Building and Environment 200 (2021) 107932

Contents lists available at ScienceDirect

Building and Environment

journal homepage: www.elsevier.com/locate/buildenv

Developing a parametric morphable annual daylight prediction model with improved generalization capability for the early stages of office building design

Yunsong Han [a,b], Linhai Shen [a,b], Cheng Sun [a,b,*]

[a] School of Architecture, Harbin Institute of Technology, Harbin, 150001, China
[b] Key Laboratory of Cold Region Urban and Rural Human Settlement Environment Science and Technology, Ministry of Industry and Information Technology, Harbin, 150001, China

ARTICLE INFO

Keywords:
Daylighting
Neural networks
Early design stage
Building performance

ABSTRACT

Building form and fenestration design decisions made in the early stages of design have considerable impact on the annual daylight performance of office buildings. Annual daylight performance needs to be evaluated at the conceptual design stage to support the building form and fenestration design decision-making process. However, the simulation modeling and ray-trace calculation required for annual daylight prediction are extremely time consuming, with an adverse impact on its feasibility in the early design stages. Machine learning-based models have received much attention to reduce the daylight simulation time; however, the generalization capability of these models is limited. This study develops an artificial neural network-based modeling approach to predict annual daylight performance in the early stages of the design process. A workflow to develop an annual daylight prediction model with higher generalization capability is proposed, through feature selection, feature engineering, and hyperparameter optimization, with an accompanying tool to integrate the machine learning model into the early design environment. The developed prediction model was validated against Radiance simulation results with a high accuracy setting and attained R^2 scores of 0.988 and 0.996, MAE scores of 1.58 and 1.37, MAPE scores of 2.10% and 2.36% for UDI and DA_{300}, respectively, while being 250 times faster. The proposed modeling approach can be extended by adding more types of parametric room modules.

1. Introduction

Natural daylight offers considerable health benefits to building occupants and reduces the energy required for artificial lighting. Access to daylight can reduce stress levels and increase the productivity of occupants [1–3]. The advantages of daylighting designs in the built environment have been widely demonstrated [4].

The quality and intensity of daylight vary according to building form, fenestration, optical properties of materials, geographical latitude and local climate. Accurate daylight predictions can support the design decision-making process, and could assist architects to understand the relationship between design variables and building daylight performance [5]. This decision-making feedback is especially impactful during the early design stages, as earlier decisions can significantly reduce the remaining design space and determine the optimization potential [6].

However, daylight prediction at the early design stages is an arguably and difficult task due to its inherently multi-faceted non-linear nature. The unpredictability of detailed information during the early stages of design makes it difficult to evaluate daylight performance accurately. Moreover, the balance between time and accuracy of daylight simulation is important during the early design stage as designers have to compare multiple design sketches and proposals.

A wide variety of methods to predict annual daylight performance have been explored by global scholars since the 1980s [7]. Existing daylight prediction methods can be classified into three categories: white box, black box, and grey box [8]. The white-box prediction methods evaluate daylight performance using physical principles. Contrary to the white-box methods, machine learning-based approaches are classified as black-box methods, as these assess annual daylight performance without understanding its internal correlation. The grey box approach consists of hybrid methods integrating both the white-box and the black-box methods.

* Corresponding author. School of Architecture, Harbin Institute of Technology, 1510# 66 West Dazhi Street Nangang Dist., Harbin, 150001, China.
E-mail address: suncheng@hit.edu.cn (C. Sun).

https://doi.org/10.1016/j.buildenv.2021.107932
Received 20 January 2021; Received in revised form 17 March 2021; Accepted 26 April 2021
Available online 5 May 2021

Both the white-box and the grey-box methods require detailed building information to evaluate annual daylight performance, and this is the major stumbling block for these methods to be used during the early design stage. In addition, the building daylighting simulation modeling process is a time consuming and laborious task, and this creates a divide between daylight performance evaluation and early design decision-making, limiting the potential benefits from performance evaluation [9]. However, machine learning-based methods can predict building annual daylight performance based on correlated variables, and this is a promising way to improve the efficiency of building design in its early stages.

Amongst the machine learning-based methods currently in use, this is an apparent preference for the utilization of ANN above other machine learning algorithms [10–14]. This is due to its good approximation capabilities for complex problems, better performance with large datasets, and fast processing times. ANNs are comprised of a group of neurons connected to each other through weights. Through the inductive learning process, the weights between the neurons are evaluated, and this enables the ANN-based model to overcome the nonlinearity between daylight performance and related design variables. While the ANN based prediction method has demonstrated its ability in predicting building total energy consumption [15,16], electric [17] and thermal load [18,19], its application for daylight prediction is relatively rare [20].

Several researchers have adapted the ANN model to forecast indoor illuminance. Kazanasmaz et al. [21] developed a 3-layer ANN-based model to predict indoor illuminance values using the time (date, hour), outdoor temperature, solar radiation, humidity and building form parameters as the input variables. Hu and Olbina [22] developed a multi-layer feed-forward ANN model to predict the illuminance level at a sensor point based on horizontal illuminance and sun angles as the input variables, and the illuminance percentage errors were less than 10% for most of the year. Logar et al. [23] proposed a fuzzy black-box model to estimate indoor illuminance with solar radiation, the external illuminance, the position of the blinds and the status of the lights measured in situ as inputs, trained on twelve days of measured data with reported MAE of 25 lux, RMSE of 12.60% and MBE of 7.76%. Ahmad et al. [24] compared the performance of ANN with random forest in predicting the hourly energy consumption and daylight illuminance values for a classroom, and the inputs included weather determines, time, window blind position, occupancy and previous hour values of daylight illuminance and energy consumption. The results show that random forest performed slightly better than ANN for predicting daylight illuminance, with coefficients of determination (R^2) of 0.9881 and 0.9799 respectively. Beccali et al. [25] trained several ANNs to predict the illuminance of a desired point on a working plane with data measured at other candidate positions, whose input variables included the illuminance values measured by four different sensors one at a time, the horizontal global irradiation, absorbed power, solar elevation, solar azimuth, time of day and day of the year.

These works focus on real time daylight prediction for the built environment, and are intended to improve the performance of various daylight control systems during the post occupancy stage, e.g., shading control [26–28]. Although these models show high prediction accuracy, they can't be generalized to other cases with different room sizes, space configurations, and orientations, since these models were exclusively trained with data measured at a specific site.

To address this problem, many authors have combined machine learning methods with daylight simulation to predict annual daylight metrics, as the methods can produce a large amount of structured data across long periods of time, which is otherwise highly impractical to obtain. Radziszewski et al. [29] applied ANNs to predict averaged daylight autonomy (DA), daylight factor (DF), and daylight glare probability (DGP), based on a dataset of 2763 random office spaces sampled from seven normally distributed design parameters. This method produced percentage errors for DA, DF, and DGP of 2.82%, 0.15%, and 1.06%, respectively, with gains speed boosts of 31, 3, and 17 times, respectively, compared to Daysim simulation. Lorenz et al. [30] trained ANNs to predict DA for 28 rooms of the same size with different window dimensions and locations. A total of 2057 training samples were obtained using Daysim, with DA at each sensor point as the output, and the sensor point coordinate, point id, window dimensions, and location as inputs. The results showed that the mean absolute error (MAE) and RMES ranged from 2.71, 3.44% to 3.99, 6.96%, respectively, for four different test cases. Besides ANNs, Verso et al. [31] developed a mathematical model using a polynomial function to estimate DA, continuous DA, and spatial DA, with orientation, room depth, window-to-wall ratio, and glazing transmittance as variable inputs, using 400 Daysim simulations as a dataset. R^2 scores of 0.985, 0.997, and 0.969 were achieved for DA, cDA, sDA respectively. Nault at el [32]. combined a parametric environment with Radiance simulation to generate a dataset for multiple linear regression and Gaussian processes. For sDA prediction, they obtained R^2 values of 0.65 and 0.48, respectively, on unseen test data. Ayoub et al. [33] developed a multivariate linear regression equation to predict sDA and annual sunlight exposure (ASE). Some 6480 training samples were simulated using a combination of 18 facade models of different window configurations with different building contexts. The resulting models achieved R^2 values of 0.884 and 0.877 for sDA and ASE, respectively.

Although these researches focus on daylight evaluation in various space, the existing model inputs still do not provide sufficient flexibility to make the necessary design changes requested by architects during the early stages of design, and their outputs are limited in the amount of information they provided. The various DA and DF metric used in these researches doesn't include an upper bound of daylight illuminance to indicate excessive daylighting. In addition, averaged daylight metric was used to represent a whole space, which oversimplified the indoor daylight distribution characteristic.

In summary, although various ANN-based methods for daylighting prediction have been developed, most daylight illuminance prediction methods and models developed are tailor-made or designed for specific cases only, and are thus infeasible in early design stages. Furthermore, models developed for annual metric prediction have critical generalization limitations, provide little spatial design information, and are often difficult to integrate into an existing design workflow, which may explain why machine learning-based daylight prediction is rarely used during the early stages of design.

In light of these shortcomings, this study proposes a daylight prediction modeling approach through the development of an ANN-based daylight prediction model with high generalization capability. Feature selection, feature engineering and hyper parameter optimization are conducted to improve the generalization capability of parametric daylight prediction model. This paper is organized as follows. Section 2 describes the methodology while Section 3 illustrates the case study results and discusses the findings. Section 4 provides the conclusions from this study.

2. Methods

2.1. Workflow

This section presents a detailed workflow for developing a daylight prediction model with relatively high generalization capability with weather data input (Fig. 1); it includes parametric prototype modeling, training data acquisition, feature engineering and data pre-processing, neural network structure modeling, and hyperparameter optimization. In the following sections, explanations for each step are provided to facilitate the implementation of the proposed workflow.

2.2. Parametric prototype modeling

A parametric model of an office space with a rectangular plan, which

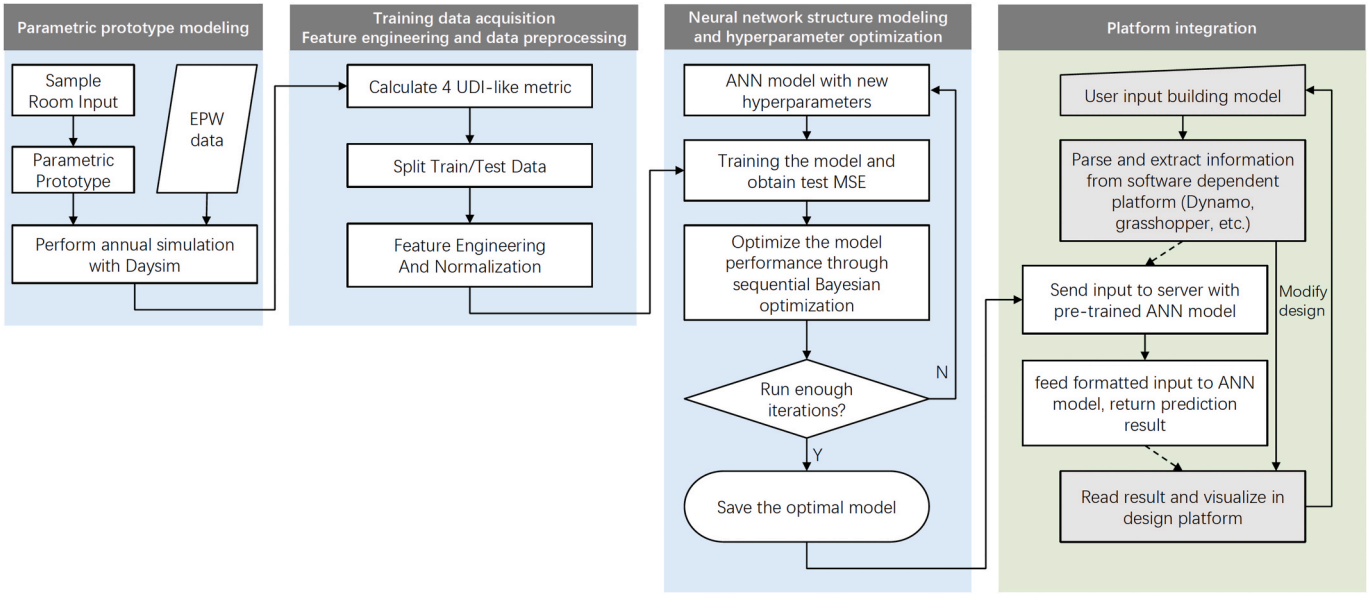


Fig. 1. The workflow for the predictive annual daylight modeling.

is one of the most commonly occurring space types in office buildings, is developed (Fig. 2). The characteristics of the office space model, such as width, depth and height of space, window size and position, surface reflectance of outside envelope, office orientation (angle between south and windows direction), transmissivity of window, thickness of the windowed wall can be parametrically changed.

To satisfy the engineering demands, the office building variables need to be limited in certain domains and steps. The bounds on the design variables were based on a typical cellular office in Harbin. The surface reflectance was based on previous research, which typically used a fixed combination of 0.5 for wall reflectance, 0.8 for ceiling reflectance, 0.2 for floor reflectance, and 0.7 for window transmittance. The corresponding range was set by extending the base value to both sides for more general use. The domain and steps for the office space variables are shown in Table 1.

Noteworthily, the choice of parameters and their ranges are not restricted to those presented, as the proposed workflow is designed to work with different parametric models, with possibly different input types and ranges. Since inputs beyond the decision boundary can be unreliable, parameter ranges should be modified according to specific application. For example, we can extend this model to residential space by lowering the minimum height to 2.0 m, while the rest of the workflow remains the same.

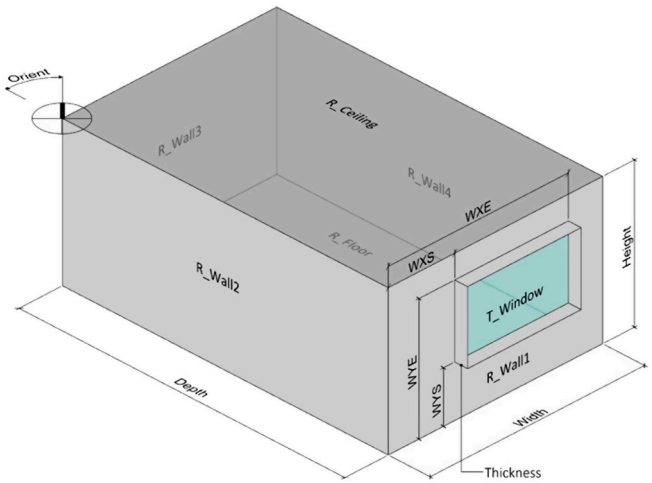


Fig. 2. The parametric prototype model developed in the study.

Table 1
Range and steps of the design variables.

Input	Range	Steps	Explanation
Width	3.0–8.0	0.01	Width of room(m)
Depth	3.0–8.0	0.01	Depth of room(m)
Height	2.6–4.2	0.01	Height of room(m)
Thickness	0.2–0.5	0.01	Thickness of the exterior wall
WXS	0.05–0.65	0.01	Window start position/room width
WXE	0.35–0.95	0.01	Window end position/room width
WYS	0.05–0.65	0.01	Windowsill height/room height
WYE	0.35–0.95	0.01	Window top height/room height
Orient	0–359	0.10	Angle between south and window direction
R_Wall1	0.3–0.7	0.01	Surface Reflectance for front wall
R_Wall2	0.3–0.7	0.01	Surface Reflectance for left side wall
R_Wall3	0.3–0.7	0.01	Surface Reflectance for back wall
R_Wall4	0.3–0.7	0.01	Surface Reflectance for right side wall
R_Ceiling	0.5–0.9	0.01	Surface Reflectance of ceiling
R_Floor	0.1–0.5	0.01	Surface Reflectance of floor
T_window	0.5–0.8	0.01	Transmissivity of window

Some combinations of the parameters are considered invalid. For example, WXE should always be larger than WXS; the same applies to WYE and WYS. In addition, to ensure that the window wall ratio was not too small, the additional constraints that WXE-WXS $>$ 0.3 and WYE-WYS $>$ 0.3 were imposed. This removed roughly three-fourths of all possible combinations, to be filtered out in subsequent steps.

2.3. Training data acquisition

The development of the ANN-based prediction model is based on a synthetic database containing the design variables and daylight performance indexes, which are used to train and test the prediction model. The parametric model developed in the previous section is used to generate.idf files for different office spaces for the daylight simulation.

Following a careful determination of the types, ranges, and steps for each of the variables for the office building, representative room samples are required for the training data. This is a crucial step, as a biased sample space will lead to overfitting only a subset of the whole design space, thus impairing the generalization performance.

The Latin hypercube sampling method with multi-dimensional

uniformity [34] is selected, as this ensures uniform distribution of samples in a multi-dimensional sample space. The Latin hypercube is the generalization of the Latin square concept to multi-dimensional space, whereby each sample is the only one in the axis-aligned hyperplane containing it. This can cover a large design space with the least possible number of samples and has been demonstrated to be much more efficient than random sampling for many applications. The Python library, LHSMDU [35], is used to generate the sample vectors for 16 inputs. All samples are then used to create the Rad files for the simulation process using the proposed parametric prototype model.

The daylight simulations are conducted using Daysim, which has been widely used and verified in indoor illuminance simulation studies [36]. The Rad files are imported to Daysim to calculate the annual daylight performance of the samples. The weather data for Harbin are input into the simulation tool, as the presented simulation methodology is demonstrated for office spaces in Harbin. However, the proposed daylight simulation method can be utilized for office buildings in other climate zones.

For each sample room, the light sensor points are located at the center of a 0.4 m * 0.4 m analysis grid on the working plane, at a height of 0.8 m. The total number of points is Width/0.4 * Depth/0.4 (rounded up). The time step for the daylight simulation is hourly, which generates 8760 hourly illuminance values for each sensor point. The detailed Daysim simulation settings are shown in Table 2.

An alternative approach is to use Useful Daylight Illuminance (UDI) to train an hourly illuminance prediction model first, and then to proceed to calculate UDI and DA. While this method can provide more flexibility for occupancy schedules, preliminary experiments show that it is much harder to train, and obtains poorer results when compared to predicting UDI directly, due to the significant increase in training time. The prediction of hourly data, while vital for real time applications, is less useful than the annual metric for early stage decision making.

The space occupancy schedule for the UDI calculation is 8 a.m. to 5 p.m., Mondays to Fridays. To retain more information than the standard DA and UDI metric, the resulting illuminance values at each sensor points are processed to obtain four UDI-like metrics representing the percentage of time across the daylight study period (one year), where the illuminance lies between the selected minimum and maximum values, which are 0–100 lux for UDI-n, 100–300 lux for UDI-s, 300–2000 lux for UDI-a, and above 2000 lux for UDI-x. These four indexes can be converted to UDI and DA using UDI = UDI-s + UDI-a and DA = UDI-a + UDI-x. UDI-a can serve as an alternative metric for space with higher daylight requirements, while UDI-x can be useful in detecting potential overexposure. Unlike DA and UDI, these 4 metric outputs should always sum up to 100, which provides the neural network with extra information. A finer division is also possible if desired; however, without a more detailed annual daylight metric, the extra information would be lost after translation to UDI and DA.

To train an accurate machine learning model, abundant data must be collected. However, the exact size of the dataset required is highly dependent on the inputs for the parametric model and the weather data used for simulation (e.g., a city within the ring of the Arctic Circle has significantly less varied sky conditions than other cities in the world, due to polar night; thus, its model would be easier to train); there is no rule of thumb to determine it in advance. Additionally, the annual simulations are very time-consuming; therefore, it is not ideal to create extremely large datasets that are more than sufficient. A data-driven approach can be adopted, instead, to obtain satisfactory performance with minimal time investment. With each additional 150 samples simulated, a new model was trained and evaluated. When the newly added data no longer improved the accuracy significantly, the data generation was stopped.

Table 2
Daysim simulation parameters.

Parameter	Range*	Explanation
Ab	5	Ambient bounces
Ad	512	Ambient divisions
As	256	Ambient Samples
Ar	128	Ambient resolution
Aa	0.15	Ambient accuracy
Lr	8	Limit reflection

2.4. Feature engineering and data pre-processing

This section describes the data normalization and feature selection and engineering steps. Having obtained the raw daylight simulation data, 85% of the them were randomly chosen for the training process, while the remainders were reserved as unseen test data for final validation. The train/test split must be performed before any subsequent data manipulation, as, otherwise, the information from the test dataset could be leaked to the training dataset, invalidating the test result. Thus, all the processes that follow in this section were performed on the training and the test datasets separately.

The raw simulation dataset consists of 4 * N (number of points per room) prediction targets for each input sample, and is referred to as the room-wise dataset. In previous research, these are often simply averaged into a single data point; in this study, we preferred to retain the daylight distribution characteristics rather than compress the points. However, a problem of incompatible data arose when attempting to merge the data from rooms of different sizes: If a 0.4 m analysis grid was used to generate the sensor points, each room presented a different number of target outputs. One workaround was to use a fixed number of sensor points for all rooms, regardless of their dimensions. In related works, each point in a room is assigned an ID number as a categorical feature; however, the spatial relationship between points is lost during the process. Each sample is required to have the same number of sensor points, and this results in non-uniform sensor point distribution when the room dimensions are varied. While this is a solution to the problem of the non-uniform analysis grid, it results in limited prediction resolution, due to the inefficiency of the multi-output prediction problem.

The solution to this problem was found by feature engineering the spatial features of the sensor points by reorienting and normalizing their position to the coordinate system local to the room. As illustrated in Fig. 3, the spatial relationship between the sensor points is represented by the X and Y coordinates on the 2-D working plane. However, these coordinates become highly dependent on the room width, depth, and orientation, introducing unnecessary complexity. To orthogonalize all inputs, a local coordinate system is created on the bottom left corner of the room (when viewed from the top with windows facing down), and with the X axis set along the width direction and the Y axis along the depth direction. All coordinates are then normalized by dividing the X and Y components by room width and depth, respectively. By doing this, the relative point position becomes irrelevant to the room orientation and dimensions.

The benefits of this feature engineering are threefold: First, it explicitly defines the spatial relationship between the different points, and decouples the spatial feature from the room dimension and orientation. Second, the ANN can learn to predict UDI with continuous spatial input, which means that it becomes possible to obtain the UDI at any position, without being restricted to a predefined simulation grid. Third, this significantly increases the number of samples, which has the potential to improve the model performance.

Consequently, every room sample yields N * training entries, each containing 16 (the input size of the parametric model) plus 2 spatial features as inputs and 4 daylight metrics as output. The converted dataset is referred to as a point-wise dataset. The min-max normalization function is applied to the new dataset to normalize all the input values from 0 to 1 along the feature axis.

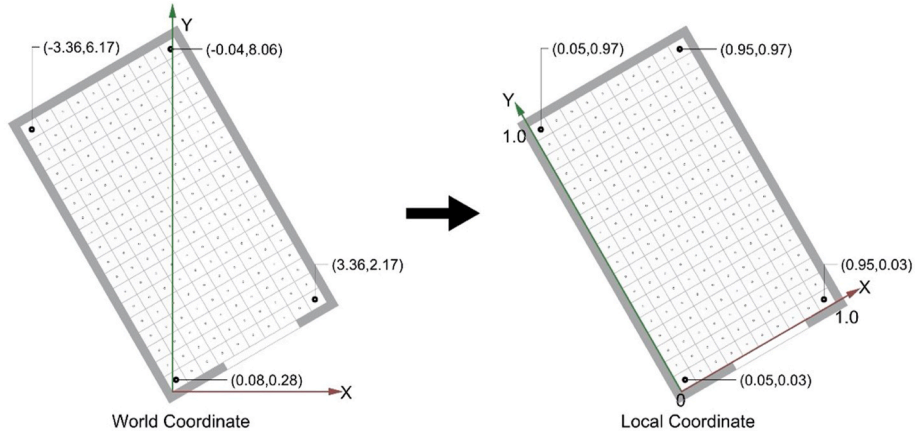


Fig. 3. Coordinates remapping from world space to local space, by rotating the coordinate basis and normalizing the X and Y components by width and depth, respectively.

2.5. Neural network architecture and hyperparameter optimization

Following data acquisition and pre-processing, the next step is to define the network type and architecture. A fully connected feedforward neural network, with multiple hidden layers, is selected as the base model. Bayesian hyperparameter optimization [37] is performed using skopt [38]. The neural network model is developed using the Keras library [39]. The structure of the proposed ANN is shown in Fig. 4, which is composed of 1 input layer with 18 (inputs dimension) neurons, N hidden layers with M neurons, and 1 output layer with 4 (inputs dimension) neurons.

The configuration of hidden layers is determined by parameters M and N. These model-dependent parameters, whose values are set before the training (thus not learnable but defined by the user), are referred to as hyperparameters. Different combinations of hyperparameters typically lead to very different results. Systematic hyperparameter optimization can help reduce the variance of network performance and produce optimal results, and hence this is preferable to tuning parameters manually. In this study, Bayesian optimization was used as a hyperparameter searching algorithm, as this has been shown to be more efficient and effective than the random and grid search methods [40].

For ANN, prominent hyperparameters include batch size, number of hidden layers, hidden layer size, activation function between layer and learning rate. In this study, the hyperparameters to be optimized include the hidden layer count N and layer neurons M, which determine the model complexity and capacity, plus the training batch size K, which does not influence the network itself but determines the amount of data to be observed before updating the network weights. The ranges of these parameters are set empirically based on preliminary experiments, with N ranging from 2 to 5, M ranging from 128 to 800 and K ranging from 20 to 200.

The neural network uses a gradient descent method with mini batching, instead of the Levenberg–Marquardt and L-BFGS algorithms commonly used, since the latter are limited to smaller datasets and networks because of the significant computational burden they impose [41]. In this study, training with such algorithms would take too long, and they could even fail to converge due to the large amount of data generated during data pre-processing. Specifically, the Adam training algorithm was used, with an initial learning rate of 0.001. As this algorithm can regulate it automatically, the learning rate is not part of the optimization. Regularization was unnecessary in this case, as the training and test sets shared the same distribution and contained little noise. The ReLU activation function was used for all hidden layers, whereas linear activation was used for the output layer. The uniform layer weights initialization method was applied for each hidden layer.

During the training process, an additional learning rate scheduler and an early stop detector are adopted. It is found that reducing the learning rate on the plateau helped the model to converge much faster

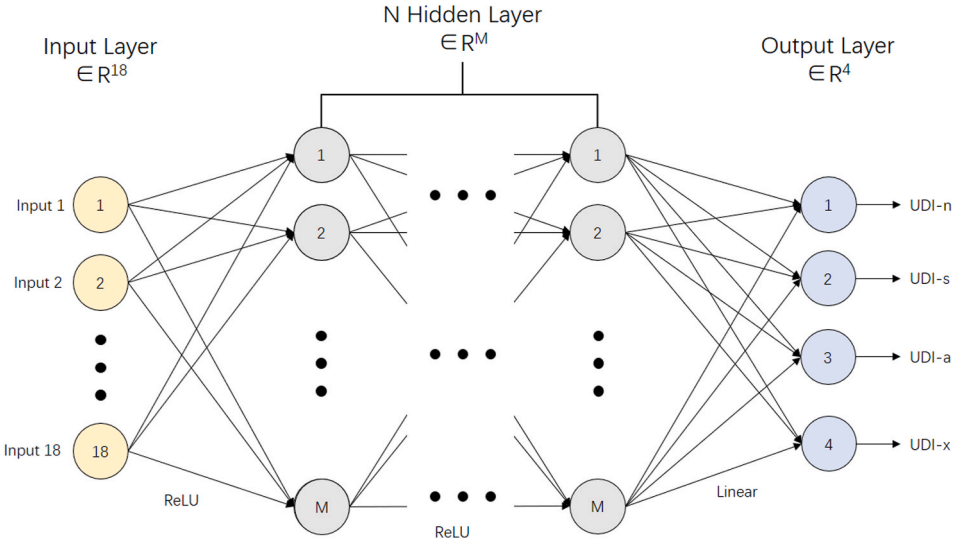


Fig. 4. Structure of the proposed ANN.

when combined with the early stop detector. An NVIDIA 1080Ti GPU was used as training device.

Having obtained the final model, a simple constrain function was applied to clamp all prediction values within [0,100]. In this experiment, the difference was barely noticeable; however, this step could be improved with different datasets. The mean square error (MSE) metric, calculated by Equation (1), was used to determine the optimal hyperparameters:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(predict_i - true_i)^2 \quad (1)$$

The number of optimization iterations, like the data samples mentioned in Subsection 2.3, was determined through a data-driven approach. At the beginning of the optimization, a relatively large max iteration limit of 60 was set. For each subsequent iteration, the algorithm explored a possibly better hyperparameter combination, based on the performance of previous models, thus yielding a new prediction model. The MSE score for the newly trained model, with five-cross validation, was then recorded and compared to the previous lowest. After ten iterations without significant (>2%) improvement, the optimization process terminated and output the model with the lowest MSE in the records as the final optimal model.

2.6. Validation of the optimal model

To validate the accuracy and generalization capability of the optimal model, it must be tested on the previously reserved test dataset described in Subsection 2.4. In addition to the MSE, the MAE, mean absolute percentage error (MAPE), and R^2 were used (Equations (2)–(4)). A sensitivity analysis was also performed on all the input parameters, using the Sobol method.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|(predict_i - true_i)| \quad (2)$$

$$\mathrm{MAPE} = \frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}}\left|\left(\frac{predict_i - true_i}{true_i}\right)\right| \times 100\% \quad (3)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(predict_i - true_i)^2}{\sum_{i=1}^{N}\left(predict_i - \overline{true}\right)^2} \quad (4)$$

The previous validation would be enough if only one averaged value was predicted for each room sample; however, the proposed model is meant to predict the daylight metric distribution in the whole room. Thus, the spatial error distribution is as important as the overall error. To investigate the spatial error distribution, the MAE and MAPE scores were calculated separately for points at different locations, using a 20*20 grid. Besides, as mentioned in Subsection 2.4, one benefit of feature engineering is that it allows the model to represent the space as a continuous 2-D space. To validate this, 100 more randomly sampled simulations were created and processed using the same setting as the original, except that, for their analysis, the grid was set to 0.2 m, instead of 0.4 m. The spatial error distribution of the high-resolution prediction was then calculated and investigated.

Two more experiments were carried out to compare the results of the prediction model to the Radiance simulation under various conditions. In the first experiment, a sample room was simulated using a lower ambient bounce setting, ranging from two to four; additionally, the aim was to trade accuracy for shorter simulation time. The errors of the lower accuracy simulations were then calculated and compared to the errors of the model prediction. In the second experiment, a series of rooms were sampled and arranged side by side, such that they did not shadow each other. The time cost to calculate UDI was measured for the prediction model, with simulation at high (ab = 5) and low (ab = 2) accuracy settings.

2.7. Design platform integration

The previous sections present the workflow to develop a functional daylight prediction model. For the ANN-based model to be accessible to architects, the gap between the design platform and the prediction model must be filled [42,43]. Current research has not yet identified a reliable and convenient manner of integrating machine learning models with design tools, which is the main obstacle to their use in practice. This section describes the development of a platform-agnostic framework to host the prediction model which can be implemented with various design software, like the popular Ladybug Tools [44]. Two popular parametric design platforms, Rhino with Grasshopper and Revit with Dynamo, are considered.

The proposed model creation workflow is developed within the Python environment. However, both Grasshopper and Dynamo use Iron Python, which is based on C# and only support the 2.7 version, whereas many machine learning libraries, such as Keras and Tensorflow, are only available in 3.x version, and different versions of Python cannot communicate directly. This problem is solved by developing a framework consisting of a simple HTTP server hosting the prediction model and three platform dependent modules. The backend server has three main functions: (1) Load a pre-trained ANN model and its data preprocessing pipeline. (2) Receive incoming data from external source, transform this according to pre-processing pipeline. (3) Feed the transformed input to ANN and send the output back to the source.

On the application side, the workflow consists of three parts: the input collection module, the proxy prediction module, and the visualization module. During the design process, the input collection module collects the required input parameters and sends the data to the backend server via the proxy prediction module. The server then pre-processes the input data, as described in Section 2, feeds the data to the ANN model, and then sends the result back. The visualization module presents the results in visual format directly on the corresponding platform.

During the entire process, it is only primitive number data types that are transferred, solving the cross-platform compatibility problem. This also enables remote access to the prediction model by hosting it on a cloud server. The user neither needs to know the details of the model nor how to configure the Python environment; the only requirement is access to the internet. Alternatively, if the model is only intended to be used locally, the exchange of data via file is a better choice, since this offers faster transmission speeds. In this case, two temporary files are created on disk to store the input and output data. As proof of concept, the framework is implemented in both Grasshopper and Dynamo.

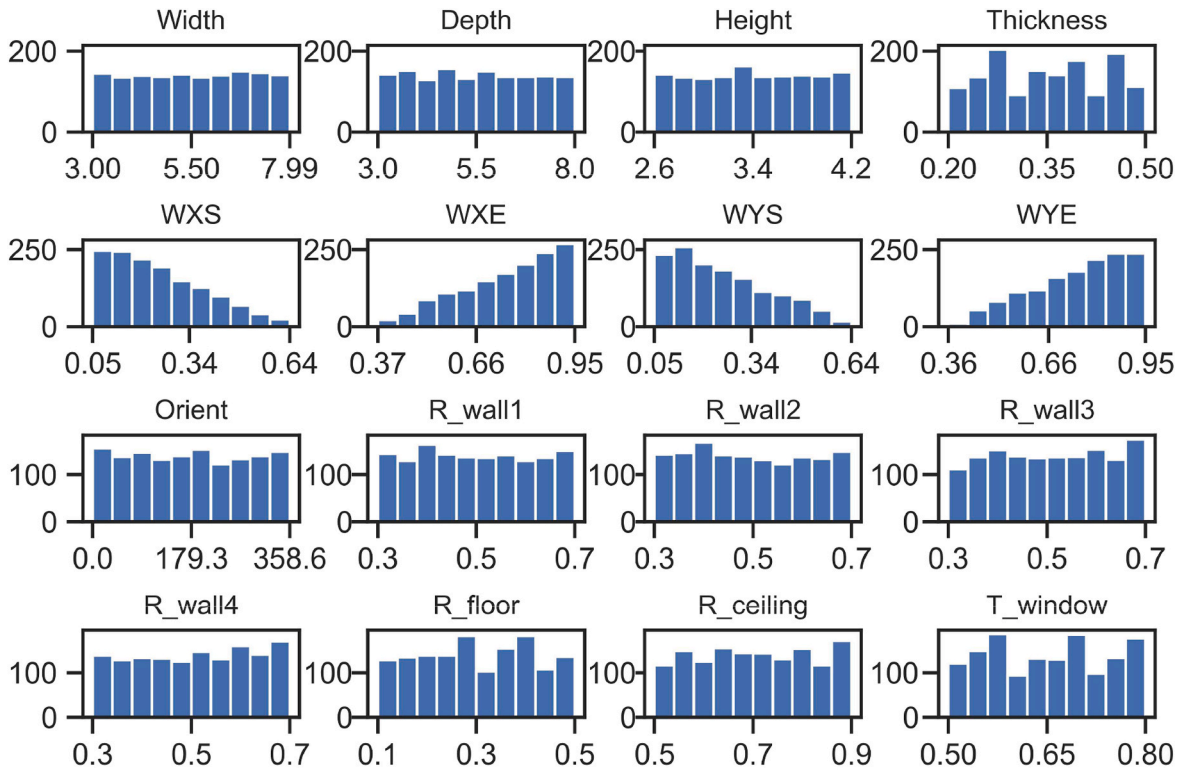
3. Results and discussion

3.1. Results of data acquisition and processing

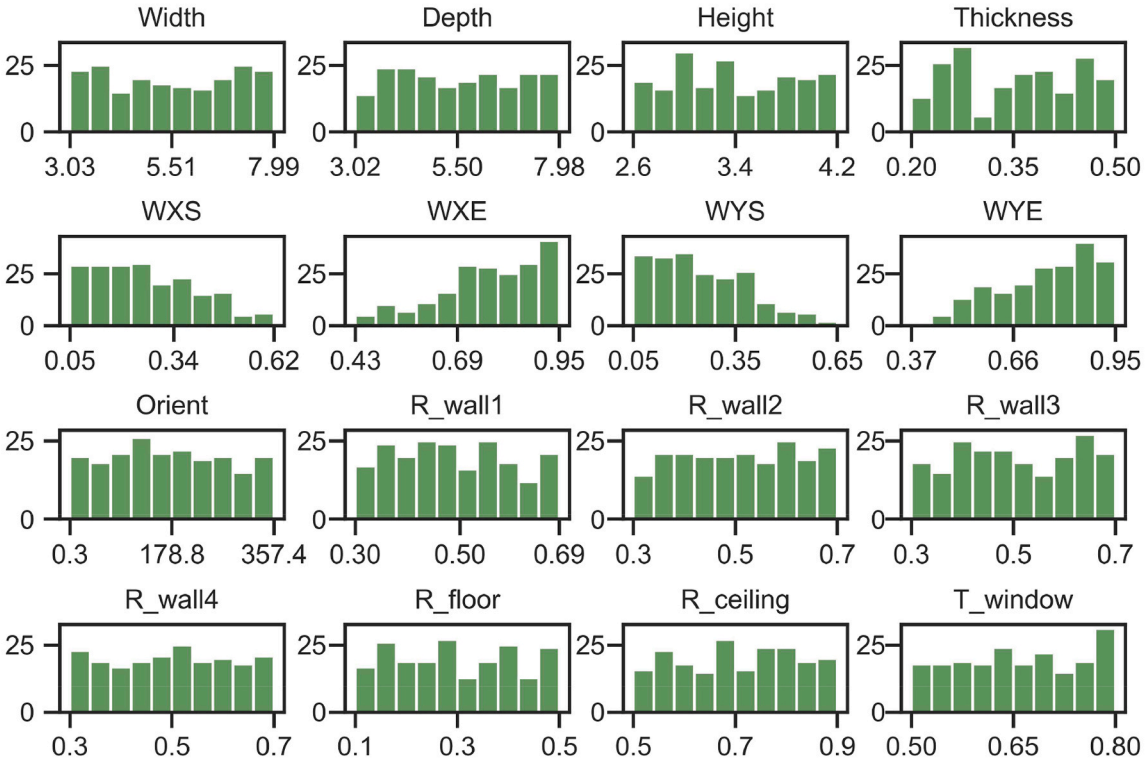
Initially, only 1600 rooms were sampled using the LHSMDU Python library. After filtering out invalid data according to the parametric room setting, 415 valid combinations were simulated with Daysim and converted into training and test datasets. More simulation data were progressively created in batches of 150, depending on the model's performance, until total simulation reached 1610, including 1410 for training and 200 for testing. The simulations for all 1610 samples took approximately 64 h in total, with an average of 2.4 min per sample on an Intel i7-4790k CPU using 8 threads. Each sample had 8*8 to 20*20 points, depending on the width and depth, resulting in a total of 281894 datasets composed of 246130 training data and 35764 test data. As shown in Figs. 5 and 6, all input dimensions have an overall uniform distribution, except for the four window position inputs, whose values are culled.

3.2. Results of Bayesian optimization

The results from the hyperparameter optimization, which



(a) Training



(b) Test

Fig. 5. Variable distribution of training and test samples. The Y axis represents the number of samples.

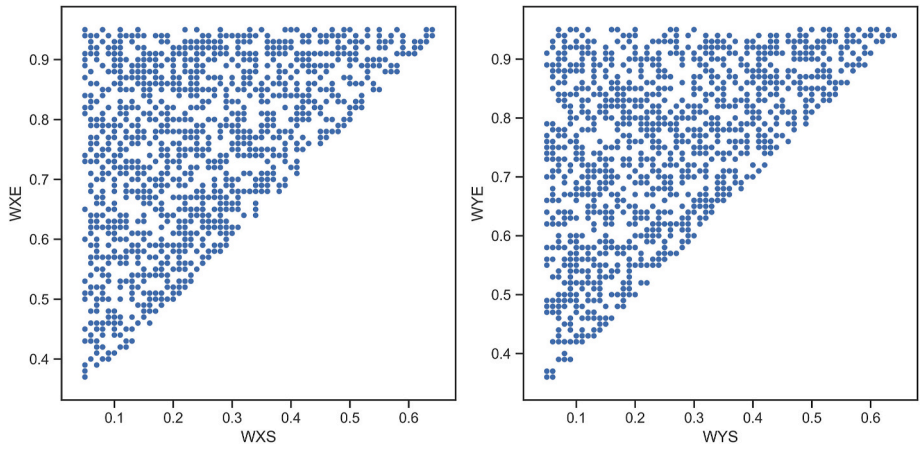


Fig. 6. Joint distribution for two sets of dependent variables.

successfully converged after 40 iterations, are presented in Fig. 7. The shaded contour plot shows the estimated joint influence of two hyperparameters, with each dot representing a model, while the lighter colors indicate better performance. The curve plots demonstrate how a single hyperparameter affects the test MSE.

Both increase the hidden layer count, and neuron size improves the model performance before reaching 4 layers and 600 neurons. Thereafter, the curve starts to plateau, as more complex networks require larger data sizes to train; otherwise, the model eventually overfits the data and lowers the test score instead. For this problem, 4–5 layers deep and 600–800 neurons are sufficient.

The same is true for batch size, for which a decrease in value generally leads to better performance before overfitting starts. Besides, the training time is roughly inversely proportional to batch size, and therefore a larger value is preferred.

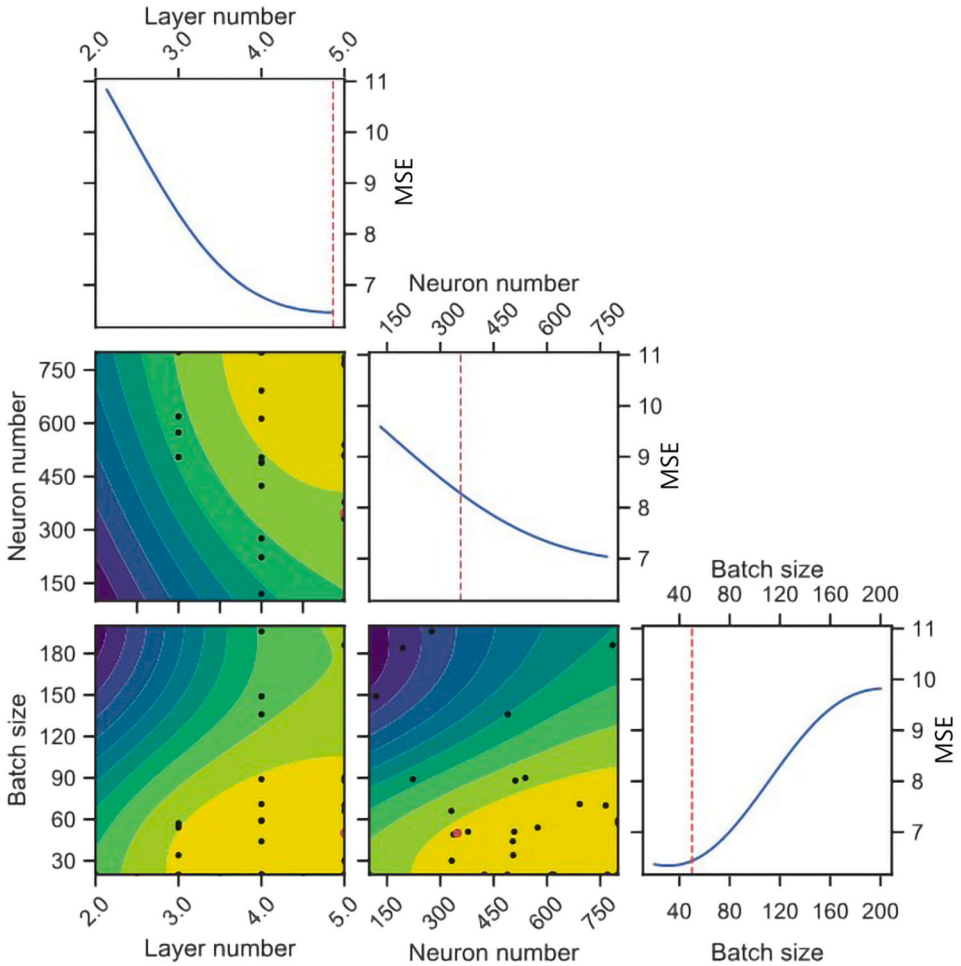


Fig. 7. Performance metrics for training and test datasets for different hyperparameters.

3.3. Validation of the optimal model

Hyperparameter optimization is used to determine the theoretical best hyperparameters of 5 layers, with 800 neurons each, trained with a batch size of 50. The following section analyses the performance characteristics of the optimal model in greater depth.

The training process is plotted as shown on the left of Fig. 8. The neural network reaches a performance plateau after eight epochs in 15 min: however, the performance stops improving at epoch 8, with 1.67 MAE and 0.992 R^2. The final performance of the prediction model is detailed in Table 3. The DA and UDI are calculated based on the four predicted metrics.

The relationship between training sample size and model performance is presented in the form of a learning curve (Fig. 8, right). Judging from the derivative of the curve, the ANN can still be improved if provided with more training data than 1410 samples; however, it is very close to the limit and unlikely to drop below five. Thus, we stopped generating more training samples at that point.

The error distribution and regression plots are shown in Fig. 9. While 99.2% of errors are within ±10, 95.2% fall within ±5, indicating an overall good fit. A few extreme values are observed; one of these extreme cases, with the highest error, is investigated and depicted in Fig. 10.

In this case, the opening is high and narrow. The highlighted red points are scattered around the periphery of the over-lit area. The drastic change from the lit to the shaded points causes significant spatial discontinuity. Therefore, even a small displacement of the prediction value can lead to a large error when this occurs near the discontinuity curve. However, the prediction can still be considered valid, as it captures the overall distribution characteristics accurately. These errors are rare, and generally appear as scattered points, which are less detrimental than smaller errors that occur in clusters. In practice, the subtle spatial error should have little effect on designs and can be averaged with surrounding values in the optimization algorithm.

Fig. 11 presents the sensitivity analysis for each input. Room orientation has the highest impact on UDI, followed by depth and height, which are all early stage decision variables. The floor and ceiling reflectance has much less impact than the other parameters, which can be attributed to its limited range (0.1–0.5 and 0.5–0.9, respectively).

3.4. Validation of spatial distribution prediction

For a more comprehensive demonstration of performance, several cases that were randomly chosen from the test sets are presented and compared to the corresponding simulation results in Fig. 12. As previously shown in Fig. 9, most errors are within a range of 5. The predicted UDI tends to be higher than the simulated values.

Fig. 13 illustrates the remapped MAE and MAPE for each metric at each point. The width of each rectangle can range from 0.2 m to 0.4 m in actual space. Four prediction targets and the derived UDI and DA values are presented.

Higher MAEs tend to be around room corners and close to the walls with openings. The calculated UDI distribution has a lower error compared to UDI-s and UDI-a, as many errors result from misclassification between these two metrics. These errors cancel each other out when the metrics are combined. The same is true for DA, as the MAPE distribution plot suggests. The MAPE of UDI-x is much higher, as the absolute values of UDI-x are usually lower, corresponding to the lower MAE scores.

Table 3
Performance on unseen test dataset.

Metric	MSE	MAE	MAPE (%)	R^2
UDI-n	4.70	0.93	6.02	0.990
UDI-s	7.82	1.63	6.20	0.987
UDI-a	6.88	1.75	3.58	0.993
UDI-x	2.10	0.79	8.55	0.994
UDI	7.10	1.58	2.10	0.988
DA	5.30	1.37	2.36	0.996

As presented in Fig. 14, the high-resolution prediction result also matches the simulated values closely, despite the higher error near the front wall and corners. This is due to the spatial features of high-resolution data were slightly beyond the original input range. E.g., for 8 m*8 m room, the resolution is 20 * 20 with 0.4 m analysis grid, the maximal range of normalized spatial feature was therefore 0.05–0.95. But with 0.2 m analysis grid, the resolution was doubled to 40*40, resulting in larger range of 0.025–0.975. This can be addressed by adding some high-resolution samples in data. On the other hand, points with spatial feature value within the original range have consistent result. In practice, these peripheral regions with higher error would only cover a portion of the working plane. Thus, the averaged error on the whole working plane would be smaller. It can safely be assumed that the neural network learns the rule of daylight distribution.

Apart from the errors around the corner, a ripple-like error pattern is observed in Fig. 15. This is because the predictions tend to give smoother results, whilst the simulation has more noise and discrete values, since the annual simulation is calculated using representative hours over the year.

Fig. 16 compares the accuracy of the prediction model to the simulation, at lower ambient bounce setting. The prediction result, with the lowest average error against the benchmark, outperforms the other simulations; however, the regional error is not as consistent as the other simulation results. The prediction error distribution is irregular, while all the simulation results display a gradient transition from positive to negative error.

Table 4 details the time cost to obtain the UDI metric with different methods. For a single 7.2 m*7.2 m room, the prediction takes only 5 ms plus 31 ms data transmission time. The prediction time for ANN increases linearly in proportion to the number of predicted points. An increase in the number of points has relatively less impact on the simulation time. For most building floor plans (100*100 points or 50 × 50 m space with 0.5 m resolution), the ANN model is at least 200 times faster than Daysim, with 1.58 MAE for UDI and 1.37 MAE for DA. In

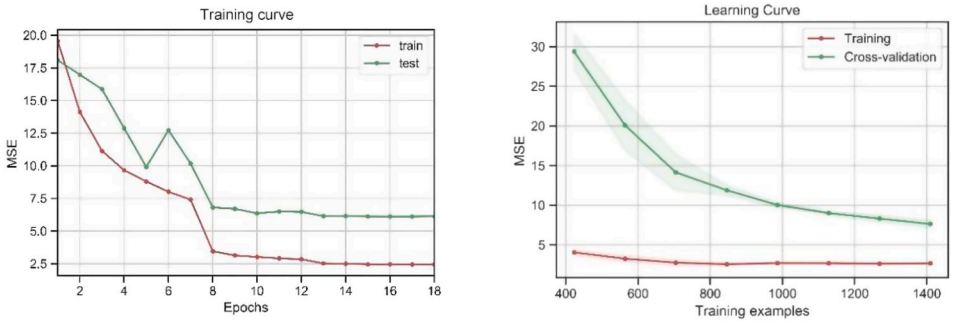


Fig. 8. Training curve(left) and Learning curve(right).

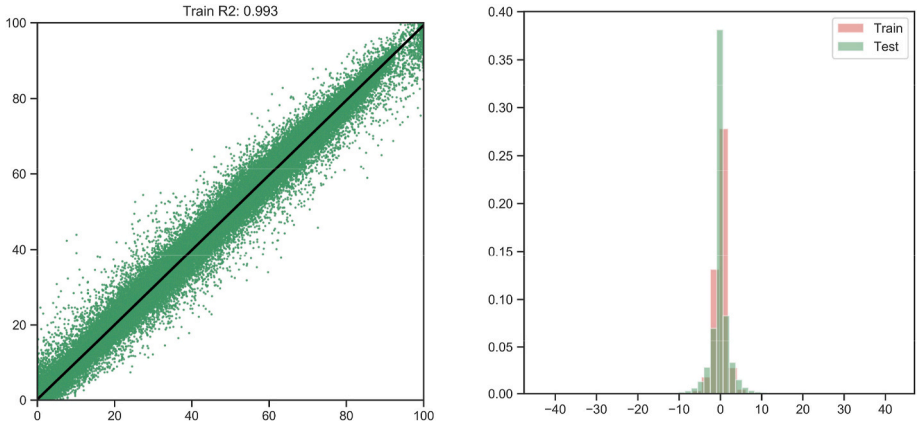


Fig. 9. Regression (left) and error distribution(right).

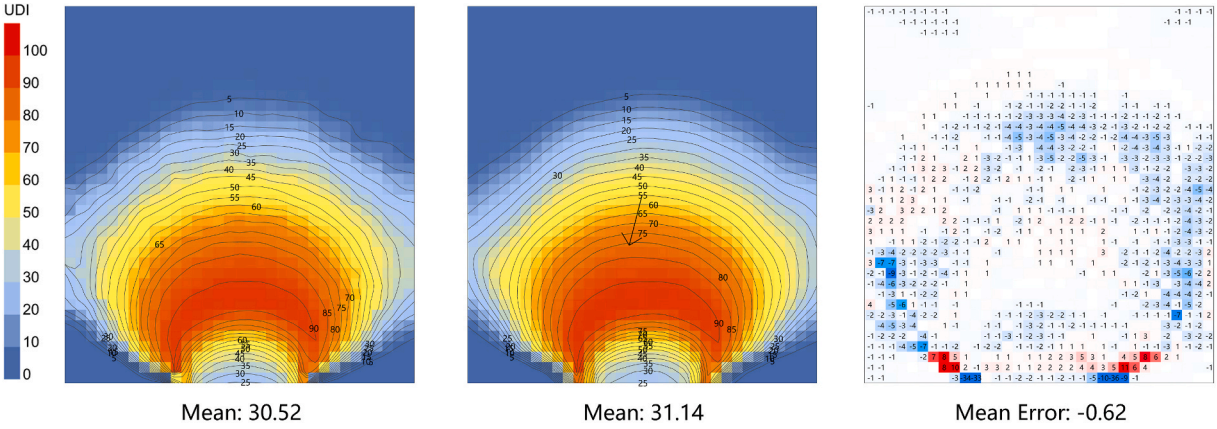


Fig. 10. Sample room with highest error. Left: simulated (ground truth). Middle: predicted. Right: errors (simulated-predicted).

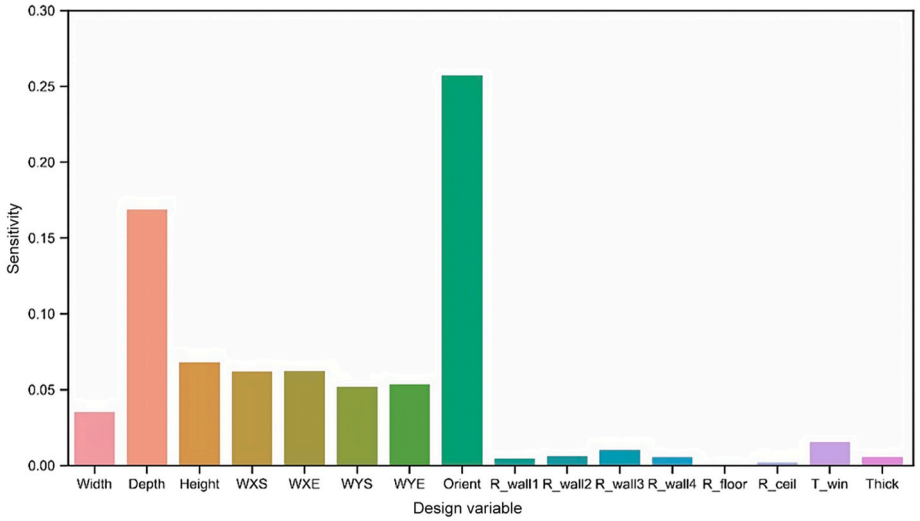


Fig. 11. Sensitivity analysis of UDI for all design variables.

comparison, the GPU accelerated simulation only achieves up to 10 times the speed boost for annual simulation [45].

The massive speed boost not only enables an interactive design experience that benefits decision making, but also greatly reduces the optimization cost, as the proposed daylight prediction model can serve as a faster replacement for the simulation engine for building form optimization tasks to improve daylight performance.

3.5. Design tool integration

Besides the prediction speed boost, the modeling process can be further simplified through design tool integration to provide improved support to the designer. With the proposed daylight prediction workflow, the parametric prototype model based on the integration plugin developed in Grasshopper (Fig. 17) can be repurposed to define all room

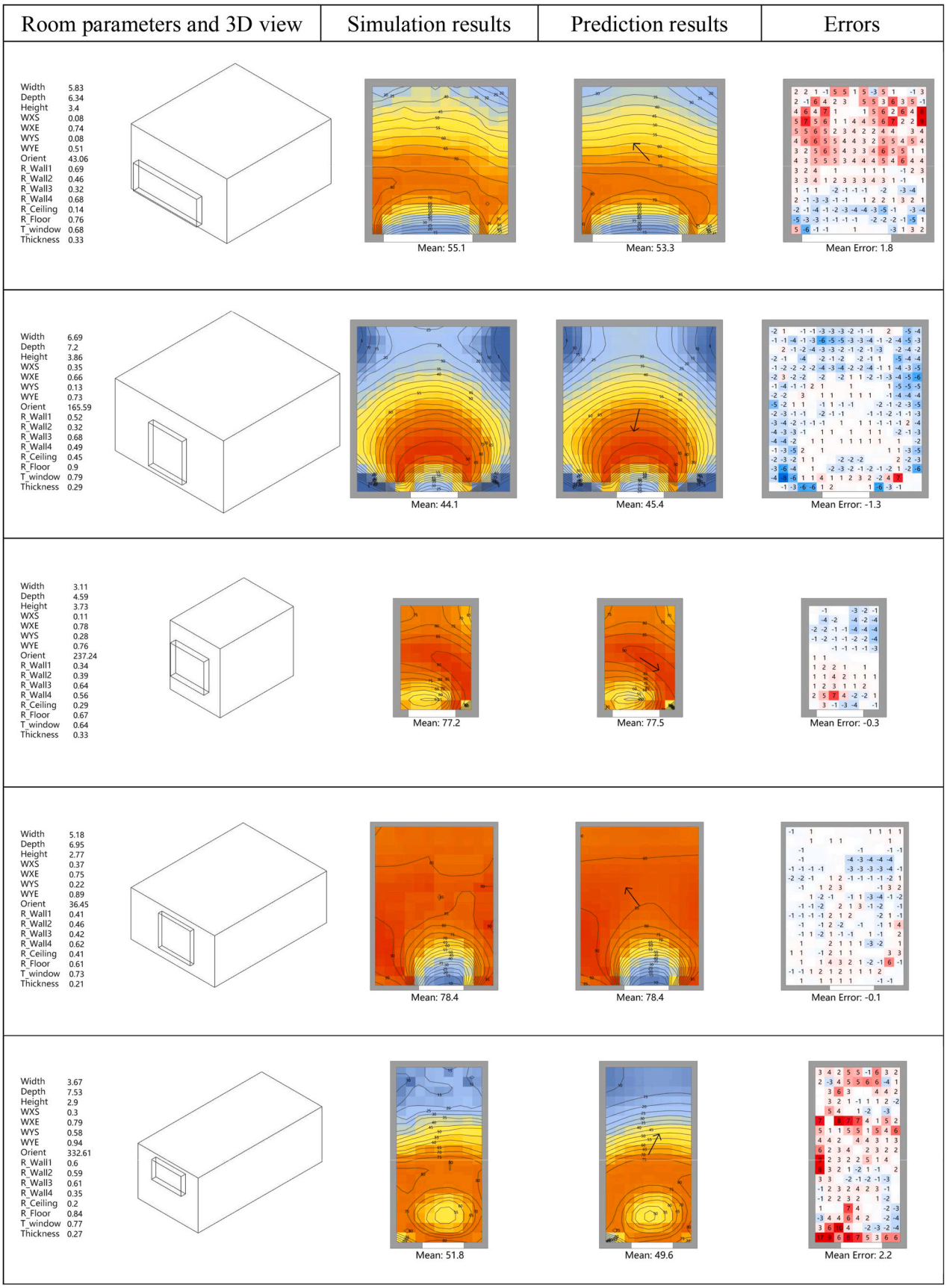


Fig. 12. Comparative results for several random samples of test data.

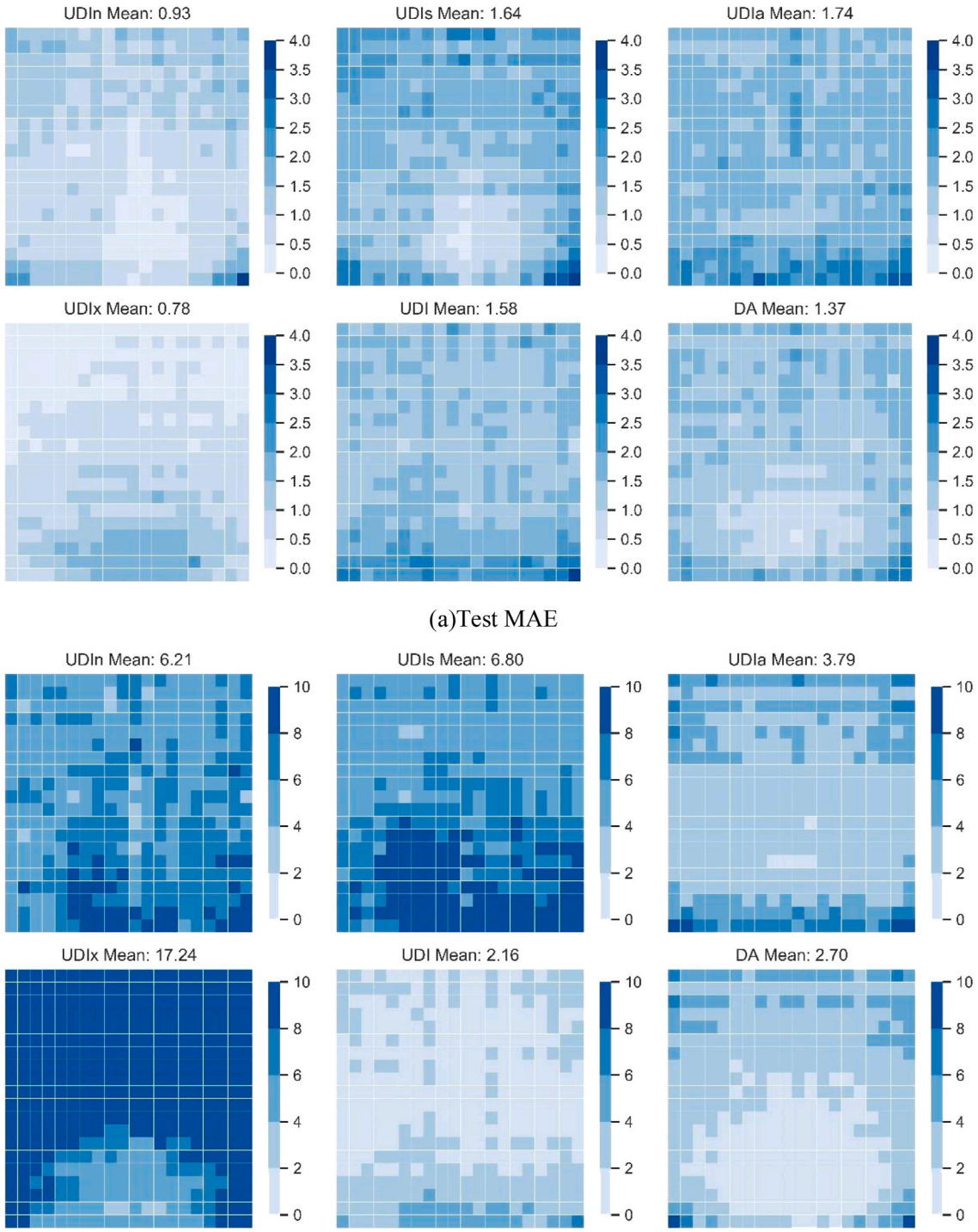
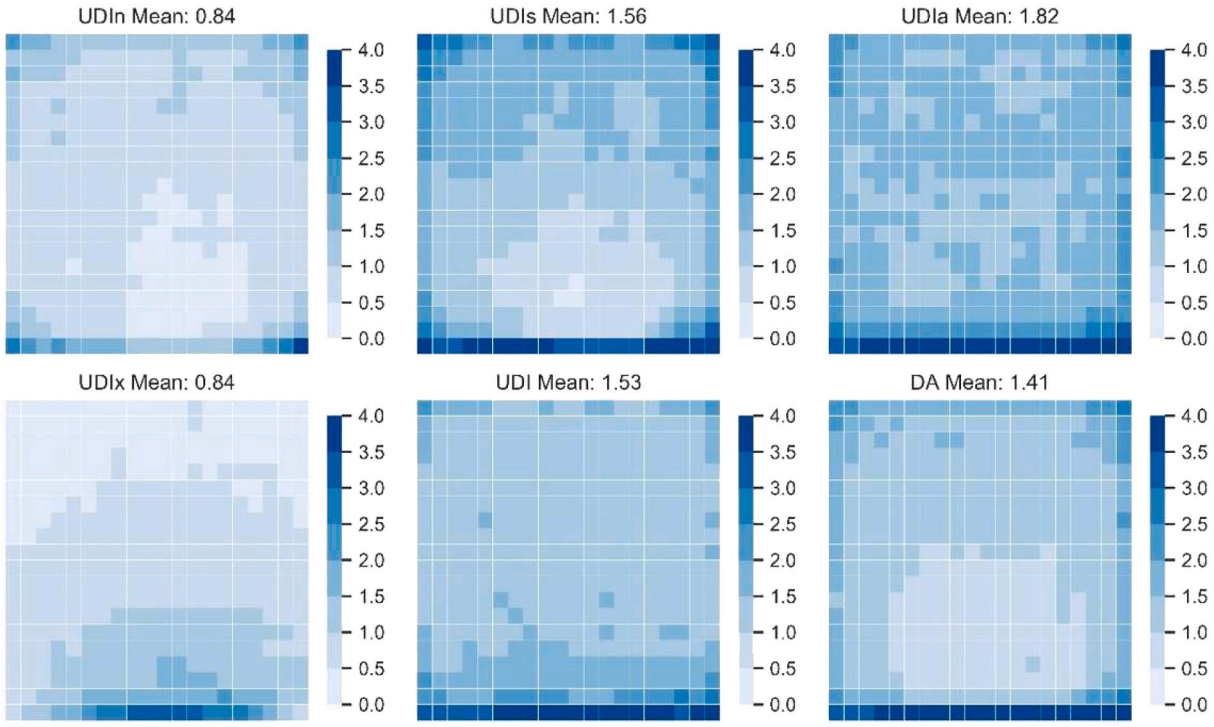


Fig. 13. Spatial MAE and MAPE (%) of best model.

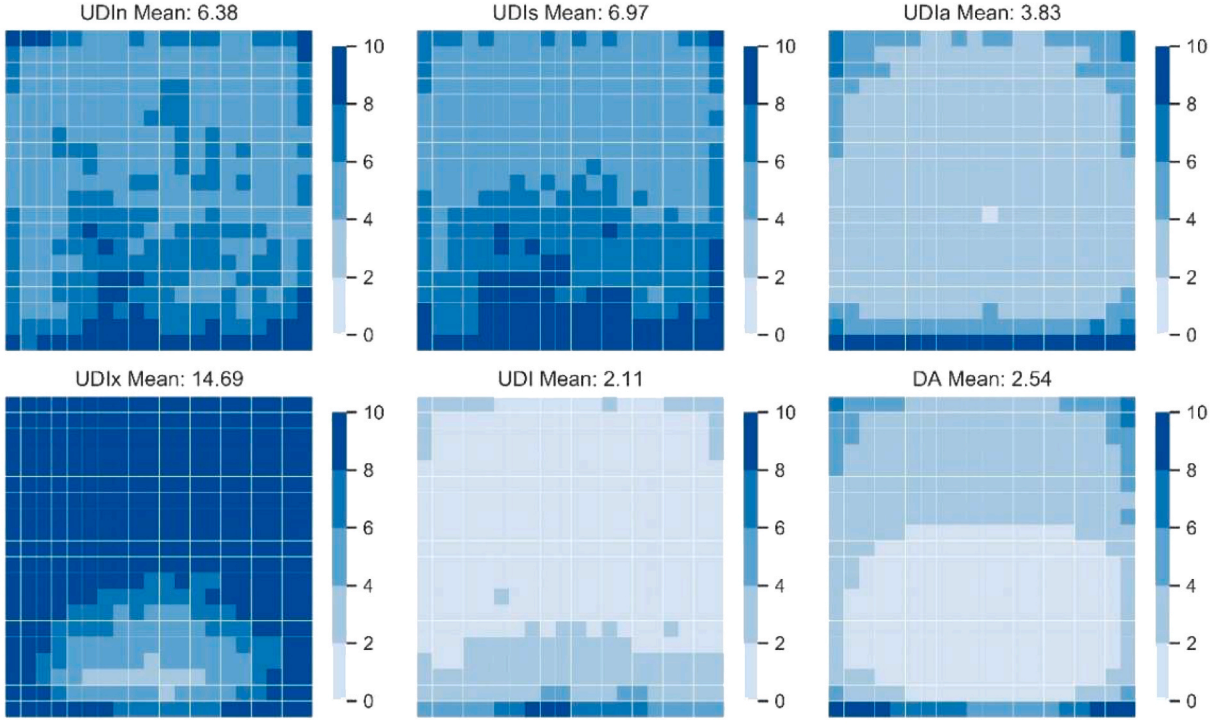
parameters. The retrieved prediction results can then be visualized in Rhino viewport, providing valuable daylight distribution information. With the existing Grasshopper plugin, various optimizations can easily be performed. For example, given room size, orientation, and default material parameters, the optimal window positions, and size can be determined within seconds.

A Dynamo script was created similarly. In addition, it is no longer necessary to manually specify the model inputs, as all these parameters are already present in the BIM model. By querying the model database, the complexity of the model is hidden from the user, further simplifying the building design workflow.

The developed Dynamo definition (Fig. 18) can automatically collect all the rooms defined in the current Revit model, extract the room sizes, determine window orientation and dimensions and, using the proxy prediction, the node results can be set back to Revit and displayed on the Revit viewport. Users can directly interact with elements such as walls and windows in Revit, and obtain intuitive daylight feedback. A drawback of the Dynamo script is that it is much slower than the Grasshopper



(a)Test MAE



(b)Test MAPE (%)

Fig. 14. Spatial MAE and MAPE (%) for higher resolution test data.

version. The problem resides in the core library of Dynamo; thus, no immediate solution is apparent. Nonetheless, the prediction is still considerably swifter than regular simulation, with a latency of a few seconds.

As predictions are made per room, coverage of the design space can be extended by combining different room variants into a complex building layout. This enables a wide range of building cases to be predicted without being constrained to the training samples. A building plan can be represented by combining multiple rooms. If any of the rooms fit the model prototype, the prediction can be made and

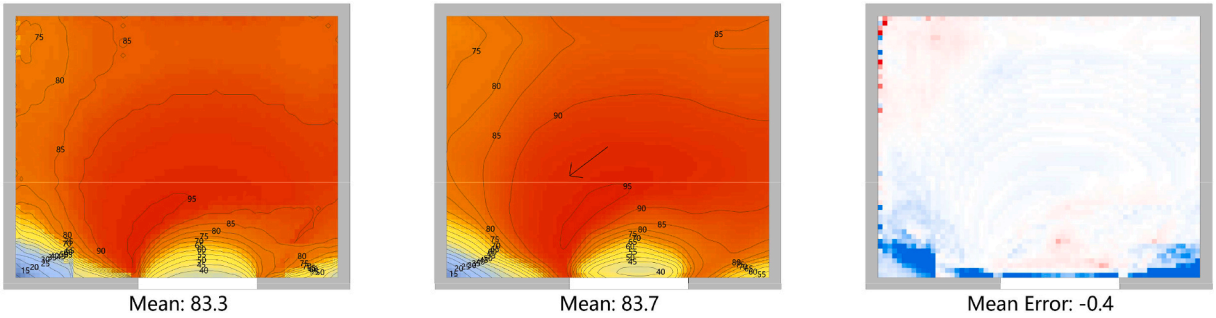


Fig. 15. Comparison between high resolution simulation and prediction, with an analysis grid of 0.1 m.

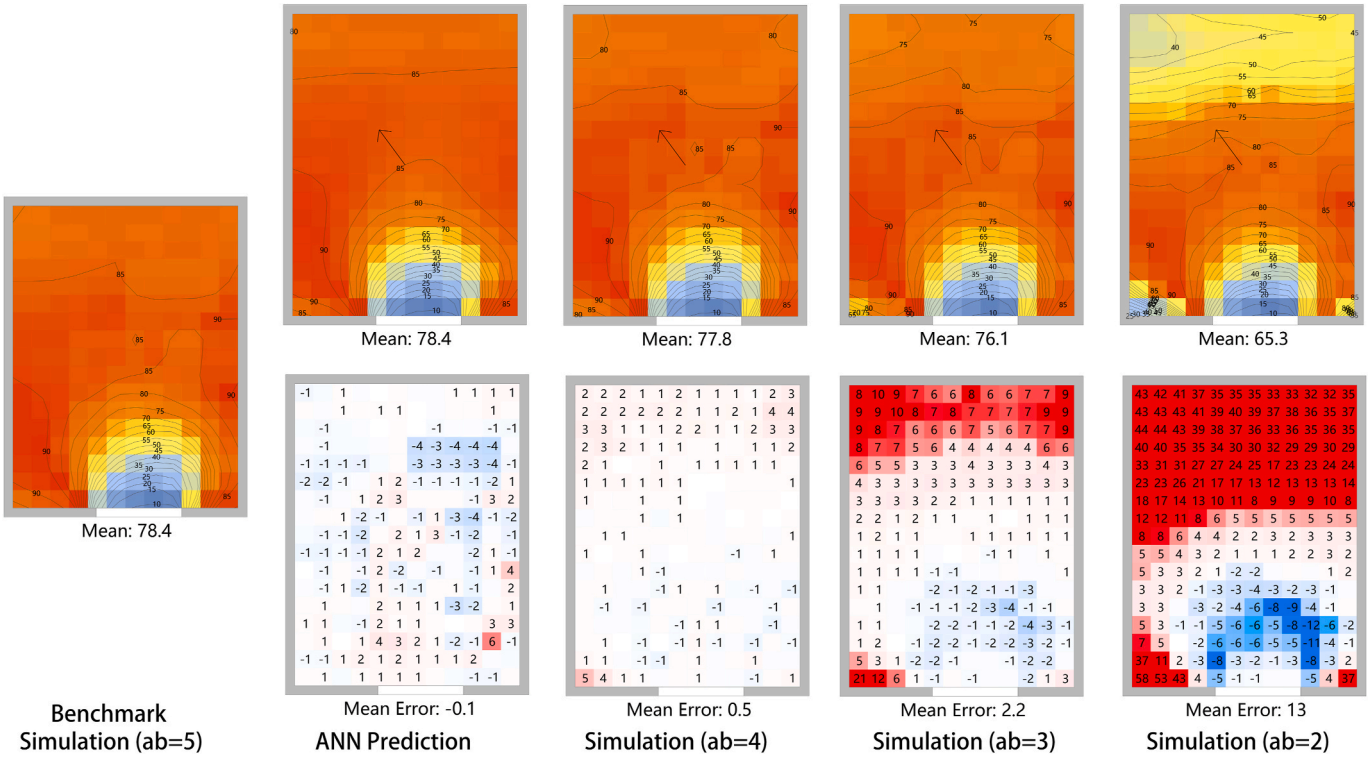


Fig. 16. Comparison of prediction and simulation results with varied ab setting.

Table 4
Time taken for ANN and simulation with different settings.

Area	52 m^2	98 m^2	222 m^2	905 m^2	2520 m^2
Number of Points	208	394	891	3618	10080
ANN Prediction	36 ms	56 ms	162 ms	593 ms	1.2 s
Simulation (ab = 5)	2.4 min	2.6 min	2.9 min	4.4 min	5.0 min
Simulation (ab = 2)	1.2 min	1.4 min	1.7 min	2.7 min	3.4 min
Speed up	4000x	2785x	1444x	445x	250x

*ANN run on the same CPU as simulation.

visualized. The normal design workflow is completely uninterrupted by this process, and no extra knowledge or effort is required. By allowing the streaming of data over the internet, this further eliminates system requirements and reduces configuration efforts to zero, at the cost of increased time for data transmission.

4. Conclusion

Traditional daylight simulation tools are too slow to use in the early design stage. Although machine learning-based methods have been demonstrated to provide fast and accurate predictions, the current applications are mainly focused on the post-construction phase, when the design variables are already determined, resulting in very limited generalization capability. Additional effort is required to apply these design tools in the early stages of the design process.

This study develops an ANN-based high granularity daylight prediction model for a generic parametric room that can represent a majority of office building spaces. A Python-based framework integration with a design platform is introduced to alleviate the setting up process. The combination of fast and accurate generic daylight prediction with close design platform integration makes early stage daylight evaluation feasible. The research addresses the gap in using machine learning methods for early stage indoor daylight prediction. The main conclusions are the following:

1. The proposed model is applicable to more diverse scenarios, has higher accuracy, and provides information with better granularity. The ability to predict daylight performance distribution over a whole space provides valuable insight for designing indoor furniture layout. This study improves model generalization capability through feature engineering and hyperparameter optimization. Parametric prototype

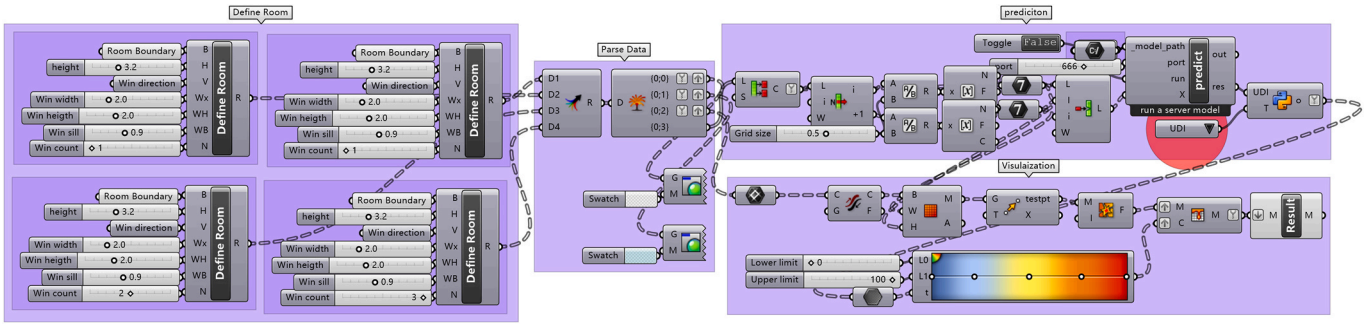


Fig. 17. The integration plugin developed in Grasshopper, including a parametric room module, a data parsing module, a ML prediction module and a result visualization model.

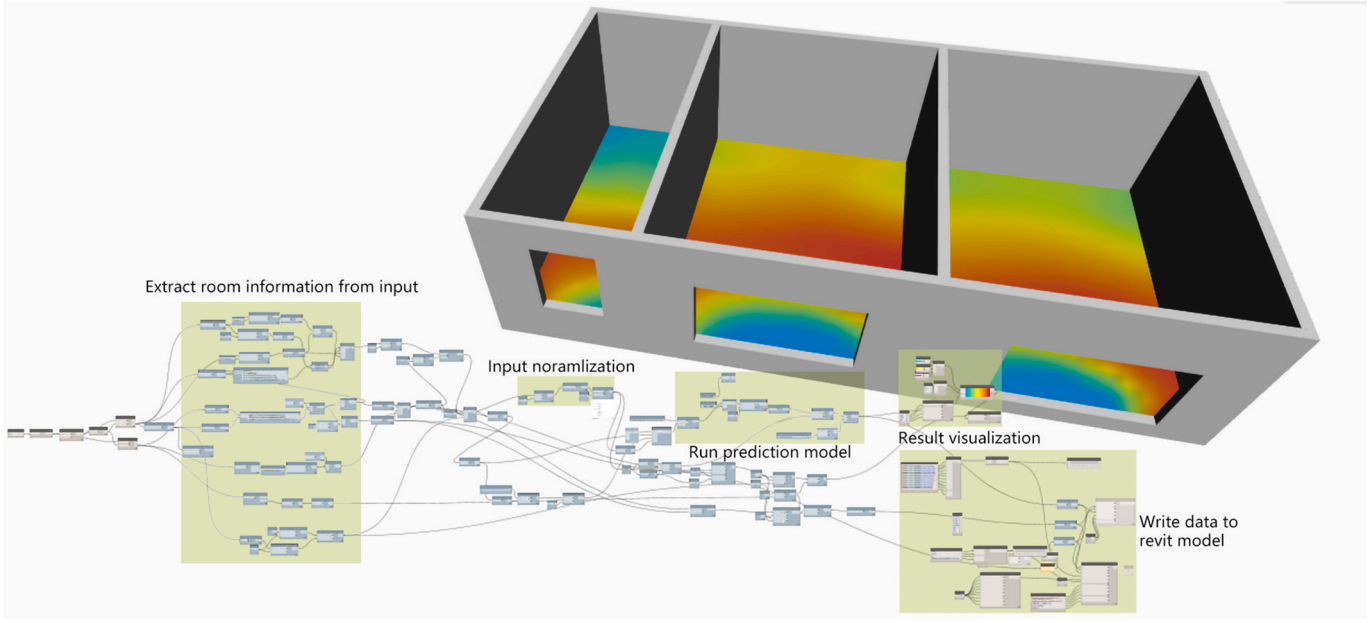


Fig. 18. An automatic information extraction and prediction plugin developed for a Revit model, structured similar to the Grasshopper plugin using different API.

rooms with a wide range of design variables are sampled and simulated as training datasets. The spatial relationship is explicitly defined by including sensor point local coordinates as input; hence, the trained model can predict DA and UDI for any point on the working plane instead of a few pre-defined ones, giving full indoor UDI distribution information.

2. Furthermore, the Bayesian optimization method is applied to find the hyperparameter combination with optimal generalization performance. It is found that five hidden layers, with around 800 neurons each, achieve the best test scores, while at least 1000 sample data points are required to achieve sufficient generalization. The final model attained R^2 scores of 0.988 and 0.996, MAE scores of 1.58 and 1.37, MAPE scores of 2.10% and 2.36% for UDI and DA_{300}, respectively, while being 250 times faster than the Daysim simulation engine.
3. The proposed integrated framework also reduces the learning cost for the non-expert user and minimizes the effort of setting up the simulation model during the early stages of design. The daylight prediction model can be accessed via the internet, avoiding the need for any configuration at the user end. This simple and intuitive feedback can be helpful for early design space exploration or educational purposes. More advanced users can also combine the ANN model with other tools on the platform, such as conducting building form or fenestration optimization.

A major shortcoming of this method is the absence of environmental obtrusion. The model works well in a low-density city context; however, in a high-density environment, the prediction results become unreliable. This is a shared problem for all statistically based models, since the environmental conditions vary drastically from site to site; currently, there is no straightforward way of encoding them as training input. This is a challenging problem that requires further investigation.

Another shortcoming relates to site specific data. As the neural network model is trained on data simulated with local weather data, the prediction is only valid for such data. For buildings located in a different city, a new model must be trained with corresponding weather data. This problem can be addressed by crowdsource training, using all available EPW data on the internet. Ideally, a government-backed site, such as the DOE, would host all pre-trained models as reference prediction models, and designers could access and download the model closest to their site's location.

Several future improvements are possible. The proposed workflow can be extended by adding more parametric prototype rooms, e.g., a top lighted room or a corner room with two windowed walls. Extra spatial distribution can be combined with the existing automatic indoor furniture layout algorithm. The main challenge, as mentioned, is the inclusion of the outside environment since this is a vital step toward more generic use.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was funded by the National Natural Science Foundation of China (Grant No. 52078157), the National Natural Science Foundation of China (Grant No. 51938003), and the China Postdoctoral Science Foundation (Grant No. 2017M621276).

References

[1] P. Boyce, C. Hunter, O. Howlett, The Benefits of Daylight through Windows, Rensselaer Polytechnic Institute, Troy, New York, 2003.
[2] M.B. Aries, M.P. Aarts, J. van Hoof, Daylight and health: a review of the evidence and consequences for the built environment, Light. Res. Technol. 47 (1) (2015) 6–27, https://doi.org/10.1177/1477153513509258.
[3] M.N. Mead, Benefits of sunlight: a bright spot for human health, Environ. Health Perspect. (2008), https://doi.org/10.1289/ehp.116-a160.
[4] I. Turan, A. Chegut, D. Fink, C. Reinhart, The value of daylight in office spaces, Build. Environ. 168 (2020), 106503, https://doi.org/10.1016/j.buildenv.2019.106503.
[5] C.S. Kim, S.J. Chung, Daylighting simulation as an architectural design process in museums installed with toplights, Build. Environ. 46 (1) (2011) 210–222, https://doi.org/10.1016/j.buildenv.2010.07.015.
[6] T. Han, Q. Huang, A. Zhang, Q. Zhang, Simulation-based decision support tools in the early design stages of a green building—a review, Sustainability 10 (10) (2018) 3696.
[7] M. Ayoub, 100 Years of daylighting: a chronological review of daylight prediction and calculation methods, Sol. Energy 194 (2019) 360–390, https://doi.org/10.1016/j.solener.2019.10.072.
[8] Y. Wei, X. Zhang, Y. Shi, L. Xia, S. Pan, J. Wu, X. Zhao, A review of data-driven approaches for prediction and classification of building energy consumption, Renew. Sustain. Energy Rev. 82 (2018) 1027–1047, https://doi.org/10.1016/j.rser.2017.09.108.
[9] Z. Wang, R.S. Srinivasan, A review of artificial intelligence based building energy use prediction: contrasting the capabilities of single and ensemble prediction models, Renew. Sustain. Energy Rev. 75 (September 2015) (2017) 796–808, https://doi.org/10.1016/j.rser.2016.10.079.
[10] J. Ngarambe, A. Irakoze, G.Y. Yun, G. Kim, Comparative performance of machine learning algorithms in the prediction of indoor daylight illuminances, Sustainability 12 (11) (2020) 4471, https://doi.org/10.3390/su12114471.
[11] S. Walker, W. Khan, K. Katic, W. Maassen, W. Zeiler, Accuracy of different machine learning algorithms and added-value of predicting aggregated-level energy performance of commercial buildings, Energy Build. 209 (2020) 109705, https://doi.org/10.1016/j.enbuild.2019.109705.
[12] S. Alawadi, D. Mera, M. Fernández-Delgado, F. Alkhabbas, C.M. Olsson, P. Davidsson, A comparison of machine learning algorithms for forecasting indoor temperature in smart buildings, Energy Systems (2020) 1–17, https://doi.org/10.1007/s12667-020-00376-x.
[13] X.J. Luo, L.O. Oyedele, A.O. Ajayi, O.O. Akinade, Comparative study of machine learning-based multi-objective prediction framework for multiple building energy loads, Sustainable Cities and Society 61 (2020) 102283, https://doi.org/10.1016/j.scs.2020.102283.
[14] Z. Liu, D. Wu, Y. Liu, Z. Han, L. Lun, J. Gao, G. Cao, Accuracy analyses and model comparison of machine learning adopted in building energy consumption prediction, Energy Explor. Exploit. 37 (4) (2019) 1426–1451.
[15] Y. Wei, L. Xia, S. Pan, J. Wu, X. Zhang, M. Han, Q. Li, Prediction of occupancy level and energy consumption in office building using blind system identification and neural networks, Appl. Energy 240 (2019) 276–294, https://doi.org/10.1016/j.apenergy.2019.02.056.
[16] S. Seyedzadeh, F.P. Rahimian, S. Oliver, S. Rodriguez, I. Glesk, Machine learning modelling for predicting non-domestic buildings energy performance: a model to support deep energy retrofit decision-making, Appl. Energy 279 (2020), 115908, https://doi.org/10.1016/j.apenergy.2020.115908.
[17] J. Bedi, D. Toshniwal, Deep learning framework to forecast electricity demand, Appl. Energy 238 (2019) 1312–1326, https://doi.org/10.1016/j.apenergy.2019.01.113.
[18] J. Zhao, X. Liu, A hybrid method of dynamic cooling and heating load forecasting for office buildings based on artificial intelligence and regression analysis, Energy Build. 174 (2018) 293–308, https://doi.org/10.1016/j.enbuild.2018.06.050.
[19] Z. Wang, T. Hong, M.A. Piette, Building thermal load prediction through shallow machine learning and deep learning, Appl. Energy 263 (2020) 114683, https://doi.org/10.1016/j.apenergy.2020.114683.
[20] M. Ayoub, A review on machine learning algorithms to predict daylighting inside buildings, Sol. Energy 202 (2020) 249–275, https://doi.org/10.1016/j.solener.2020.03.104.
[21] T. Kazanasmaz, M. Günaydin, S. Binol, Artificial neural networks to predict daylight illuminance in office buildings, Build. Environ. 44 (8) (2009) 1751–1757, https://doi.org/10.1016/J.BUILDENV.2008.11.012.
[22] J. Hu, S. Olbina, Illuminance-based slat angle selection model for automated control of split blinds, Build. Environ. 46 (3) (2011) 786–796, https://doi.org/10.1016/j.buildenv.2010.10.013.
[23] V. Logar, Ž. Kristl, I. Škrjanc, Using a fuzzy black-box model to estimate the indoor illuminance in buildings, Energy Build. 70 (2014) 343–351, https://doi.org/10.1016/j.enbuild.2013.11.082.
[24] M.W. Ahmad, J.L. Hippolyte, M. Mourshed, Y. Rezgui, Random forests and artificial neural network for predicting daylight illuminance and energy consumption, in: Proceedings of the 15th IBPSA Conference, 2017, pp. 7–9.
[25] M. Beccali, M. Bonomolo, G. Ciulla, V. Lo Brano, Assessment of indoor illuminance and study on best photosensors' position for design and commissioning of Daylight Linked Control systems. A new method based on artificial neural networks, Energy 154 (2018) 466–476, https://doi.org/10.1016/j.energy.2018.04.106.
[26] S. Jain, V. Garg, A review of open loop control strategies for shades, blinds and integrated lighting by use of real-time daylight prediction methods, Build. Environ. 135 (2018) 352–364.
[27] S. Yeon, B. Yu, B. Seo, Y. Yoon, K.H. Lee, ANN based automatic slat angle control of Venetian blind for minimized total load in an office building, Sol. Energy 180 (2019) 133–145, https://doi.org/10.1016/j.solener.2019.01.027.
[28] Z. Luo, C. Sun, Q. Dong, A daylight-linked shading strategy for automated blinds based on model-based control and Radial Basis Function (RBF) optimization, Build. Environ. (2020), 106854, https://doi.org/10.1016/j.buildenv.2020.106854.
[29] K. Radziszewski, M. Waczyńska, June). Machine learning algorithm-based tool and digital framework for substituting daylight simulations in early-stage architectural design evaluation, in: Proceedings of the Symposium on Simulation for Architecture and Urban Design, 2018, pp. 1–7.
[30] C. Lorenz, M. Packianather, A.B. Spaeth, C. B. De Souza, Artificial neural network-based modelling for daylight evaluations, in: The Symposium on Simulation for Architecture and Urban Design, SimAUD 2018), Delft, Netherlands, 2018, https://doi.org/10.22360/simaud.2018.simaud.002.
[31] V.R.L. Verso, G. Mihaylov, A. Pellegrino, F. Pellerey, Estimation of the daylight amount and the energy demand for lighting for the early design stages: definition of a set of mathematical models, Energy Build. 155 (2017) 151–165, https://doi.org/10.1016/j.enbuild.2017.09.014.
[32] E. Nault, P. Moonen, E. Rey, M. Andersen, Predictive models for assessing the passive solar and daylight potential of neighborhood designs: a comparative proof-of-concept study, Build. Environ. 116 (2017) 1–16, https://doi.org/10.1016/j.buildenv.2017.01.018.
[33] M. Ayoub, A multivariate regression to predict daylighting and energy consumption of residential buildings within hybrid settlements in hot-desert climates, Indoor Built Environ. 28 (6) (2019) 848–866, https://doi.org/10.1177/1420326X18798164.
[34] J.L. Deutsch, C.V. Deutsch, Latin hypercube sampling with multidimensional uniformity, J. Stat. Plann. Inference 142 (3) (2012) 763–772, https://doi.org/10.1016/j.jspi.2011.09.016.
[35] Sahil Moza, sahilm89/lhsmdu: Latin Hypercube Sampling with Multi-Dimensional Uniformity (LHSMDU): Speed Boost Minor Compatibility Fixes (Version 1.1.1), Zenodo, 2020, July 3, https://doi.org/10.5281/zenodo.3929531.
[36] C.F. Reinhart, O. Walkenhorst, Validation of dynamic RADIANCE-based daylight simulations for a test office with external blinds, Energy Build. 33 (7) (2001) 683–697, https://doi.org/10.1016/S0378-7788(01)00058-5.
[37] P.I. Frazier, A Tutorial on Bayesian Optimization, 2018 arXiv preprint arXiv:1807.02811.
[38] Tim Head, Manoj Kumar, Holger Nahrstaedt, Gilles Louppe, Iaroslav Shcherbatyi, scikit-optimize/scikit-optimize (Version v0.8.1), Zenodo, 2020, September 4, https://doi.org/10.5281/zenodo.4014775.
[39] F. Chollet, others, Keras. https://keras.io, 2015.
[40] J. Snoek, H. Larochelle, R.P. Adams, Practical bayesian optimization of machine learning algorithms, in: Advances in Neural Information Processing Systems, 2012, pp. 2951–2959.
[41] Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.
[42] K. Negendahl, Building performance simulation in the early design stage: an introduction to integrated dynamic models, Autom. ConStruct. 54 (2015) 39–53, https://doi.org/10.1016/j.autcon.2015.03.002.
[43] T. Østergård, R.L. Jensen, S.E. Maagaard, Building simulations supporting decision making in early design – a review, Renew. Sustain. Energy Rev. 61 (2016) 187–201, https://doi.org/10.1016/J.RSER.2016.03.045.
[44] M.S. Roudsari, M. Pak, A. Smith, Ladybug: a parametric environmental plugin for grasshopper to help designers create an environmentally-conscious design, in: Proceedings of the 13th International IBPSA Conference Held in Lyon, France Aug, 2013, August, pp. 3128–3135.
[45] N.L. Jones, C.F. Reinhart, Speedup potential of climate-based daylight modelling on GPUs, in: Building Simulation Conference, 2017, August, pp. 1438–1447.