Advanced Engineering Informatics 38 (2018) 81–90

Contents lists available at ScienceDirect

Advanced Engineering Informatics

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

Deep-learning neural-network architectures and methods: Using component-based models in building-design energy prediction

Sundaravelpandian Singaravel[a,*], Johan Suykens[b], Philipp Geyer[a]

[a] Architectural Engineering Division, KU Leuven, Belgium
[b] ESAT-STADIUS, KU Leuven, Belgium

ARTICLE INFO

Keywords:
Performance gap
Sustainability
Building performance simulation
Transfer learning
Multi-task learning
LSTM

ABSTRACT

Increasing sustainability requirements make evaluating different design options for identifying energy-efficient design ever more important. These requirements demand simulation models that are not only accurate but also fast. Machine Learning (ML) enables effective mimicry of Building Performance Simulation (BPS) while generating results much faster than BPS. Component-Based Machine Learning (CBML) enhances the capabilities of the monolithic ML model. Extending monolithic ML approach, the paper presents deep-learning architectures, component development methods and evaluates their suitability for space exploration in building design. Results indicate that deep learning increases the performance of models over simple artificial neural network models. Methods such as transfer learning and Multi-Task Learning make the component development process more efficient. Testing the deep-learning model on 201 new design cases indicates that its cooling energy prediction (R^2: 0.983) is similar to BPS, while errors for heating energy predictions (R^2: 0.848) are higher than BPS. Higher heating energy prediction error can be resolved by collecting heating data using better design space sampling methods that cover the heating demand distribution effectively. Given that the accuracy of the deep-learning model for heating predictions can be increased, the major advantage of deep-learning models over BPS is their high computation speed. BPS required 1145 s to simulate 201 design cases. Using the deep-learning model, similar results can be obtained in 0.9 s. High computation speed makes deep-learning models suitable for design space exploration.

1. Introduction

Building design is exploratory in nature, whereby architects rely on experiential knowledge [1]. Stringent energy-efficiency requirements call for high-performance building design and operation. Building Performance Simulation (BPS) helps develop buildings that adhere to these demands. Results of BPS are used by designers (architects and engineers) to take appropriate decisions.

The utilisation and generation of BPS results have posed challenges in integrating BPS in the design process. The first challenge is that designers may lack fundamental knowledge of physical phenomena in BPS, making it challenging for them to understand simulation results and take appropriate design decisions [1,2]. Designers also perceive simulations as theoretical. Lack of knowledge and perception tend to make them rely more on rules of thumb and experiential knowledge for decisions if simulations do not provide instantaneous results. Nonetheless, designers are interested in using simulations to improve their designs [1] and their awareness of the design space through simulations allows them to take appropriate design decisions. Simulating multiple design options using BPS can be time consuming. Therefore, it is critical to obtain rapid simulation results in order to avoid the use of conventional knowledge, which may not always be valid, and to explore the entire design space.

The second challenge is that the quality of BPS results depends on the simplification methods applied in the model [3], on the quality of tools and on the skill level of the simulation analyst [4]. The availability of time to perform simulations varies at different design stages. Simulation time affects the complexity of the model used. Developing complex models requires a great deal of time as well as information. Therefore, simple models are used in early design stages and more detailed models as the design stage progresses. Upon design completion, a detailed model is used to validate whether the design complies with energy efficiency requirements. It is therefore important that the simple models are good representations of the detailed model. The ability to simplify a model accurately depends on the technical skills of the analyst. The skill level determines the simplification methods utilised by the analyst to obtain simple BPS. Simplification methods can range from use of building mass to represent building geometry to

* Corresponding author.
E-mail address: sundar.singaravel@kuleuven.be (S. Singaravel).

https://doi.org/10.1016/j.aei.2018.06.004
Received 10 November 2017; Received in revised form 16 May 2018; Accepted 11 June 2018
Available online 18 June 2018
1474-0346/ © 2018 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/BY-NC-ND/4.0/).

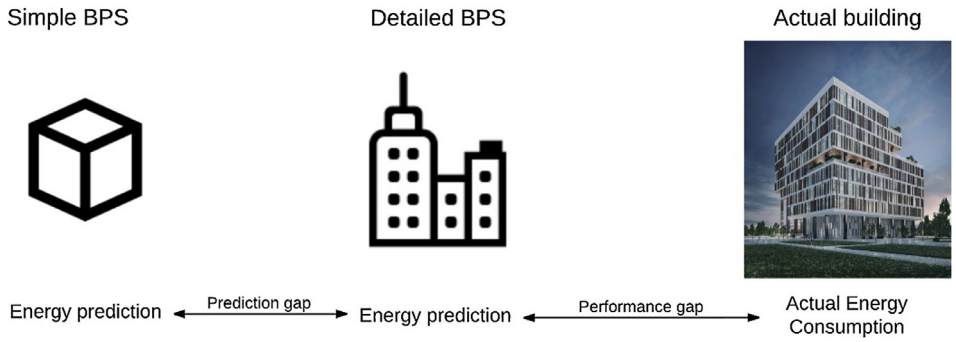

Fig. 1. Illustration of gaps.

seasonal efficiencies for representing Heating Ventilation and Air-Conditioning (HVAC) system. The continuum between simple and detailed models (i.e. simplification method utilised) can induce prediction gap (see Fig. 1) [3]. Prediction gap is the difference between predictions of simple BPS model and detailed BPS model. Definition of prediction gap is comparable to "inter-model variability" defined by Dronkelaar et al. (2016) [5]. High prediction gap compromises the reliability of the decision taken using simple BPS. Wrong decision with simple BPS could result in expensive design changes. Hence, it is important to have models with low- or no prediction gap to ensure the reliability of a design decision.

Furthermore, assumptions (e.g. about occupancy) made in a detailed simulation model are based on the simulation analyst's perceptions of the building [1]. When these perceptions are incorrect, performance gaps are observed [6,7]. The performance gap is defined as the difference between actual energy consumption and design stage energy prediction [6,7]. The key difference between a prediction and a performance gap is that a prediction gap can be reduced by means of correct simplifications, while a performance gap is difficult to reduce during the design stage as it results from uninformed assumptions. Prediction and performance gaps also influence the credibility of BPS, which, in turn, influences designers' perceptions of BPS and its benefits for the design process. Therefore, it is crucial that these gaps be reduced.

The above challenges result in unexplored low-energy design solutions. To limit designers' reliance on conventional methods and change designers' perspective on simulations, analysts must provide reliable and rapid results. Hence, prediction accuracy and computational time[1] to perform simulations play vital roles. Machine Learning (ML) provides the capability of extracting knowledge from data by recognising patterns within them [8]. ML models enable us to capture interactions observed within detailed BPS by means of simple input structures [9]. Is it possible to develop ML models that might complement BPS to overcome these challenges? This paper aims to provide and evaluate deep-learning (a sub-domain of ML) methods that enable the skilled analyst to develop reliable and rapid models for design decisions. To make a case for complementary models (meta/surrogate models) for a BPS model, we start the discussion with basic questions such as 'what is the purpose of a model?' and 'how do ML models fit the grand scheme of sustainable building design?' The discussion is followed by an evaluation of deep-learning methods to highlight their potential for complementing BPS in building design.

2. Background

A model providing the necessary information for design decisions is considered a good model [2]. Generating all of the required information using BPS is possible. Within the constraints of a design program, however, the effort required to obtain necessary information may not be feasible. To address the above challenge, models that emulate and substitute detailed physical simulations are used. In engineering, these models are also referred to as meta-models or surrogate models. ML is an effective way to develop meta/surrogate models. Artificial Neural Network (ANN) is one such model so in this context ML model is synonymous with meta/surrogate model. Meta-models typically have low computation time compared to BPS [10]. Low computation time with sufficient accuracy is ideal for situations where rapid performance feedback is required and also enable the introduction of advanced design optimisation techniques [11,12].

ANN and Support Vector Machines (SVM) are some of the popular ML algorithms used to predict the energy performance of a building [12–20]. Other potential algorithms applied in the context of building energy prediction are decision tree, random forest and multivariate adaptive regression splines [21–23]. These algorithms are applied for rapid design decision energy predictions, for design optimisation based on energy predictions and for operational energy predictions. In all of these applications, ML models provide one response or output and no extra information to support this response. Hence, analysts have to rely on their knowledge to derive possible reasons to support the response in order to make appropriate design recommendations. The authors have proposed the use of a component-based approach to overcome this limitation (i.e. lack of supporting information). The next section contains a brief description of the approach.

Deep learning has outperformed other ML algorithms in domains such as image classification. However, the application of deep-learning algorithms for building energy predictions is limited [24], because the benefits of deep learning are observed in very large datasets, while its performance on smaller datasets is similar to other ML methods. One application of deep learning found in the literature is for short-term cooling energy predictions [25]. Cheng Fan et al. (2017) state that for the developed case and dataset, deep learning did not provide many advantages over traditional methods. However, the features extracted (or representations learned) using deep auto-encoders resulted in significant improvements in prediction performance [25]. Initial research indicated that simple ANN could provide predictions with accuracy similar to deep-learning models [26]. However, initial research only evaluated simple datasets. Data covering broader design parameters can induce more non-linearity and require larger datasets to effectively cover the design space. This paper expands the dataset to evaluate deep-learning architectures and development methods. Finally, implementing models using deep-learning architectures can result in a model size of polynomial nature, while the same representation using a shallow model results in an exponentially sized model [27].

3. Deep learning for the component-based approach

3.1. What is and why use a component-based approach?

ML models are confined to the development dataset's distribution.

[1] Including development and simulation time.

Cases that are not sufficiently covered by the training data will not be predicted correctly. At the same time, no two building designs are the same and developing monolithic ML models for predicting the performance of a whole building at once for design applications requires a lot of data and highly complex parameterisation to cover the entire design space. For example, modelling a building with three floors with varying floor areas and geometries will be difficult to parameterize, and a large number of parameters will make it difficult to find enough representative data. Hence, a monolithic ML approach limits the applicability of ML models for design applications to simplified building representations only.

We propose a component-based machine learning (CBML) approach to overcome the limitations of a monolithic ML model [28,29]. A component is a model that defines a sub-element of a building (wall or room). The response of components, e.g. the heat flows through walls, floor, roof, etc., is aggregated to obtain the building energy performance. The main reasons for the use of components are:

(1) Components extend the reusability of ML models for design situations different from the training situation.
(2) Designers/analyst are able to quantify the reason for a prediction, allowing for finer analysis of a prediction by looking at the quantities between components.
(3) Models can be directly linked to building information modelling (BIM) elements.
(4) Monitoring data can be used in the development of components to achieve more realistic predictions.

The component-based ML approach very well supports the design process of buildings, especially, in early phases. Components are plugged together as needed in the process and prediction is performed immediately without additional modelling or significant computation time. The component paradigm equals that of modular simulation methods, such as Modelica or TRNSYS [30,31]. However, the prediction is not based on physical simulation but on ML models that jointly predict relevant performance indicators.

Fig. 2 shows potential model structures that can be developed based on need in design stages. Fig. 2a shows a model component structure that provides information on total energy demand but also heat flows that result in the total energy demand. Fig. 2b shows a simpler model structure that provides information on total energy demand together with monthly zone-level energy demands. With components, interactions are captured hierarchically rather than iteratively. Hence, the final prediction (i.e. total energy) relies on general abstract representation (heat flows in Fig. 2a and zone heating and cooling load in Fig. 2b) of a building rather than specific design parameters. Initial research shows that this abstraction method results in reusability of ML models within a design. The limitations and potential benefits of this abstraction needs to be researched further.

Given the benefits of the component-based approach, its challenge is the need to model multiple components. Depending on the component-based model structure, developing these models could become time consuming. In this paper we therefore also explore possible methods that will reduce development time. Since the main objective of the paper is to evaluate the potential of the deep-learning methods and highlight potential benefits for building design, only the Fig. 2b model structure is presented with simple building design.

3.2. Brief theory of evaluated deep-learning methods

Deep learning is an extension of ANN providing a multi-level representation of data that enhances better performing models and eliminates the need for manual representation learning [32]. From a structural point of view, component-based ML (CBML) is a form of deep learning because deep learning reduces the input data to a general data subset before making the final predictions. CBML reduces the input data to process-related data (in this case, zone monthly energy demand), which is used for final predictions. However, there are crucial differences between CBML and deep learning. The output of component is technically interpretable whereas outputs between layers of a deep-learning model are not. Furthermore, deep-learning networks organise the weightings and thus connections between layers through a training process. Component-based networks use pre-defined structures with connections guided by engineering.

Can the layers from a deep-learning model be reused in similar context? One powerful capability of deep learning is transfer learning; representations learned in one situation can be transferred to similar situation. These transferable layers, through Multi-Task Learning (MTL), can also act as shared layers [33,34]. These methods can potentially reduce the development time and improve reusability of ML models. We believe that flexibility in model architecture for state-of-the-art performance combined with methods such as transfer learning and MTL are ideal for a component-based ML approach. The deep learning techniques evaluated in this paper are (1) deep-learning architectures that use long short-term memory (LSTM) layers, (2) transfer learning and (3) Multi-Task Learning (MTL). In this section, we provide a brief description on these methods.

3.2.1. Deep-learning architecture

Deep-learning architectures are comprised of three major layers: the input layer, hidden layers and the output layer. The number of hidden layers defines the depth of the architecture. Depending on the type of hidden layers used, different non-linear functions can be learned. Eqs. (1) and (2) represent a simple ANN model. Eq. (1) (hidden layer) represents the non-linearity observed within the data, where the activation function determines the characteristics of the non-linearity represented by the model. Eq. (2) (output layer) translates this non-linearity characteristic into a prediction (e.g. energy prediction). A simple deep-learning model has multiple hidden layers (i.e. Eq. (1)). In this section, we provide a brief description of activation function and hidden and output layers.

$$A_1 = Activation\ (W_1 X + b_1) \tag{1}$$

$$\hat{y} = W_2 A_1 + b_2 \tag{2}$$

3.2.1.1. Activation functions. Activation functions in the hidden layer help in mapping the non-linearity relationship between input and output. Commonly used activation functions in hidden layers are sigmoid and hyperbolic tangent function (i.e. tanh). There is no rule for applying specific activation functions. Different activation functions need to be evaluated for specific datasets. In this paper, we use the rectified linear (also referred as 'relu') activation function (see Eq. (3)) because it facilitates model training using gradient-based optimisation methods [8].

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geqslant 0 \end{cases} \tag{3}$$

3.2.1.2. Hidden layer. Eq. (1) represents a simple hidden layer. The complexity of each hidden layer is defined by the number of parameters used to represent it. Number of parameters is controlled by hyper-parameters, which are the number of hidden units and the parameter of the L2 regularisation. The number of hidden units indicates the number of parameters (i.e. weights) in each layer and L2 regularisation reduces the magnitude of these parameters to avoid over-fitting [8]. Tuning hyper-parameters is important to obtain a good model fit.

3.2.1.2.1. LSTM layer. In this paper, we are modelling monthly energy demand, whereby the energy demand from February is influenced by weather conditions in January. This influence is also referred to as autocorrelation. For data with autocorrelation, the use of

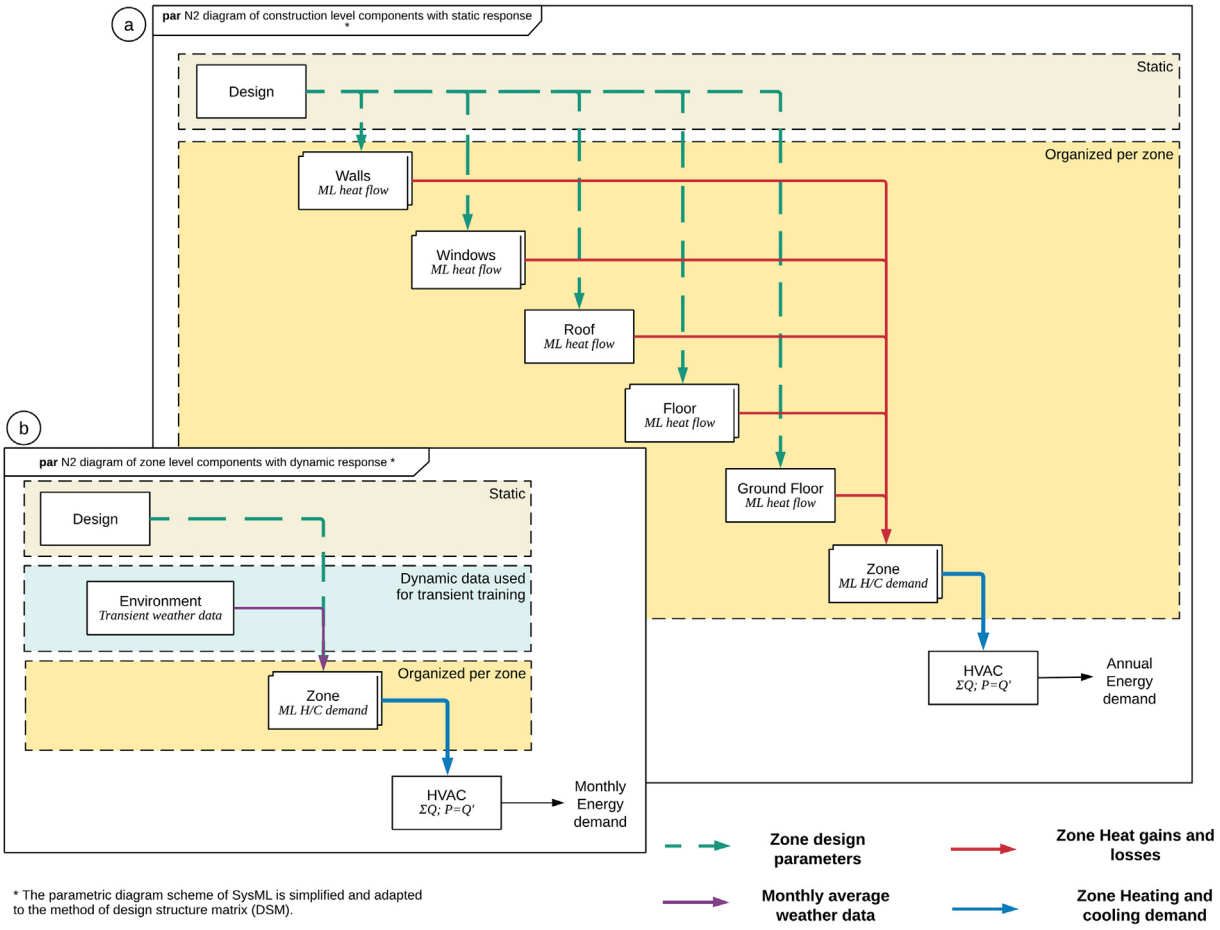

Fig. 2. Possible component-based approach machine learning model structures.

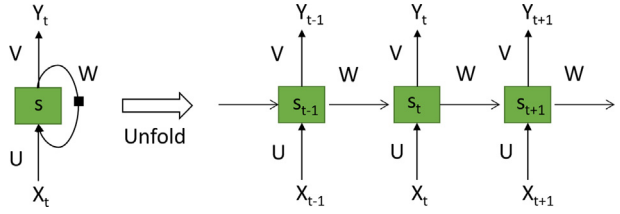

Fig. 3. Recurrent layer and unfolding in time [32].

Eq. (1) as a hidden layer may not be efficient. A hidden layer that remembers the output of the previous prediction for future predictions may be more appropriate. Remembering previous predictions can be achieved using recurrent hidden layers. Recurrent layers unfolded in time can be considered as deep feed-forward network [32,35]. Fig. 3 shows the workings of a recurrent hidden layer, where predictions of the hidden layer at x_{t-1} is used during predictions at x_t and so on.

LSTM layers are a type of recurrent layer that can retain information from a longer time interval. LSTM layers consist of memory cell blocks (referred to as hidden units in simple hidden layers) that facilitates information storage [36]. Currently, LSTM networks are used predominantly in natural language processing [37–39], acoustic modelling [40] and speech recognition [41]. In this paper, we evaluate the effectiveness of LSTM layers for modelling auto-correlated data (i.e. monthly energy).

3.2.1.3. Output layer. Typically, for regression problems, an output consists of an identity activation (also referred to as linear activation). Identity activation makes negative predictions possible. Since cooling and heating energy demands are non-negative response values with a minimum of 'zero,' identity activation is not suitable for this application. Instead, the rectified linear activation function is utilised.

3.2.2. Transfer learning and multi-task learning

During the training process, hidden layers learn a non-linear representation by optimising model weights while minimising the loss function. A simple example of a representation is shown in Eq. (1), where 'W_1' and 'b_1' are weights of a layer and 'X' is the layer input. The non-linear representation learned can be reused to develop models applied in similar conditions. The process of reusing layers in similar situations is called transfer learning. In this case, the representation encodes the characteristic of the building in terms of its energy performance. Heating and cooling ML models have the same inputs and are applied in similar contexts. This means that non-linear representations learned by heating models can be reused while developing cooling ML models (see Fig. 4) and only the final layer, i.e. Eq. (2) will have to be trained thus making the development time shorter.

In Multi-Task Learning (MTL), Eq. (1) acts as a shared hidden layer and each component has its own output layer (see Fig. 5). The approach of using shared representations for related tasks is referred to as Multi-Task Learning (MTL) [33,34]. Successful application of MTL includes natural language processing, speech recognition, computer vision and drug discovery [33,42,43]. Using MTL approach, both models are developed simultaneously. A model architecture with shared/common layers enabled more compact model representation and faster development time.

We evaluate these aspects of deep learning in the following sections. More details about CBML, such as different building design situations and alternative component decompositions, are presented by Geyer and Singaravel (2017) [29].

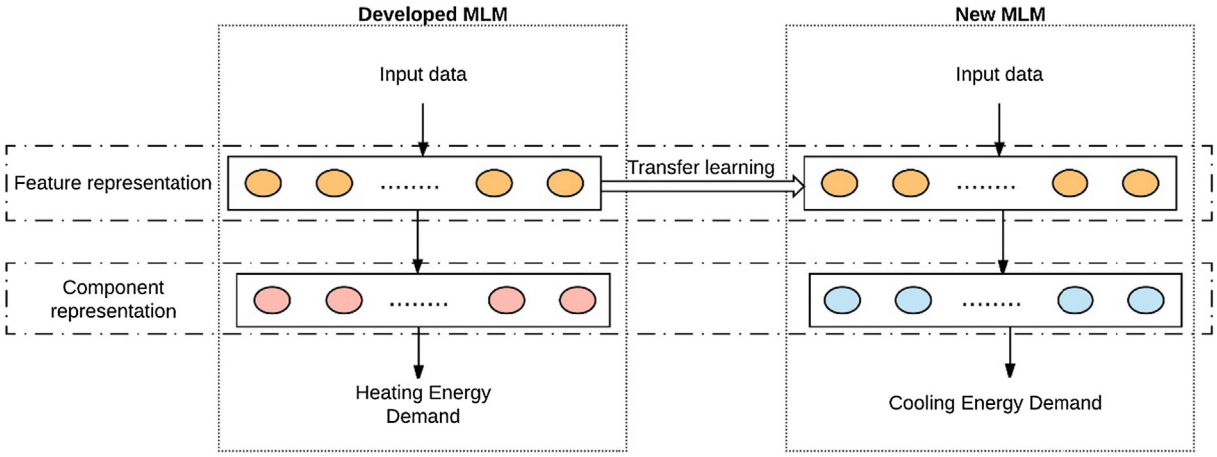

Fig. 4. Transfer learning of feature representation.

4. Evaluating deep learning for building design

The evaluation of deep-learning algorithms and development method starts by developing a Component-Based ML (CBML) model structure that can be used to evaluate different building designs. This is followed by evaluating (1) the performance of different deep-learning architectures on the heating ML model, (2) the effectiveness of transfer learning for the cooling ML model, (3) the effectiveness of MTL for modelling both heating and cooling demand, and (4) the suitability of deep-learning models for design stage applications.

4.1. Representation of the CBML building energy model (BEM)

Fig. 2b shows the component-based ML model structure used in this paper. The developed heating and cooling ML models represent a thermal zone's demand. The monthly heating and cooling demand of a thermal zone are aggregated to obtain the total building energy. The thermal zone demand details are obtained by parametric simulation of a two-story building model that resembles an actual building as shown in Fig. 6.

4.1.1. Data description

Training data is generated for a two-story building (i.e. three floors building) using parametric simulations with Brussels weather condition. Monthly simulation data is obtained using EnergyPlus simulation software (version 8.7). Design parameters sampled for data generation and included ML model inputs are shown in Table 1. Furthermore, by adjusting the height of the zones from ground parameter in Table 1, heating and cooling demand of higher levels are generated. Adjusting zone height is done to extend the training distribution to buildings with more stories. For the design parameters shown in Table 1, 800 design combinations are generated using the Latin hypercube sampling scheme (a near-random number generation method for multidimensional distribution). 800 design combinations resulted in 9600 training datasets. Major assumptions in the simulation model are the use of ideal HVAC calculations, HVAC schedule is always used, and the building is always occupied with fixed an internal gain scenario. Internal gains consist of occupancy gain of 125 W/person with an occupancy density of 10 m^2/person, lighting gain of 10 W/m^2 and equipment gain of 15 W/m^2. Use of Brussels weather condition suggests the generated data to be heating dominated. However, the utilised internal gains result in cooling dominated building energy data.



Fig. 6. Actual building that resembles the training building model (Photo courtesy of EnergyVille/Nathalie Belmans).

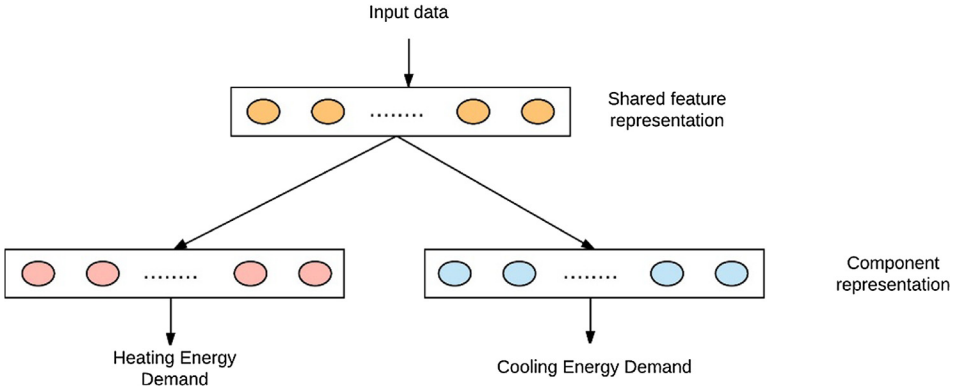

Fig. 5. Multi-Task representation.

Table 1
Training model parameterisation.

		Unit	min	max
Length (l)		m	4	80
Width (w)		m	1	80
Window to Wall Ratio (WWR)	South		0	0.95
	North		0	0.95
	East		0	0.95
	West		0	0.95
Orientation (ORI)		°	−180	180
Wall U-value (U_{Wall})		W/m^2K	0.411	0.776
Window U-value (U_{Win})		W/m^2K	0.5	2
Floor U-value (U_{gfloor})		W/m^2K	0.411	0.864
Floor heat storage capacity (C_{floor})		J/kg K	900	1200
Roof U-value (U_{Roof})		W/m^2K	0.191	0.434
Window g-value (g_{Win})			0.5	0.95
Air Change Rate (ACH)		h^{-1}	0.2	1
Height of ground thermal zone from ground in storeys			0	10

4.2. Deep-learning architectures evaluated

The evaluated architectures have the following hidden layer configuration (see Fig. 7):

- Single LSTM layer (referred to as 'LSTM model' in this paper).
- Single LSTM layer + one simple hidden layer (referred to as 'Two-layer Deep LSTM model' in this paper).
- Single LSTM layer + two simple hidden layers (referred to as 'Three-layer Deep LSTM model' in this paper).

These architectures are benchmarked against a simple ANN that has one simple hidden layer and an output layer. The suitable hyper-parameters, i.e. number of hidden units and L2 regularisation parameter are chosen using a cross-validation strategy.

4.3. Development of the thermal zone component

The thermal zone component consists of a heating and a cooling ML model. The component is developed using two different methods: (1) developing the heating and cooling ML models individually, and (2) jointly developing heating and cooling ML models. In Section 4.3.1, an individual heating ML model is developed for different model architectures from the bottom up. However, the individual cooling ML model is developed (in Section 4.3.2) by means of transfer learning. In Section, 4.3.3, the heating and cooling ML models are jointly developed using MTL.

Deep-learning models are developed in Python using the Keras library with Tensorflow backend [44]. The Adam optimisation algorithm with a learning rate of 0.001 is used to train the model. Manual tuning determined this learning rate. Apart from the learning rate, the number of epochs (or number of training iterations) and batch sizes (number of samples propagated through the network) are also determined manually by observing the reduction in training and CV error. For the generated data set, an epoch number ranging from 500 to 600 and a batch size of 50 provides a low CV error. *Computer hardware has a high influence on training time. For this paper, the models are trained on a CPU with a clock speed of 2.7 GHz.*

4.3.1. Performance of different network architectures on heating demand data

Monthly heating demand data for the thermal zone component are modelled using different architectures to evaluate their performance compared to simple ANN. The modelling process consists of two parts: cross-validation (CV) and testing. CV is used to tune the hyper-parameters of different architectures while testing determines model generalisation. Generalisation refers to the validity of the model beyond the training data, assuming the same underlying distribution for training, validation and test data. CV and testing are performed by splitting the generated simulation data into training, CV, and test data. Performance of architecture on CV data is used to finalise the hyper-parameters while test data performance is used to evaluate the generalisation error. The hyper-parameters tuned are number of hidden units and L2 regularisation parameter. Hyper-parameters with low cross-validation error is selected to compare different architectures. The performance metric used to measure CV and generalisation error is the coefficient of determination (R^2). The performance criteria used to evaluate an architecture are:

(1) Generalisation error (i.e. test data R^2),
(2) Training time,
(3) Number of trainable parameters (indicates the complexity of a model to be trained).

4.3.2. Performance of different network architectures developed using transfer learning on cooling demand data

Transfer learning is carried out by freezing the first hidden layer of the heating ML model developed in Section 4.3.1 and training the remaining layers to suit the needs of the cooling ML model (see Fig. 4). For example, in the two-layer deep LSTM model, the non-linear representation learned by the LSTM layer for heating demand predictions is utilised to develop the cooling ML model. This representation does not change during the development of the cooling ML model. Only the weights of the simple hidden and output layers are optimised to meet the needs of the cooling ML model. The effectiveness of transfer learning for developing components is evaluated using the performance

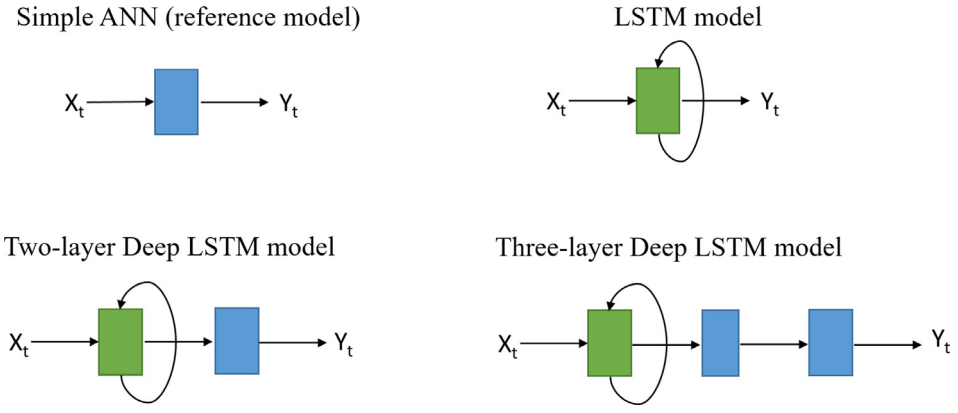

Fig. 7. Evaluated architectures.

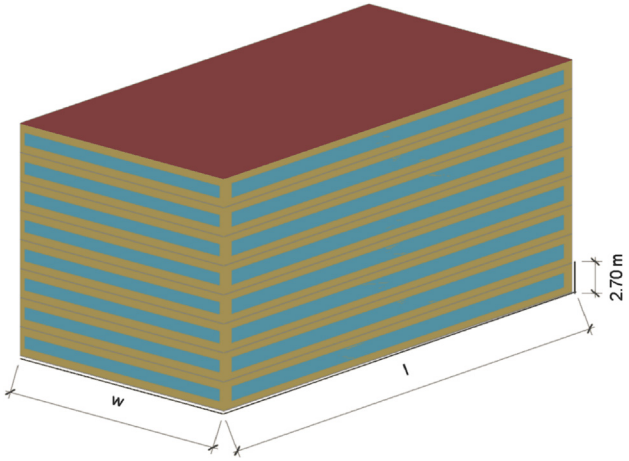

Fig. 8. Design test case layout.

criteria defined in Section 4.3.1.

4.3.3. Performance of multi-task representation on heating and cooling demand data

In this section, a deep-learning model developed using MTL with multiple outputs is evaluated. Since the Multi-Task model predicts both the heating and cooling demand of a thermal zone, it is referred to as the thermal zone ML model. The shared and component representations are defined based on the findings on the performance of heating and cooling ML model. Multi-Task representation is also evaluated with the performance criteria set in Section 4.3.1.

4.4. Suitability of deep learning and CBML of building design stage application

Models need to be accurate and fast to effectively explore designs. For the building layout shown in Fig. 8, accuracy and computation time are evaluated in 201 samples generated using the Latin hypercube sampling method. Accuracy is measured by the model's ability to predict the thermal zone's monthly demands similar to EnergyPlus simulation results. Computation time is the time needed to obtain results for 201 design combinations from EnergyPlus compared with the ML model.

5. Results

5.1. Performance of the deep-learning architectures for modelling heating ML model

Fig. 9 shows the performance of the different architectures on test data. The test data R^2 for deep architectures is ~0.993, while simple ANN has an R^2 of 0.965. Table 2 shows the number of hidden units in each layer, training time, and number of trainable parameters. The training time for simple ANN is 4.16 min, and deep-learning architectures are 11.12, 11.15 and 11.42 min respectively. The increase in training time (compared to simple ANN) is also reflected by the number of parameters to be optimised during the training process. Simple ANN has 1801 parameters to be optimised. Optimisable (or trainable) parameters increase to 17,261 parameters for the three-layer deep LSTM model.

Furthermore, the three-layer deep LSTM model has fewer parameters to be trained compared to the LSTM model. The reason for this difference is that the LSTM model has 70 memory cell blocks in its LSTM layer, while the three-layer deep LSTM model has 50 memory cell blocks. By adding simple hidden layers to the deep LSTM model, the complexity of the model is lowered while still possessing high test-data performance.

5.2. Effectiveness of transfer learning for modelling the cooling ML model

Fig. 10 shows the performance of the different architectures developed using transfer learning on test data. As shown in Fig. 10, all of the evaluated architectures have similar R^2 as the heating ML model, which ranges from 0.96 to 0.99. Table 3 shows the training times of different architectures developed using transfer learning. Training a simple ANN without transfer learning takes 4.6 min (see Table 2), while transfer learning reduced training time to 1.3 min (see Table 3). The training time for deep-learning models using transfer learning is 2.9 or 3.9 min respectively. This training method is about 65–73% faster than training from scratch. Transfer learning also increases the reusability of model layers developed from similar situations.

5.3. Effectiveness of multi-task representation for modelling both heating and cooling

Since both the two and three-layer deep LSTM models have a high R^2 and relatively low numbers of trainable parameters, these architectures are extended to a Multi-Task representation. The developed Multi-Task representations are shown in Table 4.

Table 5 shows the performance of the models developed through Multi-Task Learning (MTL). It can be noted that the R^2 of these model for predicting heating and cooling are similar to models trained separately (see two and three-layer deep LSTM model performance in Figs. 9 and 10). The training times are 8.79 and 8.89 min and the number of trainable parameters are 17,882 and 17,702 respectively.

Using MTL, training times for developing the components are lower than developing them separately. Developing heating and cooling models using three-layer deep LSTM model architecture took ~15.32 (11.42 + 3.9) minutes. Developing a similar configuration (i.e. configuration 2 in Table 4) took only 8.89 min, which is about 42% faster.

5.4. Suitability for building design space exploration

Since the thermal zone ML model (i.e. Multi-Task representation) performs equally well as heating and cooling ML model; it is used to evaluate suitability for building design space exploration. Fig. 11 shows the R^2 for 201 test design cases. Monthly cooling predictions have an R^2 of 0.983, and monthly heating predictions have an R^2 of 0.848. The component-based ML model took 0.9 s to predict monthly demands for the same 201 design combinations. EnergyPlus simulating 201 design combinations took 1145 s. Component-based ML model computation time includes the time to load input data from CSV file, scale input data, make a prediction, and rescale the output data.

6. Discussion

Deep-learning architectures perform relatively better than simple ANN on test data used to evaluate different architectures. Both LSTM and deep LSTM models have about 2.8% higher prediction accuracy compared to simple ANN on test data. However, the training time of deep-learning architectures is higher than simple ANN. The training times can be reduced by using methods such as transfer learning or GPU computing. Furthermore, within the evaluated deep-learning architectures, it can be noted that the deep LSTM models have fewer trainable parameters (two-layer: 16,841, and three-layer: 17,261 trainable parameters) than LSTM model (27,791 trainable parameters). A similar trend is observed for Multi-Task representation models indicating that having more layers could result in a compact model representation.

Input parameters or features for both the heating and cooling ML models are the same. Hence, features learned by initial layers of the heating ML model can be reused during the development of the cooling ML model. The test R^2 of cooling ML for different model architectures is

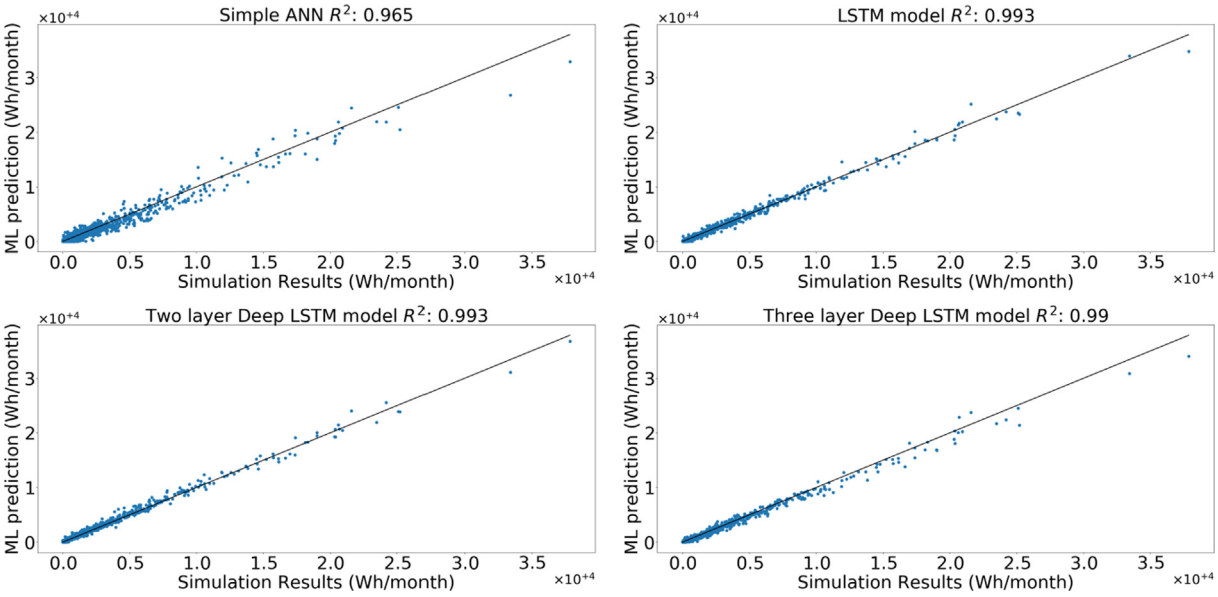

Fig. 9. Test data fit.

Table 2
Number of hidden units, training time and number of trainable parameters of different architectures.

Name of the architecture	Number of inputs	Number of hidden units	Number of output	Training time (min)	Number of trainable parameters
Simple ANN	28	[60]	1	4.16	1801
LSTM model		[70]		11.12	27,791
Two-layer deep LSTM model		[50, 20]		11.15	16,841
Three-layer deep LSTM model		[50, 20, 20]		11.42	17,261

Table 3
Performance of different deep architectures developed using transfer learning to model zone cooling demand.

Name of architecture	Training time (min)	Number of trainable parameters	Total number of parameters
Simple ANN	1.3	61	1801
LSTM model	2.9	71	27,791
Two layer Deep LSTM model	2.9	1041	16,841
Three layer Deep LSTM model	3.9	1461	17,261

similar to the heating ML model. This shows that features learned by simple ANN or deep architectures can be transferred to similar models and obtain similar prediction accuracies. The use of transfer learning for the development of components/ML models can drastically reduce the training time of the models and increases the reusability of model layers.

Having individual components offers the flexibility of using or adapting them as needed. This flexibility may not always be required, in which cases Multi-Task representation can be utilised to model the system. The effectiveness of such an approach for more than two response parameters needs to be further evaluated. Furthermore, it can be

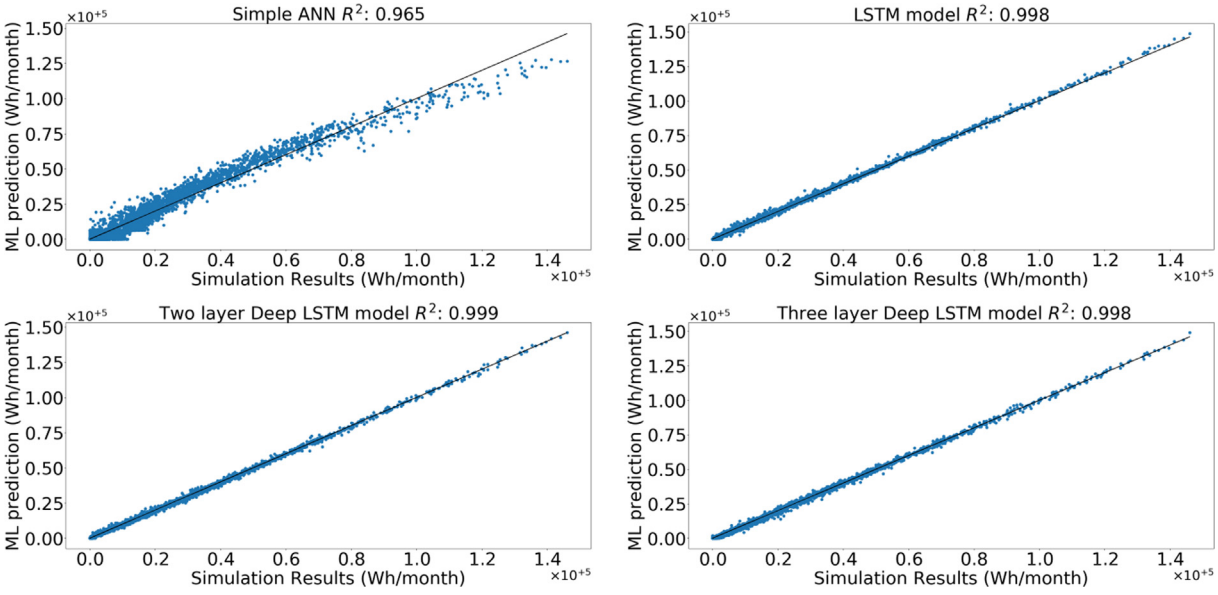

Fig. 10. Test data fit - Model fit through Transfer learning.

Table 4
Architecture of Multi-Task representation.

	Number of inputs	Shared layers	Component representation	Number of outputs
Configuration 1	28	**LSTM layer:** 50 memory units	Simple NN layer: 20 hidden units	2
Configuration 2	28	**LSTM layer:** 50 memory units **Simple NN layer:** 20 hidden units	Simple NN layer: 20 hidden units	2

Table 5
Performance of Multi-Task representation to model zone heating and cooling demand.

	Test R^2	Training time (min)	Number of trainable parameters
Configuration 1	**Heating:** 0.993 **Cooling:** 0.998	8.79	17,882
Configuration 2	**Heating:** 0.994 **Cooling:** 0.999	8.89	17,702

noted that the time required to model training using MTL is lesser than for developing individual components with deep architectures.

Nevertheless, deep-learning architectures performed well on test data. The performance of the deep-learning model dropped considerably in heating predictions on the evaluated test design cases. However, the cooling predictions on test cases generalised well. Their performance in cooling predictions indicates that through the deep learning and component approach, the ML model can be applied in new design cases. Exploration of training data indicated that the design space is cooling-dominated and sufficient high heating-demand training examples are not present. This is also reflected in Fig. 11(left) where low heating-energy areas have lower error rates compared to high heating-energy areas. This can be addressed by sampling the design space more effectively. The next step for research will be to investigate sampling methods on effective building design space that will result in representative data.

Regardless of the performance of the models for heating predictions on test design cases, the computation time of the deep-learning model is very low compared to BPS. Hence, applying deep-learning models for design space exploration will speed up the calculation time by 1272 times compared to BPS. This speed-up results in the ability to explore more design possibilities at early design stages. Component-based ML can be used to narrow design cases while BPS can be used to finalise the design. The complementary nature of Component-based ML and BPS can make the design process more efficient.

7. Conclusion

The ability to provide rapid design feedback together with information to back up design recommendations is critical for sustainable building design. A component-based ML approach provides rapid feedback together with information to support it. ML model inputs are directly linked to the design parameters and capture only the relationship between input parameters and response variables. This nature of ML allows energy modelling experts to produce training data with no prediction gap. Having ML models on such data reduces the uncertainties that result from a modeller's technical bias and skills.

However, the component-based approach increases model development time and the need for models with high accuracy. This paper evaluated deep-learning architectures and development methods in the context of component-based modelling to predict building heating and cooling energy demand. A component-based modelling approach is an engineering type of deep learning. Modelling components using deep learning architectures, transfer learning and MTL approaches turned out to be beneficial. Deep-learning architectures have higher prediction accuracies compared to simple ANN. Transfer learning and Multi-Task learning lower development time and increase the reusability of model layers. Potential benefits of deep learning for building energy predictions based on this study are:

- **Higher accuracy compared to Simple ANN.** This results in the applicability of the ML model in more design situations.
- **High computation speed compared to BPS.** This means more design options can be evaluated using optimisation techniques or other sensitivity analysis methods.
- **Reusability of model layers in a new situation.** This results in more efficient ML model development and sharing of the model layer between design teams or organisations.
- **A compact mathematical model through Multi-Task representation.** As a result, more efficient calculation speeds up

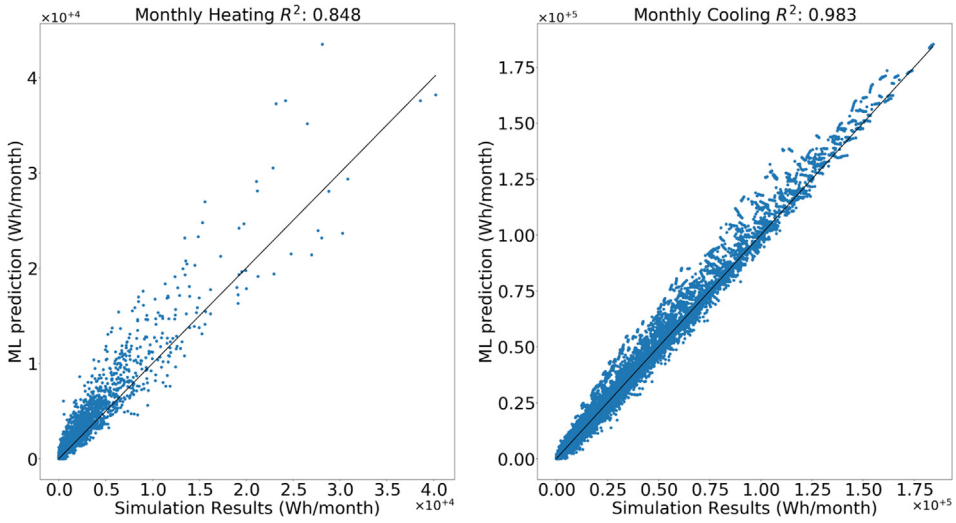

Fig. 11. Test design cases fit. Monthly heating (left), cooling (right).

prediction of the Component-based ML model.

More design case testing is required to evaluate the potential of these methods for their application in design-stage energy predictions. Furthermore, better design space sampling methods need to be evaluated to generate training data that results in general models suitable for new design applications. Given the potential of deep-learning models, more research is required to further evaluate their potential for this domain.

Acknowledgements

The research is funded by STG-14-00346 at KUL and by Deutsche Forschungsgemeinschaft (DFG) in the Researcher Unit 2363 “Evaluation of building design variants in early phases using adaptive levels of development” in Subproject 4 “System-based Simulation of Energy Flows.” The authors acknowledge the support of ERC AdG A-DATADRIVE-B (290923), KUL: GOA/10/09 MaNet, CoE PFV/10/002 (OPTEC), BIL12/11T; FWO: G.0377.12, G.088114N, G0A4917N; IUAPP7/19 DYSCO.

References

[1] G. Zapata-Lancaster, C. Tweed, Tools for low-energy building design: an exploratory study of the design process in action, Archit. Eng. Des. Manage. 12 (4) (2016) 279–295.
[2] C. Bleil de Souza, Contrasting paradigms of design thinking: the building thermal simulation tool user vs. the building designer, Autom. Constr. 22 (Mar. 2012) 112–122.
[3] S. Singaravel, P. Geyer, Simplifying building energy performance models to support an integrated design workflow, in: International Workshop of the European Group for Intelligent Computing in Engineering edition: 23, 2016.
[4] G. Augenbroe, Trends in building simulation, Build. Environ. 37 (8–9) (2002) 891–902.
[5] C. van Dronkelaar, M. Dowson, C. Spataru, D. Mumovic, D. Mumovic, A review of the regulatory energy performance gap and its underlying causes in non-domestic buildings, Front. Mech. Eng. 1 (Jan. 2016) 17.
[6] P. De Wilde, The gap between predicted and measured energy performance of buildings: a framework for investigation, Autom. Constr. 41 (2014) 40–49.
[7] A.C. Menezes, A. Cripps, D. Bouchlaghem, R. Buswell, Predicted vs. actual energy performance of non-domestic buildings: using post-occupancy evaluation data to reduce the performance gap, Appl. Energy 97 (2012) 355–364.
[8] I. Goodfellow, Y. Bengio, A. Courville, Deep Learn. (2016).
[9] H. Horsey, K. Fleming, B. Ball, N. Long, Achieving actionable results from available inputs: metamodels take building energy simulations one step further, in: ACEEE Summer Study on Energy Efficiency in Buildings, 2016.
[10] L. Van Gelder, P. Das, H. Janssen, S. Roels, Comparative study of metamodelling techniques in building energy simulation: Guidelines for practitioners, Simul. Model. Pract. Theory 49 (2014) 245–257.
[11] L. Magnier, F. Haghighat, Multiobjective optimization of building design using TRNSYS simulations, genetic algorithm, and Artificial Neural Network, Build. Environ. 45 (3) (Mar. 2010) 739–746.
[12] B. Eisenhower, Z. O’Neill, S. Narayanan, V.A. Fonoberov, I. Mezić, A methodology for meta-model based optimization in building energy models, Energy Build. 47 (2012) 292–301.
[13] A. Chari, S. Christodoulou, Building energy performance prediction using neural networks, Energy Efficiency (2017) 1–13.
[14] Z. Hou, Z. Lian, Y. Yao, X. Yuan, Cooling-load prediction by the combination of rough set theory and an artificial neural-network based on data-fusion technique, Appl. Energy 83 (9) (2006) 1033–1046.
[15] S.M.C. Magalhães, V.M.S. Leal, I.M. Horta, Modelling the relationship between heating energy use and indoor temperatures in residential buildings through Artificial Neural Networks considering occupant behavior, Energy Build. 151 (2017) 332–343.
[16] F. Ascione, N. Bianco, C. De Stasio, G.M. Mauro, G.P. Vanoli, CASA, cost-optimal analysis by multi-objective optimisation and artificial neural networks: A new framework for the robust assessment of cost-optimal energy retrofit, feasible for any building, Energy Build. 146 (2017) 200–219.
[17] S. Paudel, et al., A relevant data selection method for energy consumption prediction of low energy building based on support vector machine, Energy Build. 138 (2017) 240–256.
[18] F. Zhang, C. Deb, S.E. Lee, J. Yang, K.W. Shah, Time series forecasting for building energy consumption using weighted Support Vector Regression with differential evolution optimization technique, Energy Build. 126 (2016) 94–103.
[19] B. Dong, C. Cao, S.E. Lee, Applying support vector machines to predict building energy consumption in tropical region, Energy Build. 37 (5) (2005) 545–553.
[20] R.E. Edwards, J. New, L.E. Parker, Predicting future hourly residential electrical consumption: A machine learning case study, Energy Build. 49 (2012) 591–603.
[21] Z. Yu, F. Haghighat, B.C.M. Fung, H. Yoshino, A decision tree method for building energy demand modeling, Energy Build. 42 (10) (2010) 1637–1646.
[22] A. Tsanas, A. Xifara, Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools, Energy Build. 49 (2012) 560–567.
[23] M.-Y. Cheng, M.-T. Cao, Accurately predicting building energy performance using evolutionary multivariate adaptive regression splines, Appl. Soft Comput. 22 (2014) 178–188.
[24] K. Amasyali, N. Gohary, A review of data-driven building energy consumption prediction studies, Renew. Sustain. Energy Rev. 81 (2018) 1192–1205.
[25] C. Fan, F. Xiao, Y. Zhao, A short-term building cooling load prediction method using deep learning algorithms, Appl. Energy 195 (2017) 222–233.
[26] S. Singaravel, P. Geyer, J. Suykens, Deep neural network architectures for component-based machine learning model in building energy predictions, in: EG-ICE, 2017, pp. 260–268.
[27] N. Cohen, O. Sharir, A. Shashua, On the expressive power of deep learning: a tensor analysis, arXiv, vol. cs.NE, 2015.
[28] S. Singaravel, P. Geyer, J. Suykens, Component-based machine learning modelling approach for design stage building energy prediction: weather conditions and size, in: Proceedings of the 15th IBPSA Conference, 2017, pp. 2617–2626.
[29] P. Geyer, S. Singaravel, Component-based performance prediction for sustainable building design based on systems engineering and machine learning: the energy perspective (submitted for publication).
[30] Modelica, “Modelica.” Available: https://modelica.org/.
[31] U. of W.–M. S. E. Laboratory, TRNSYS, a transient simulation program, Madison, Wis.: The Laboratory, 1975, 1975.
[32] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[33] S. Ruder, An overview of multi-task learning in deep neural networks, arXiv Prepr, June 2017.
[34] R. Caruana, Multitask learning, Mach. Learn. 28 (1) (1997) 41–75.
[35] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Networks 61 (2015) 85–117.
[36] S. Hochreiter, J.J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1–32.
[37] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, Adv. Neural Inf. Process. Syst. (2014) 3104–3112.
[38] M. Sundermeyer, R. Schl, H. Ney, LSTM neural networks for language modeling, Proc. Interspeech (2012) 194–197.
[39] M. Sundermeyer, H. Ney, R. Schluter, From feedforward to recurrent LSTM neural networks for language modeling, IEEE Trans. Audio Speech Lang. Process. 23 (3) (2015) 517–529.
[40] H. Sak, A. Senior, F. Beaufays, Long short-term memory recurrent neural network architectures for large scale acoustic modeling, Interspeech 2014 (September) (2014) 338–342.
[41] A. Graves, A. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, ICASSP 3 (2013) 6645–6649.
[42] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, A.Y. Ng, Multimodal deep learning, in: Proc. 28th Int. Conf. Mach. Learn., 2011, pp. 689–696.
[43] R. Kiros, R. Salakhutdinov, R. Zemel, Multimodal neural language models, in: ICML, 2014, pp. 595–603.
[44] F. Chollet, Others, “Keras.” GitHub, 2015.