel introduced in this paper is to provide a numerically efficient solution to predict $(X_k)_{k\geqslant 0}$ from other variables and external observations such as meteorological data. Such a metamodel is described by several sets of input variables. A parameter θ_{build} containing all unknown parameters useful for the geometrical description of the buildings (windows area ratio, etc.) and parameters related to heat transfer (capacitance, airchange infiltration, etc.). Choosing such parameters allows to build a data set and design a

metamodel able to mimick various buildings. A sequence $(W_k, O_k, I_k)_{k\geqslant 0}$ providing at each hour the building management system variables in I_k (comfort and reduced temperatures for the HVAC), the occupancy O_k (described as a percentage of a given maximum number of people) and in W_k the weather data at time k. In this section, we describe how a simulation program may be used to train the metamodel which aims at mimicking the outputs of this simulation program for various choices of θ_{build}, $(I_k)_{k\geqslant 0}$, $(O_k)_{k\geqslant 0}$ and of meteorological data $(W_k)_{k\geqslant 0}$. The appendix displays a complete list is of the variables contained in θ_{build}, $(I_k)_{k\geqslant 0}$, $(O_k)_{k\geqslant 0}$ and in $(W_k)_{k\geqslant 0}$ for the numerical experiment of this paper.

2.2 Models

In most recent works, a great deal of research activities focused on FFN as surrogate models, [Bre et al., 2020, Magnier and Haghighat, 2010, Reynolds et al., 2018]. Although they may lead to interesting performance during the training phase, these fully connected architectures are not well suited for time series prediction, in particular for long time spans. We ceased this opportunity to explore other approaches that have proven to be more relevant for solving time series tasks in the past few years. Therefore, we decided to evaluate the go-to architectures for time series: a standard LSTM, a bidirectional GRU (BiGRU), a hybrid model mixing both convolutional and GRU layers (ConvGru), and a Feed Forward Network (FFN) as used in previous works. In addition to those models, a Transformer model which introduces an attention mechanism to model dependencies is also considered. These models have been implemented using the deep learning framework PyTorch and can be found on our Github[1].

Recurrent Neural Network (RNN) were first introduced as a more suited architecture for dealing with time varying input patterns [Mozer, 1989]. By replacing buffer based approaches with an updated context state, RNN are able to solve time series problems with short time dependencies, but are lackluster in problems requiring long term memory due to vanishing and exploding gradient [Bengio et al., 1994]. Long Short Term Memory proposed in [Hochreiter and Schmidhuber, 1997] aim at bridging that gap by enforcing error flow throughout time in the network. Later, the authors of [Cho et al., 2014] modified the LSTM architecture in order to simplify implementation and improve computation times, resulting in a novel model called Gated Recurrent Unit.

In parallel to these advances on recurrent architectures, Convolutional Neural Networks (CNN), rendered popular by [Krizhevsky et al., 2012] for image classification, have been adapted to time series problem. The approaches proposed in [Józefowicz et al., 2016, Kim et al., 2016] outperformed traditional Natural Language Processing (NLP) models by replacing the embedding layer with a character-level convolutional layer. Following this idea, [van den Oord et al., 2016] considerably improved on the speech to text state of the art, by using dilated convolutions, increasing the receptive fields of WaveNet at each layer. One year later, [van den Oord et al., 2018] improved on the existing architecture by introducing Parallel WaveNet, which provided similar performance for a lower computational cost.

Recurrent and convolutional approaches coincide in that temporally close time steps data are matched together. In 2017, [Vaswani et al., 2017] proposed an attention based approach to solving NLP tasks that consider the entire input sequence in parallel. The Transformer model is based on a self-attention mechanism, that computes an attention value for every element of a sequence with respect to all others to model their dependency. This attention mechanism allows to understand at each time step k which input elements are crucial to predicting the new state X_k. This makes these

[1] https://pytorch.org and https://github.com/maxjcohen/transformer

networks more interpretable than their most widely-used recurrent counterparts such as LSTM or GRU networks and motivate a keen interest for such approach to predict complex time series.

Transformer differ from sequential architectures in that they compute prediction at each time step in parallel. In our context, we propose to use a Transformer architecture as follows. Let $F_{\theta_{\text{meta}}}$ be the Transformer mapping which computes a prediction $(\widehat{X}_k)_{1\leqslant k\leqslant n}$ of the states $(X_k)_{1\leqslant k\leqslant n}$:

$$(\widehat{X}_1,\dots,\widehat{X}_n) = F_{\theta_{\text{meta}}}(\theta_{\text{build}},(W_k,O_k,I_k)_{1\leqslant k\leqslant n})\,,$$

where θ_{meta} contains all the unknown parameters specific to the metamodel. Following the state of the art in sequence to sequence modeling, the Transformer adopts an encoder-decoder architecture. The encoder computes a latent vector from the input data, which is fed to the decoder in order to predict the outputs. These sub-networks are trained jointly and are supposed to foster learning of a meaningful representation of the data. The encoder and decoder consist of a self attention block, responsible for leveraging the relationship between time steps in the sequence, and a feed forward network, which contains the non linearity of the Transformer.

Embedding. Similarly to the original embedding layer, our metamodel is first based on a linear map, that allows setting the dimension d_{emb} of the latent representation of the inputs. Let $\Delta > 0$ be an attention window and k be a given time step. For all $k-\Delta \leqslant j \leqslant k+\Delta$, let $U_j = (\theta_{\text{build}}, I_j, W_j, O_j) \in \mathbb{R}^d$ be the vector at time j which stacks all inputs, and U_j^{emb} the latent vector for the corresponding time step,

$$U_j^{\text{emb}} = W_{\text{emb}}\cdot U_j + b_{\text{emb}}\,,$$

where $W_{\text{emb}} \in \mathbb{R}^{d_{\text{emb}}\times d}$ and $b_{\text{emb}} \in \mathbb{R}^{d_{\text{emb}}}$ are the unknown weight matrix and bias respectively, that are estimated during the training phase.

Encoder. The encoder block proceeds by computing the query, key and value from R_j for this state with a linear transform:

$$q_j = W^q U_j^{\text{emb}}\,,\quad \kappa_j = W^\kappa U_j^{\text{emb}}\,,\quad v_j = W^v U_j^{\text{emb}}\,, \tag{1}$$

where W^q, W^κ and W^v are the unknown $r\times d_{\text{emb}}$ matrices (parameters of the metamodel to be estimated, r chosen by the user). Then, let K_k denote the matrix whose columns are κ_j, $k-\Delta\leqslant j\leqslant k+\Delta$, and compute for all $k-\Delta\leqslant j\leqslant k+\Delta$,

$$s_k^j = q_j^T K_k \quad\text{and}\quad \pi_k^j = \sigma(s_k/\sqrt{r})_j\,,$$

where σ is the softmax function. Finally, self-attention is computed as

$$z_k^{\mathsf{enc}} = \sum_{\ell=k-\Delta}^{k+\Delta} \pi_k^\ell v_\ell\,. \tag{2}$$

The output of the encoder is then given by a final transform of z_k^{enc} which is considered as the input of a FFN:

$$r_k^{\mathsf{lat}} = \texttt{FFN}_{\theta_{\text{att}}}(z_k^{\mathsf{enc}})\,,$$

where $\theta_{\text{att}} = \{W_1, b_1, W_2, b_2\}$ and

$$\texttt{FFN}_{\theta_{\text{att}}}(z) = W_2 \cdot max(0, W_1 \cdot z + b_1) + b_2\,.$$

In practice, the self attention computation (2) is replicated h times, each referred to as "head", that are concatenated before being fed to the FFN. Having multiple heads, i.e. computing multiple instances of self attention in parallel, allows the transformer to set attention to multiple aspect of the input sequence at the same time. In a multi-layer Transformer, the output of each layer is used as an input for the next layer before being processed similarly.

Decoder. The decoder block acts similarly, except for one added attention step where the keys and values are computed from the latent vectors r_j^{lat}, $k - \Delta \leqslant j \leqslant k + \Delta$. This produces a vector z_k^{enc} as in (2) which is a mixture of the values associated with the latent vectors. This mixture is fed to a FFN to produce $\widehat{X}_k$. The parameters to train are therefore W_{emb}, b_{emb}, θ_{att} and W^q, W^κ and W^v.

2.3 Training and validation

The first step consists in sampling a dataset with TRNSYS to learn the metamodel and defining ranges for each input parameters in θ_{build}, $(I_k)_{k\geqslant 0}$ and $(O_k)_{k\geqslant 0}$ with the help of energy managers, such as highest and lowest scheduled temperature, or the most early and late hour of arrival of occupants, see the appendix for a complete list of these ranges. In addition, real weather data $(W_k)_{k\geqslant 0}$ acquired between May and December 2019 where used to obtain a dataset consistent with the real building. As discussed in the previous section, some related papers use Latin Hypercube sampling, introduced in [McKay et al., 2000], to form their dataset. In our numerical experiments, we chose instead a uniform sampling method over the ranges of each variable. This allows us to easily split the dataset into k-folds, which will be useful for the validation step discussed in the next section.

During this step, daily values defined in the appendix are converted to a time series whose value changes with every day. This way, there are 38 variables in the input vector at each time step: 19 variables from θ_{build}, 7 from W_k, 1 from O_k and 11 from I_k. A total of 38000 training examples were sampled, an example being a week i.e. 168 hours. During the training phase, the parameters of each metamodel described in Section 2 are estimated based on this dataset (called θ_{meta} in the detailed case of the Transformer approach). The metamodels compared in this section are defined with a latent dimension of $d_{emb} = 64$ and a total of $N = 8$ layers. These values were obtained through a grid search, see the appendix for additional information. Other hyper parameters, such as learning rate dropout, number of epochs or batch size, were chosen empirically.

During training, for each example, we use a loss function defined by Energy Management experts, consisting of a combination between mean squared consumption and temperature errors:

$$\Delta_T^{\theta_{\text{meta}}} = \left(\frac{1}{N}\sum_{k=1}^{N}(\widehat{T}_k^{\theta_{\text{meta}}} - T_k)^2\right)^{1/2} \quad \text{and} \quad \Delta_Q^{\theta_{\text{meta}}} = \left(\frac{1}{N}\sum_{k=1}^{N}(\widehat{Q}_k^{\theta_{\text{meta}}} - Q_k)^2\right)^{1/2},$$

$$\text{loss}(\theta_{\text{meta}}) = \alpha \log(1 + \Delta_T^{\theta_{\text{meta}}}) + \beta \cdot \log(1 + \Delta_Q^{\theta_{\text{meta}}})\,,$$

where N is the number of data in each example, T_k and Q_k are the ground truth at time k, and $\widehat{T}_k^{\theta_{\text{meta}}}$ and $\widehat{Q}_k^{\theta_{\text{meta}}}$ are the predictions given by the metamodel with the current value θ_{meta} of the

Table 1: Metrics (means and standard deviations) of the metamodels on the validation dataset. The best mean values are displayed in bold (the lowest losses and mean squared errors and the coefficient of determination closest to 1).

		Transformer	BiGRU	LSTM	ConvGru	FFN
Loss	$(\times 10^{-4})$	**1.13 (0.746)**	1.43 (1.06)	13.8 (4.55)	2.78 (1.77)	61.1(27.4)
MSE_T	$(\times 10^{-5})$	**3.86 (4.53)**	4.28 (5.18)	7.32 (7.75)	9.37 (11.5)	178(205)
MSE_Q	$(\times 10^{-4})$	**2.47 (2.30)**	3.34 (2.98)	43.7 (14.7)	6.16 (4.30)	146(54.2)
$\text{MSE}_\text{T}^\text{occ}$	$(\times 10^{-5})$	**1.08 (1.32)**	1.18 (1.54)	2.02 (2.52)	2.77 (3.37)	51.2(64.1)
$\text{MSE}_\text{Q}^\text{occ}$	$(\times 10^{-4})$	**1.06 (1.29)**	1.21 (1.92)	3.61 (2.93)	2.28 (2.35)	43.2(25.1)
R_T^2	$(\times 10^{-3})$	**996 (0.832)**	996 (1.40)	992 (1.64)	990 (2.10)	829(43.2)
R_Q^2	$(\times 10^{-3})$	**760 (240)**	657 (593)	559 (473)	707(268)	−738(3080)

metamodel for temperature and consumption respectively. In this experiments below, we chose $\alpha = 1$ and $\beta = 0.3$. We chose the Adam optimizer [Kingma and Ba, 2014] ; all simulations were computed on a single 1080TI GPU card. Table 1 displays the mean values and standard deviations of the loss function on the validation dataset after training. The table also displays the mean squared error MSE_T (resp. MSE_Q) on the temperatures (resp. consumptions) only, and these metrics computed only during occupation time $\text{MSE}_\text{T}^\text{occ}$ and $\text{MSE}_\text{Q}^\text{occ}$. In addition, the coefficients of determination (rescaled mean squared errors relative to the target data) of the temperatures R_T^2 and consumptions R_Q^2 are given. These coefficients of determination are computed with the Python function *sklearn.metrics.r2_score*.

3 Energy Optimization in a real building

The experiments conducted in our paper to analyze the performance of the metamodel trained in Section 2.3 focused on the optimization of a $28733\,m^2$ building located in the Parisian region. The total building is represented by a single thermal zone including 5 vertical walls with respective following areas $3521\,m^2$, $2692\,m^2$, $3257\,m^2$, $599\,m^2$ and $16329\,m^2$, a horizontal roof and a horizontal ground. Based on a commonly used rule, it is assumed that 2/3 of the full area is occupied by people. Assuming that each occupant requires $12\,m^2$, this allows to set the initial values for the number of occupants and the number of PCs (set to 1.2 times this value) in the building during occupancy hours. These values are assumed to be known and fixed and used to sample the training dataset.

3.1 Calibration

During the training phase, metamodel parameters are estimated by minimizing the loss function on the simulated dataset which corresponds to various choices of θ_{build}, $(I_k, O_k, W_k)_{k\geqslant 0}$, associated with building behaviors $(X_k)_{k\geqslant 0}$. This metamodel has been trained on a dataset containing only simulated data, ignoring real building related noise and measurement errors. Additionally, both the BEM and our surrogate model take as input a number of variables, such as window to wall ratio, window leakage, or wall construction, that cannot be properly identified for each building. By comparing the metamodel predictions to real historic data during the calibration phase, we search for a set of building related parameters that best match reality.

Table 2: Metrics after calibration for two weeks, beginning the 4th and the 30th of November 2019. Calibration run for 500 epochs (resp. 2500 epochs) for the metamodel (resp. for TRNSYS).

	MSE_T	MSE_Q	$\text{MSE}_\text{T}^\text{occ}$	$\text{MSE}_\text{Q}^\text{occ}$	R_T^2	R_Q^2	time (h)
Week 1							
TRNSYS	$1.04 \cdot 10^{-1}$	4967	$3.31 \cdot 10^{-2}$	1434	0.644	0.848	2
Metamodel	$1.62 \cdot 10^{-2}$	3241	$4.71 \cdot 10^{-3}$	477	0.945	0.901	2
Week 2							
TRNSYS	$2.66 \cdot 10^{-1}$	16067	$6.58 \cdot 10^{-2}$	6782	0.592	0.761	2
Metamodel	$1.42 \cdot 10^{-1}$	10493	$6.55 \cdot 10^{-2}$	5162	0.782	0.844	2

During this step, the weights θ_meta of the metamodel are frozen, meaning that we no longer back propagate the error, nor do we update each weight matrix of the neural network. Using the coefficient of determination as a cost function, we can compute, for each given set of input parameters θ_build, $(I_k, O_k, W_k)_{k \geqslant 0}$, the difference between estimated and real historical data. Because this is a non differentiable problem, the cost function cannot be minimized using the same algorithm as in the training step; instead we use the CMA evolution strategy (CMA-ES, [Hansen, 2016]), an evolutionary algorithm adapted to derivative free non-convex optimisation problems in continuous domain. It is implemented by the author of the paper in the pycma library[2].

Calibration was run until convergence for the metamodel, and for a maximum of 8 hours for the original BEM (TRNSYS). We can see the advantage of going through the training of a metamodel when comparing a calibration for both TRNSYS and the metamodel, as we are now able to reach lower costs in a much shorter time frame. This is confirmed by Table 2 which displays the Mean squared error for the temperatures and heating consumption after calibration using TRNSYS and the Transformer-based metamodel, for two different weeks shown in Figure 1.

3.2 Optimization

Once the metamodel is calibrated, we can use it as an accurate simulator for how the building will react to changes in its usage. After a successful calibration, all building related variables contained in θ_build are correctly estimated. The parameters I_k associated with the HVAC system can be optimized for a given set of weather data W_k. The optimization tasks consists in finding a set a usage related parameters that reduce consumption while keeping the same level of comfort. Optimizing energy consumption requires minimizing two conflicting objectives, making it impossible to find a solution that optimize both objectives simultaneously. Instead, we search for optimal compromises between energy consumption and comfort, in the form of a Pareto front. Indeed, for any such optimal compromise, we can always get a higher level of comfort, for the price of a higher consumption. The consumption criteria is the energy load during the week ; the comfort criteria is the gap between indoor temperature and a constant reference temperature T^*:

$$\Delta_T^\text{opt} = \frac{1}{N_{Occ}^\text{opt}} \left(\sum_{k=1}^{N^\text{opt}} \mathbb{1}_{k \in \text{Occ}} (\widehat{T}_k - T^*)^2 \right)^{1/2} \quad \text{and} \quad \Delta_Q^\text{opt} = \frac{1}{N^\text{opt}} \sum_{k=1}^{N^\text{opt}} \widehat{Q}_k \,,$$

[2]https://github.com/CMA-ES/pycma

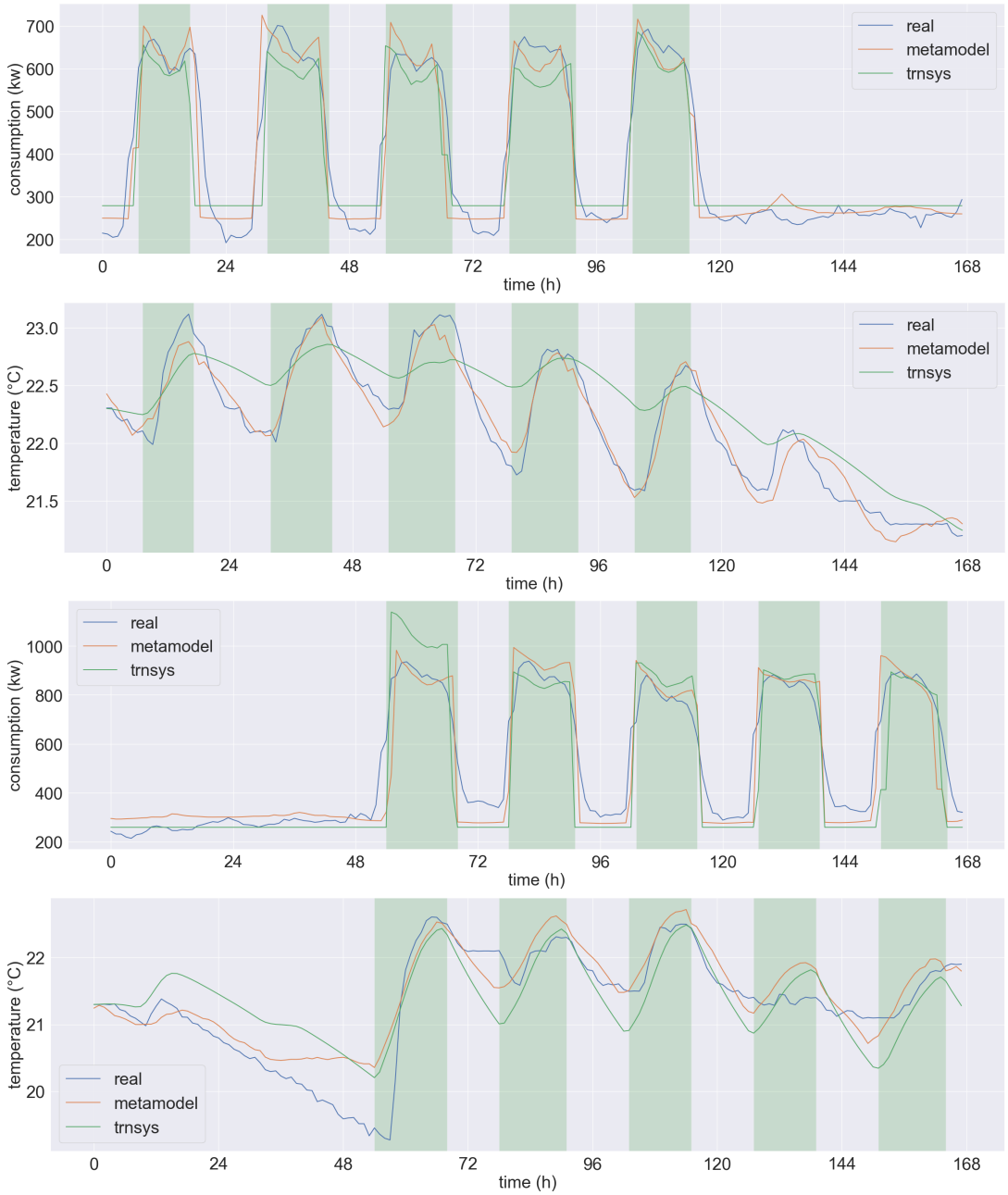

Figure 1: Consumption and temperature simulations after calibration, for both the metamodel and TRNSYS, for week 1 (top) and week 2 (bottom). Green bars indicate occupation periods.

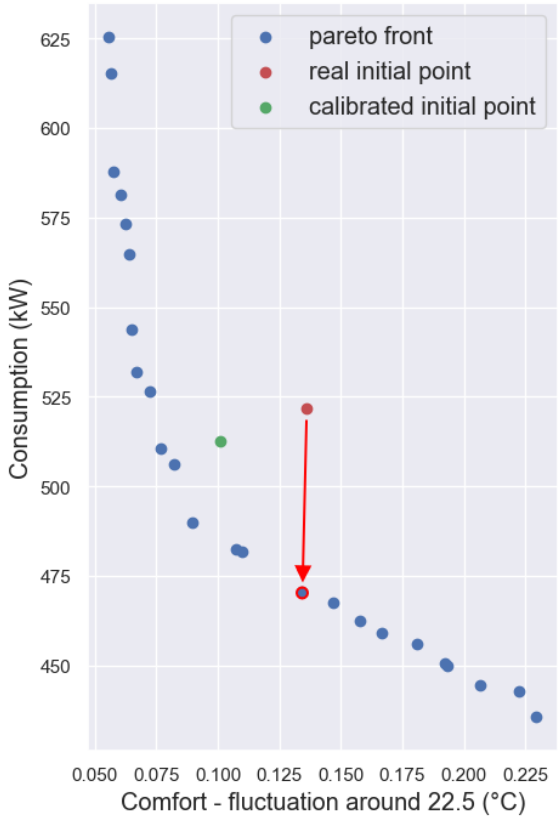

Figure 2: Pareto front after optimization for the second week. We select the point of closest equivalent comfort, corresponding to a 9.31% reduction in consumption.

where $T^* = 22.5°C$, N^{opt} is the number of hours to be considered in the optimization process and Occ is a subset of daytime hours specifying at which hours the target temperature has to be reached in the building. Following recent works in building energy optimization, we search for a set of optimal parameters using NSGA-II ([Deb et al., 2000]), another evolutionary algorithm, but adapted to multi objective problems. An implementation can be found in the Pygmo[3] library. In the absence of a stopping condition, we simply run the optimization for a set 3000 epochs (2 hours). The result can be viewed as a Pareto front which is given in Figure 2 for the second week used in the calibration process. As observed during calibration, this process can take a colossal number of epochs before achieving satisfactory results, once again justifying the use of a much faster metamodel. The time series of consumption and temperatures associated with the BMS parameters selected in Figure 2 are given in Figure 3.

[3]https://esa.github.io/pygmo2/

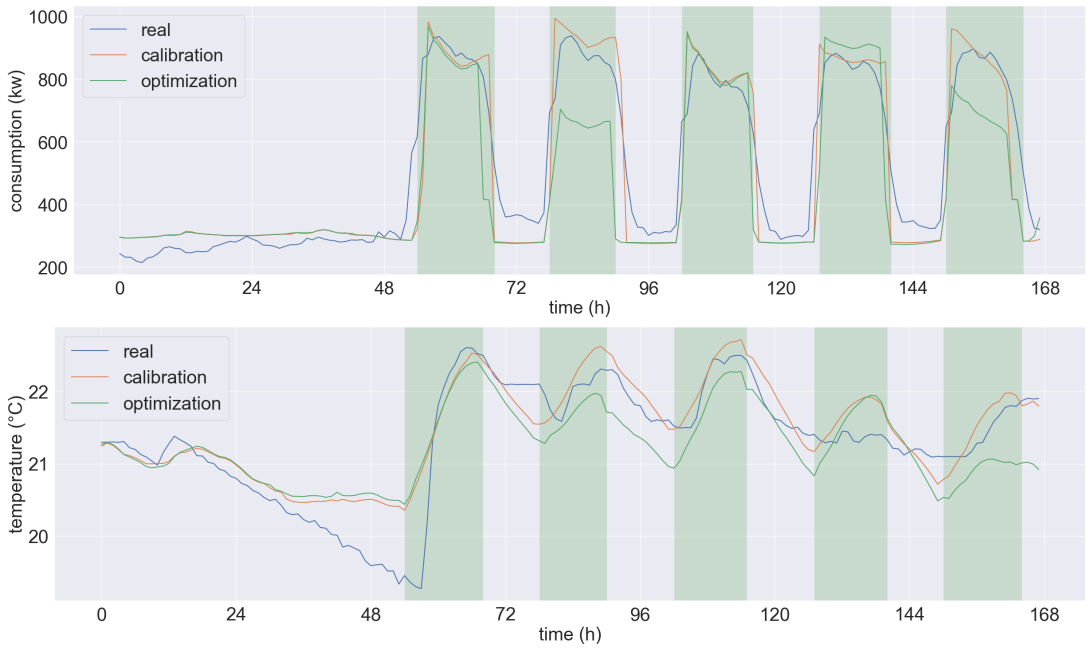

Figure 3: Consumption and temperature simulations after optimization (metamodel) for the second week. Green bars indicate occupation periods.

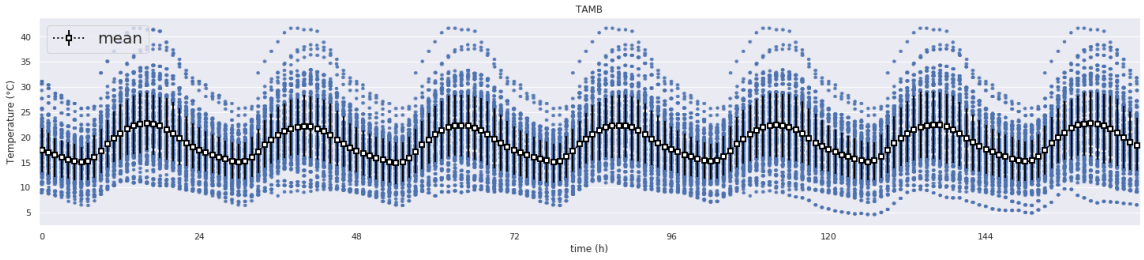

Figure 4: Distribution of the outdoor temperature in the dataset, stored in the `T_AMB` variable in the Table 6. Squares indicates the mean value, while vertical bars represent 85% of the data.

4 Conclusion

In this paper, we proposed an end-to-end metamodeling methodology to optimize building energy loads and to reduce computational costs. The proposed metamodel ensures compatibility between simulations and real building observations through a calibration step. We experimented with various deep learning architectures more suited to recurrent problems than Feed Forward Networks. Results show that a wide variety of models display encouraging results on our sampled dataset, while largely outperforming FFN. During optimization, we chose to maintain the same level of comfort as the historical data, in order to have as little impact as possible on the working environment. Compared to calibrated simulations, we were able to reduce consumptions significantly.

A Dataset

A.1 Inputs

The dataset is divided in four sub-variables, each representing a different aspect of the simulation. When creating the dataset using the Building Energy Model (here TRNSYS), we sampled each variable uniformly in a given interval. These intervals are also used for the calibration process.

- $\theta_{building}$ represents the geometric properties of the building, see Table 3.
- I_k stores the schedules and settings of the heating and ventilation, see Table 4. This is the only parameter tuned during the optimization process.
- O_k stores the occupation schedules of the building, see Table 5. These values are calibrated to match real data, and kept fixed during the optimization.
- W_k stores the weather data for the week, see Table 6. In this paper, we exclusively use data collected by weather institutes, that are available afterward. In practice, these data would be replaced by weather forecasts.

The distribution of the outdoor temperature can be found in Figure 4. We sampled a total of 40 000 examples for our dataset, of which 38 000 were used for training, 1 000 for validation and 1000 for testing purposes.

Variable	Minimum	Maximum	Step
airchange_infiltration_vol_per_h	0.1	0.5	0.1
capacitance_kJ_perdegreK_perm3	50	300	10
power_VCV_kW_heat	0	1000	100
power_VCV_kW_clim	0	1000	100
nb_occupants	1000	2000	200
nb_PCs	1000	2000	200
percent_light_night	0	70	10
percent_PCs_night	0	70	10
facade_1_thickness_2	0.05	0.15	0.05
facade_2_thickness_2	0.05	0.15	0.05
facade_3_thickness_2	0.05	0.15	0.05
facade_4_thickness_2	0.05	0.15	0.05
roof_1_thickness_3	0.05	0.15	0.05
facade_1_window_area_percent	40	50	5
facade_2_window_area_percent	40	50	5
facade_3_window_area_percent	40	50	5
facade_4_window_area_percent	40	50	5

Table 3: $\theta\_buidling$ ranges.

Variable	Minimum	Maximum	Step
start_clim_day	7	9	1
end_clim_day	18	20	1
t_clim_red_day	24	30	0.5
t_clim_conf_day	20	24	0.5
start_heat_day	6	8	1
end_heat_day	17	19	1
t_heat_red_day	17	22	0.5
t_heat_conf_day	22	24	0.5
start_ventilation_day	7	9	1
end_ventilation_day	18	20	1
t_ventilation_day	18	26	0.5
vol_ventilation_day	0.7	1.7	0.3

Table 4: I_k ranges. Each parameter can hold a different value for each day of the week. For ease of reading, we replaced them by a single line, as the ranges are the same for every day.

Variable	Minimum	Maximum	Step
start_occupation_monday	7	9	1
start_occupation_tuesday	7	9	1
start_occupation_wednesday	7	9	1
start_occupation_thursday	7	9	1
start_occupation_friday	7	9	1
end_occupation_monday	17	20	1
end_occupation_tuesday	17	20	1
end_occupation_wednesday	17	20	1
end_occupation_thursday	17	20	1
end_occupation_friday	17	20	1

Table 5: O_k ranges.

Variable	Description
DNI	Direct Normal Irradiance
IBEAM_H	Direct Horizontal Irradiance
IBEAM_N	Direct Normal Irradiance
IDIFF_H	Diffuse Horizontal Irradiation
IGLOB_H	Global Horizontal Irradiance
RHUM	Humidity
TAMB	Outdoor temperature

Table 6: Weather data as contained in W_k.

Variable	Tested values	Chosen value
Latent dimension (d_{emb})	16, 32, 64, 128	64
Queries (W^q) and Keys (W^k) matrix size	4, 8, 16	8
Values (W^v) matrix size	4, 8, 16	8
Number of heads (h)	4, 8, 16	8
Number of layers (N)	4, 8, 16	4
Attention size (Δ)	6, 12, 24	12

Table 7: Hyper parameters tuned during grid search, along with their tested and chosen values.

A.2 Outputs

The BEM outputs 8 simulated variables at each time step, representing inside temperature as well as various consumption. These variables are aggregated differently during calibration and optimization. Since the metamodel aims at replicating the BEM behavior, it is trained to output these same 8 variables. See Table 8 for a exhaustive list and description of each one. Their distributions in the dataset can be found in Figure 5.

B Metamodel training

In order to find an optimal set of hyper parameters, we conducted a grid search for the Transformer model. A list of each parameter, along with its ranges and final value, can be found in Table 7.

C Calibration

Sensors installed in the building yield two time series.

- **Indoor temperature**: we average the values from a set of sensors, in order to obtain a unique indoor temperature value at each time step. This temperature is compared to the simulated indoor temperature (`T_INT`).
- **Heat consumption**: we define building heat consumption as the sum of multiple private heat consumptions obtained from sensors. This variable contains the heating consumption (corresponding to `Q_HEAT_OFFICE` for the metamodel), as well as the heating AHU consumption (`Q_AHU_HEAT`), and the equipment and lighting consumption (`Q_EQP` and `Q_LIGHT` respectively), see Table 8 for a description of the metamodel output variables. These four simulated variables are summed and compared to the real heat consumption.

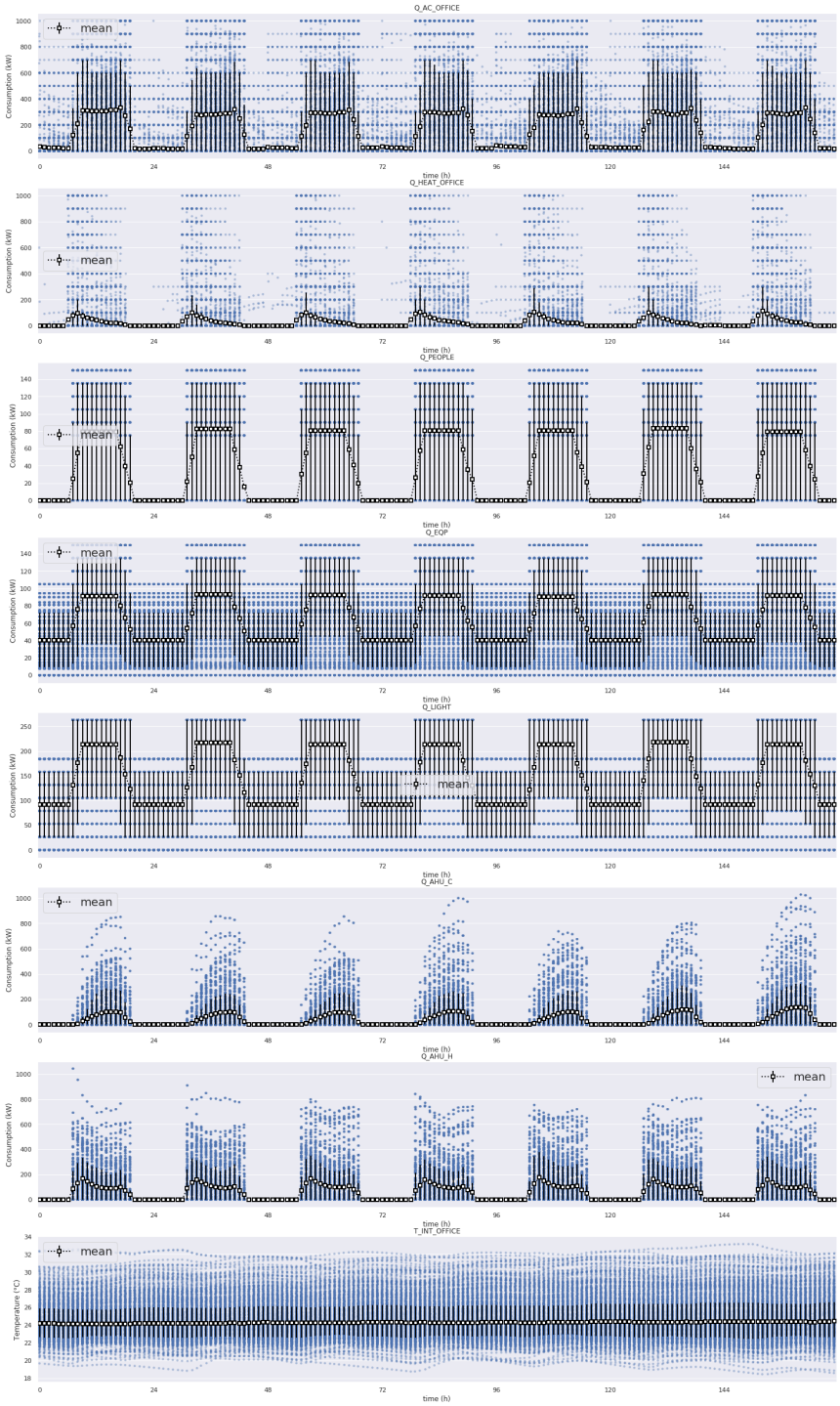

Figure 5: Distribution of the output variables of the BEM. See Table 8 for a exhaustive list and description of each one. Squares indicates the mean value, while vertical bars represent 85% of the data.

variable	description
Q_AC_OFFICE	AC consumption
Q_HEAT_OFFICE	Heat consumption
Q_PEOPLE	Heating power due to human activities in the building
Q_EQP	Consumption of equipment, such as computers, elevators, fridges
Q_LIGHT	Consumption of lights
Q_AHU_C	Consumption of AHU when cooling outside air
Q_AHU_H	Consumption of AHU when heating outside air
T_INT_OFFICE	Indoor temperature

Table 8: BEM's output variables at each time step.

References

[Bahdanau et al., 2015] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations (ICLR)*.

[Bengio et al., 1994] Bengio, Y., Simard, P. Y., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5 2:157–66.

[Bre et al., 2020] Bre, F., Roman, N. D., and Fachinotti, V. D. (2020). An efficient metamodel-based method to carry out multi-objective building performance optimizations. *Energy and buildings*, 206(1).

[Bre et al., 2016] Bre, F., Silva, A. S., Ghisi, E., and Fachinotti, V. D. (2016). Residential building design optimisation using sensitivity analysis and genetic algorithm. *Energy and Buildings*, 133:853–866.

[Cho et al., 2014] Cho, K., van Merrienboer, B., Çaglar Gülçehre, Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

[Coakley et al., 2014] Coakley, D., Raftery, P., and Keane, M. (2014). A review of methods to match building energy simulation models to measured data. *Renewable and sustainable energy reviews*, 37:123–141.

[Deb et al., 2000] Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International conference on parallel problem solving from nature*, pages 849–858. Springer.

[Hansen, 2016] Hansen, N. (2016). The CMA evolution strategy: A tutorial. *ArXiv:1604.00772*.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9:1735–1780.

[Igel et al., 2007] Igel, C., Hansen, N., and Roth, S. (2007). Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 11:1–28.

[Józefowicz et al., 2016] Józefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the limits of language modeling. *ArXiv:1602.02410*.

[Kim et al., 2016] Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *ArXiv:1412.6980*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Le Corff et al., 2018] Le Corff, S., Champagne, A., Charbit, M., Noziere, G., and Moulines, E. (2018). Optimizing thermal comfort and energy consumption in a large building without renovation works. *2018 IEEE Data Science Workshop (DSW)*, pages 41–45.

[Magnier and Haghighat, 2010] Magnier, L. and Haghighat, F. (2010). Multiobjective optimization of building design using trnsys simulations, genetic algorithm, and artificial neural network. *Building and Environment*, 45(3):739–746.

[McKay et al., 2000] McKay, M. D., Beckman, R. J., and Conover, W. J. (2000). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245.

[Mozer, 1989] Mozer, M. C. (1989). A focused backpropagation algorithm for temporal pattern recognition. *Complex Systems*, 3:349–381.

[Nagpal et al., 2018] Nagpal, S., Mueller, C. T., Aijazi, A. N., and Reinhart, C. F. (2018). A methodology for auto-calibrating urban building energy models using surrogate modeling techniques. *Journal of Building Performance Simulation*, 12(1):1–16.

[Parikh et al., 2016] Parikh, A. P., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (2016)*, pages 2249–2255.

[Reynolds et al., 2018] Reynolds, J., Rezgui, Y., Kwan, A., and Piriou, S. (2018). A zone-level, building energy optimisation combining an artificial neural network, a genetic algorithm, and model predictive control. *Energy*, 151(15):729–739.

[Shabunko et al., 2018] Shabunko, V., Lim, C., and Mathew, S. (2018). EnergyPlus models for the benchmarking of residential buildings in Brunei Darussalam. *Energy and Buildings*, 169:507–516.

[van den Oord et al., 2016] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *ArXiv:1609.03499*.

[van den Oord et al., 2018] van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., van den Driessche, G., Lockhart, E., Cobo, L. C., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., and Hassabis, D. (2018). Parallel wavenet: Fast high-fidelity speech synthesis. *Proceedings of the 35 th International Conference on Machine Learning (ICML)*, 80:3918–3926.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

[Westermann and Evins, 2019] Westermann, P. and Evins, R. (2019). Surrogate modelling for sustainable building design-a review. *Energy and Buildings*, 198(1):170–186.

[Zhao et al., 2016] Zhao, J., Lam, K. P., Ydstie, B. E., and Loftness, V. (2016). Occupant-oriented mixed-mode EnergyPlus predictive control simulation. *Energy and Buildings*, 117(1):362–371.

[Zhu et al., 2019] Zhu, X., Cheng, D., Zhang, Z., Lin, S., and Dai, J. (2019). An empirical study of spatial attention mechanisms in deep networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6688–6697.