Deep convolutional learning for general early design stage prediction models

Article *in* Advanced Engineering Informatics · September 2019

DOI: 10.1016/j.aei.2019.100982

3 authors:

Sundaravelpandian Singaravel
Bricsys

Johan A.K. Suykens
www.esat.kuleuven.be/stadius

Philipp Geyer
Technische Universität Berlin

S. Singaravel, J. Suykens and P. Geyer, "Deep convolutional learning for general early design stage prediction models," Advanced Engineering Informatics (2019): 100982. https://doi.org/10.1016/j.aei.2019.100982

Deep Convolutional Learning for General Early Design Stage Prediction Models

Sundaravelpandian Singaravel[a], Johan Suykens[b], Philipp Geyer[a]
[a] Architectural Engineering Division, KU Leuven, Belgium
[b] ESAT-STADIUS, KU Leuven, Belgium

Keyword: Convolutional neural network, Energy predictions, Machine learning, Feature learning

Abstract

Designers rely on performance predictions to direct the design toward appropriate requirements. Machine learning (ML) models exhibit the potential for rapid and accurate predictions. Developing conventional ML models that can be generalized well in unseen design cases requires an effective feature engineering and selection. Identifying generalizable features calls for good domain knowledge by the ML model developer. Therefore, developing ML models for all design performance parameters with conventional ML will be a time-consuming and expensive process. Automation in terms of feature engineering and selection will accelerate the use of ML models in design.

Deep learning models extract features from data, which aid in model generalization. In this study, we (1) evaluate the deep learning model's capability to predict the heating and cooling demand on unseen design cases and (2) obtain an understanding of extracted features. Results indicate that deep learning model generalization is similar to or better than that of a simple neural network with appropriate features. The reason for the satisfactory generalization using the deep learning model is its ability to identify similar design options within the data distribution. The results also indicate that deep learning models can filter out irrelevant features, reducing the need for feature selection.

1 Introduction

Conventionally, simulations are used to guide the design toward the required building performance. A few building performance metrics are energy efficiency, daylighting, and thermal comfort. Designers rely on rule-of-thumb knowledge when simulation models cannot provide instant design performance feedback [1], [2]. However, rule-of-thumb knowledge could potentially lead the design toward a wrong direction. Hence, having models that can provide rapid and accurate results is necessary. Furthermore, 20% of design decisions taken at the early design stage affect 80% of the subsequent design decisions [3]. Therefore, it is important to take the right decisions at the early design stages. In this study, we utilize an energy analysis as an exemplary performance criterion. The inferences from the energy analysis can be relevant to other performance analyses as well.

Early design energy analysis simulation models in practice utilize simplified thermal representations along with technical specifications, e.g., based on American Society of Heating, Refrigerating and Air-Conditioning Engineers standards [4]. As the design progresses, more detailed information is added to the simulation model. Typical simulation tools used for energy analysis are EnergyPlus, TRNSYS, IES-VE, DesignBuilder, jEPlus, and Sefaira [3]–[5]. For two different building designs, Shiel et al. [4] showed that the variations of early energy demand prediction compared to actual energy consumption were −39% and −22%. Upon addition of actual design information, the variations were reduced to 5% and −2%, respectively. Furthermore, the effort required to develop simulation models varies depending on the complexity of the information and design [4]. Therefore, the challenge for an efficient early design energy analysis is to obtain a model that balances accuracy, development effort, and computation time for analyzing design alternatives.

Simplified models developed from complex simulation data have high potential to act as a surrogate model. Machine learning (ML) offers the possibility of developing surrogate models that provide rapid and accurate building performance predictions [6]–[8]. Quick ML predictions make ML models ideal for early design stage performance analysis because they allow for more design options to be evaluated at the early design stages. Moreover, a high computation speed reduces the designer’s reliance on rule-of-thumb knowledge and enables quantitatively well-justified decisions. However, ML models generalize within the data distribution, which is determined by the input parameter/features and training data. The challenge to overcome is the development of ML models that work robustly on unseen design cases. An unseen design case is defined as a design option, which is not present in the training data. It is critical to overcome this challenge because the evaluated design need not be captured within the training design cases. Therefore, identifying methods for overcoming this challenge will increase the utilization of ML models in design, enabling rapid, accurate, and reliable early design stage predictions.

Deep learning, a sub-domain of ML, has successfully been shown in many other domains such as image recognition to automatically extract good features resulting in model generalization [9]. The objectives of this study are to propose a deep learning architecture that generalizes well in unseen design cases and obtain an initial understanding of the features extracted by the deep learning model. The research questions addressed in this study are as follows: (1) Which deep learning architecture results in a satisfactory model generalization? (2) How important is feature engineering and selection for deep learning methods? (3) What are the underlying characteristics of the features learned by deep learning models? Future research will focus on the complexity of the data used for training. Nevertheless, the utilized data are obtained from simulation models representative of early design stages. More information on the utilized data is presented in Section 3.1.

The evaluation of deep learning architectures is performed by benchmarking two types of deep learning model architectures with a simple neural network (NN) architecture. The deep learning model architectures evaluated are multilayered NN and convolutional NN (CNN). To the authors' knowledge, the CNN has not been applied for design stage energy prediction, making this contribution significant. Upon benchmarking of deep learning models with a simple NN, hidden layer outputs are analyzed using kernel-principal component analysis (PCA) to understand the features learned by the deep learning model. Kernel-PCA analysis provides an interpretation of the characteristics of features extracted by a deep learning model. This paper is organized as follows: (1) the theory on utilized deep learning model architectures, (2) the methodology to evaluate deep learning, and (3) the results, discussion, and conclusion.

1.1 Background and motivation for deep learning

The generalization of an ML model in design refers to the validity of the model beyond training design cases, assuming the evaluated design case falls within the underlying data distribution. Artificial NNs[1] (ANNs) [10]–[19] and support vector machines (SVMs) [20]–[23] are the most popular ML algorithms

[1] In this paper, ANNs are also referred to as simple neural networks.

used to model building energy data. Generalizable ML models through ANNs and SVMs can be developed through appropriate feature engineering and selection.

Good features provide selectivity invariance, which means that the features are selective/relevant to the prediction problem but removes irrelevant features [9]. Feature selection is the process of selecting relevant input parameters for model development [24], [25]. Feature engineering is an approach that identifies input parameters, which account for the interaction between a building and its environment [26]. Examples of feature engineered inputs found in the literature are building shape factor, window to floor area ratio, and heat flow (*HF*) [27], [28]. The outcome of feature engineering and selection is that ML models can identify similar design options within the data distribution resulting in model generalization. However, the current research has typically focused on validating ML models with test cases that resemble training design cases. Hence, it is not clear how to increase the applicability of ML methods in unseen design cases.

Developing ML models through feature engineering and selection will be a time-consuming process as it requires domain knowledge in both ML and simulation methods. ML knowledge allows the model developer to identify suitable algorithms and training conditions, which results in a general model. On the other hand, knowledge in simulations allows the modeler to identify and select appropriate input features. Finding an engineer with such expertise is difficult. This challenge is amplified when ML models have to be developed for many design performance metrics as well. Hence, automation in feature engineering and selection will accelerate the use of ML methods for an early design stage performance analysis.

Within deep learning, the input features are transformed hierarchically using non-linear layers before making the final prediction. Training of the hierarchical non-linear layer enables automatic extraction of good features from data by promoting selectivity invariance [9]. Furthermore, the hierarchical structure of deep learning exploits the compositional hierarchies of signals/data [9]. Compositional hierarchies are the observation of a high-level feature, which is the result of low-level features. In the case of building design's energy demand, the high-level feature is the energy demand and some low-level features are *HF*s and heat gains. The data used in this study are obtained from simulation models, which generate energy demand based on hierarchical interactions. Therefore, analyzing the features learned by the deep learning model could provide an impression on the similarities between deep learning and simulation models. Finally, utilizing features extracted to make the final prediction allows deep learning models to generalize effectively. CNNs have been shown to be easier to train and generalize better compared to multilayer NNs [9]. In Section 2, the technical details of the utilized model architectures are provided.

The similarities between deep learning and simulation models in terms of hierarchical representation make deep learning an interesting ML method to explore further. However, the application of deep learning in the domain of building energy prediction is limited [29] because it requires a huge amount of data in the training process. Given the increasing computational power, it would be possible to generate such data with multiple design options. However, before generating a lot of data, it will be beneficial to obtain insights into the deep learning model for design.

Typical applications of deep learning for predicting building energy found in the literature are for load prediction/forecasting [30]–[33] and design stage predictions [8]. In certain cases, deep learning models have similar performances as conventional ML methods [30], [31]. In other cases, they outperform conventional ML methods [8], [32], [33]. The deep learning model architectures for predicting energy are stacked auto-encoders, recurrent NNs, and Boltzmann machines. CNNs have been used for building quality classification [34], fault detection [35], mitigation of fall [36], and people detection [37]. The data types used for current applications of CNNs are text and images. Utilization of CNNs with design information has not been reported, making the current research significant.

Limited works on deep learning models for building design energy performance analysis call for more research. Finally, an upcoming trend in ML is to understand the patterns learned by the black-box model [38], which helps the community to move toward interpretable artificial intelligence (AI). Analyzing the extracted features is a step toward interpretable AI. This study extends our understanding for model generalization in unseen design and model interpretability.

2 Theory on Deep Learning Neural Network Architecture

Deep learning models are evaluated based on their ability to predict heating and cooling demands on test design options. Each model, i.e., heating or cooling demand model, has two response variables, namely the peak and annual energy demand (see Figure 1 and Figure 2). Training of models with more than one response variable related to different tasks is called multitask learning (MTL). More information on MTL for energy models can be found in [8].

In this study, a simple NN, multilayer NN, and CNN are evaluated. Because the peak and annual energy demand of a design is directly predicted (i.e., not considered as a sequence) and training is performed end to end (i.e., in a single step), model architectures using recurrent and auto-encoder layers are not applicable. If the nature of data and the training process change, these architectures can be evaluated as well. This section introduces the utilized model architectures, the description of hidden layers, and the activation functions.

2.1 Model architectures

2.1.1 Simple and multilayered neural networks

The simple NN (or ANN) has been successfully applied in predicting building energy demand. Furthermore, current deep learning methods are extensions of simple NNs. Therefore, simple NNs are selected as a reference ML algorithm. Observations made on simple NNs should be applicable for other non-linear ML algorithms. Previous research indicated that through other conventional ML algorithms, a similar performance can be achieved provided appropriate model tuning is performed [39]. Multilayer NN is also evaluated as it is an easy extension of a simple NN to form a deep learning model.

Figure 1 shows the architecture of simple and multilayer NNs. A simple NN has one fully connected (FC) layer (see Section 2.2.1) with a rectified linear unit (ReLU) activation (see Section 2.2.4). A multilayer NN has more than one hidden layer. The number of hidden units in each hidden layer is manually determined by cross-validation (CV) during the training process. In this study, the multilayer NN has two, three, and four FC layers with a ReLU activation.

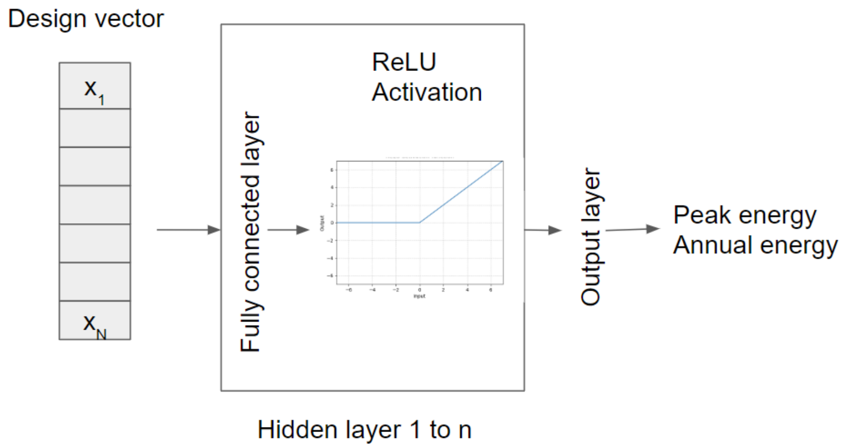

Figure 1 Illustration of simple and multilayer neural network architectures

2.1.2 Convolutional neural network

Figure 2 shows the architecture of the CNN with 1 to *n* convolutional layers, max pooling, and an FC layer. The number of convolutional operations and hidden units in each layer is manually determined through CV during the training process. The convolutional layer utilizes parametric ReLU (PReLU) activation instead of a ReLU activation. The use of PReLU activation provided better model performance than ReLU activation. The CNN with one, two, and three convolutional layers are evaluated in this study.

A CNN expects inputs in a matrix format. In this study, the input matrix is referred to as a design matrix (*DM*) as it contains all information pertaining to the design. The *DM* has a size of $M \times N$, where M is the number of parameter groups and N is the maximum number of features within all parameter groups. Section 2.2.2 describes the basic principle used to construct the *DM*. In Section 3.2.2, the method used to develop the *DM* is described.

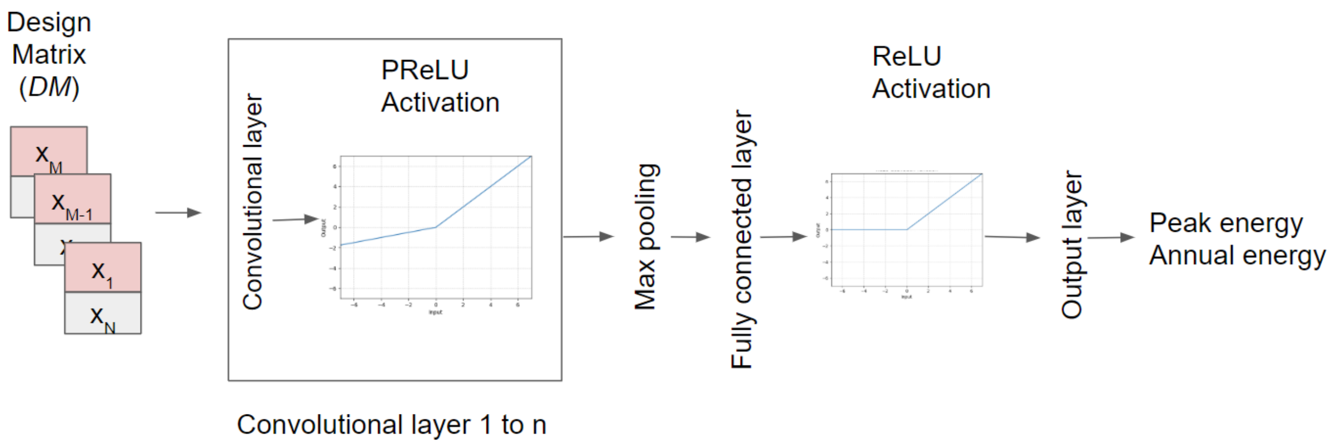

Figure 2 Illustration of convolutional neural network architecture

2.2 Description of hidden layers

2.2.1 Fully connected layer

An FC layer is the most commonly used hidden layer or output layer in any NN model. It comprises several hidden units that have to be tuned during the training process. Figure 3 shows the working of a hidden unit. The hidden unit obtains an input feature vector of length N. Each input feature in the vector is assigned a trainable weight. In Figure 3, features 1, 2, and 3 have a weight of –0.4548, 0.4118, and

0.6452, respectively. The weighted sum is the output of the hidden unit, which is referred to as the hidden feature.

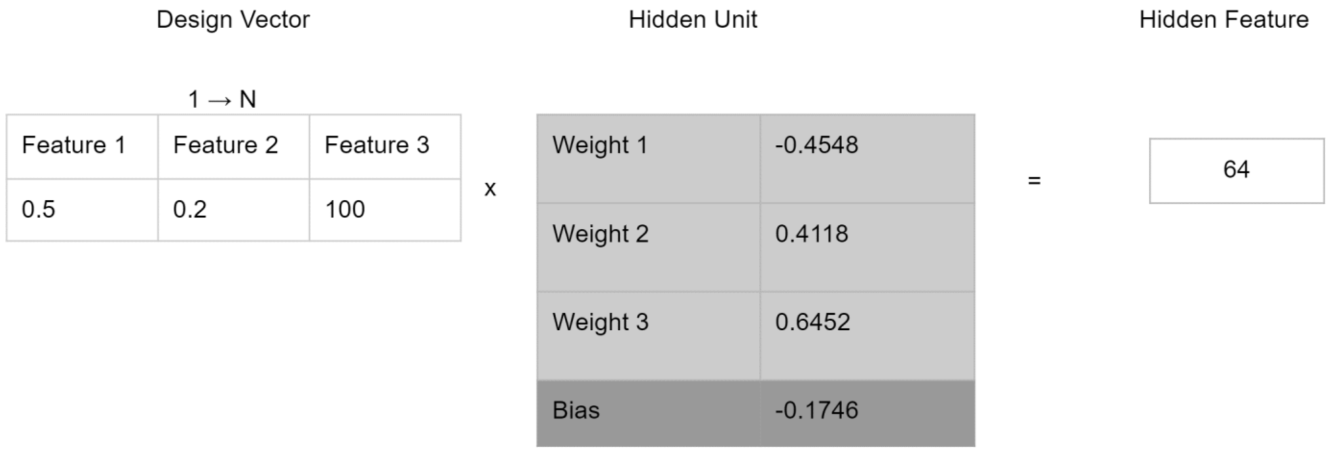

Figure 3 Illustration of a hidden unit

2.2.2 Convolutional layer

The use of a convolutional layer in NN models with images and time-series input data has provided state-of-the-art performance. However, a convolutional layer has not been used with design information. In this section, the working principle of a convolutional layer for design information is presented.

A convolutional layer obtains design inputs in the form of a *DM* instead of a vector; $DM \in$ *building design and performance related features/parameters*. The *DM* is generated by grouping similar features referred to as parameter groups, which range from 1 to M. An example of a parameter group with similar features is wall thermal conductivity and wall *HF*. Each parameter group consists of 1 to N similar features. In the above example, the parameter group consists of two similar features. The result of grouping is a *DM* with $M \times N$ dimension.

The convolutional layer consists of convolutional operations. The number of convolutional operations in a convolutional layer is determined during the training process. A convolutional operation is characterized by an $M \times K$ matrix, where K is the length of trainable weight vector per parameter group (K is also referred to as filter size). K is less than or equal to the number of features N in a parameter group. The output of a convolutional layer is referred to as a “feature map” (note that the *DM* is the input to the first convolutional layer only. Subsequent convolutional layers will receive feature maps as inputs.).

Figure 4 shows how a convolutional operation is performed for a 2×2 *DM*, i.e., a design with a two-parameter group and two features per group. The convolutional operation has a filter size (K) of 1, resulting in a convolutional operation with a matrix size of 2×1. In this example, parameter group 1 has a weight of −0.4548 and parameter group 2 has a weight of 0.4118. Features in column 1 and 2 are convoluted (through Equation 1) to obtain a feature map consisting of two features: −0.0597 and 3.7621. The first feature, −0.0597, is the weighted sum of values in feature column 1 together with the parameter group weight (PGW), followed by the addition of a bias term (i.e., $(0.2 \times -0.4548 + 0.5 \times 0.4118) - 0.1746$). Similarly, the second feature, 3.7621, is the weighted sum of values in feature column 2 (i.e., $(100 \times -0.4548 + 120 \times 0.4118) - 0.1746$).

$$\sum_{i=1}^{N} Feature\ i\ \times\ PGW + Bias\ (1)$$

Figure 4 highlights the following characteristics [40] of a convolutional layer, which results in the extraction of generalizable features [9]:

1. *Parameter (or weight) sharing*: Features within a parameter group have shared trainable weights. Parameter sharing also reduces the trainable weights compared to an FC layer with no shared weights.
2. *Sparse interaction*: Interactions captured by the convolutional operation are limited by shared parameters defined by the filter size. Figure 4 shows that the interactions observed by the model are limited to feature column 1 and 2 and not the entire matrix.
3. *Equivalent representation*: Parameter sharing results in a PGW that is equivalent to the entire parameter group, rather than each feature defined with a weight.

In this study, only the number of convolutional operations is tuned during the training process. Other hyperparameters such as the filter size are fixed. Evaluating the effect of other hyperparameters on model generalization is out-of-scope of the current study, as this study only evaluates the feature extraction capability of deep learning models for generalization. Future research will be performed to analyze the effects of other hyperparameters on model generalization.

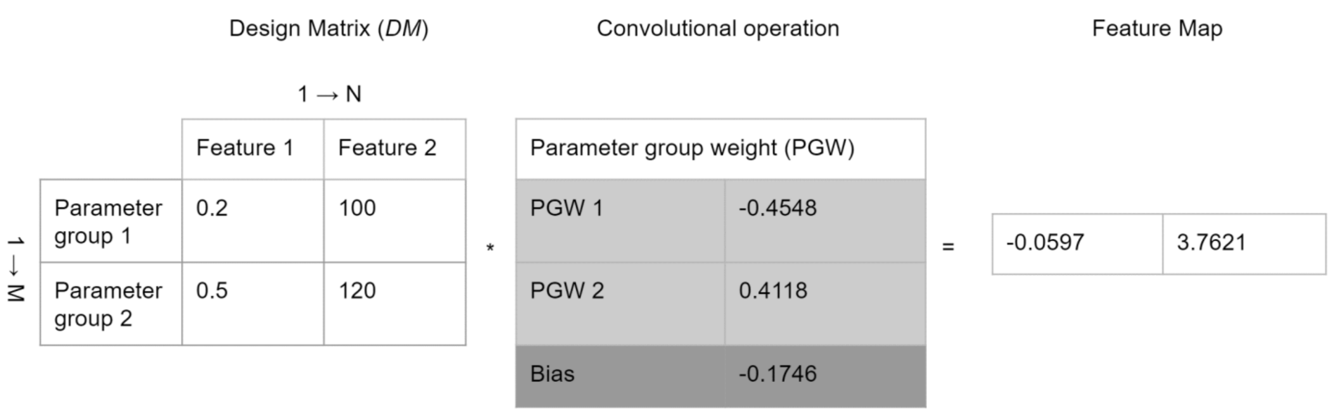

Figure 4 Illustration of a convolutional operation

2.2.3 Max pooling layer

Pooling layers are typically present in a CNN. This study utilizes a max pooling layer. The effectiveness of such layer compared with other types of pooling layers need to be evaluated in future research. A max pooling layer (see Figure 5) reduces the feature map by retaining only dominant (or high value) features. This layer promotes invariance (or insensitivity) through bottlenecks, as the dimension of the feature vector after max pooling is less before max pooling [41].

The hidden layer after max pooling learns to represent the prediction task with a smaller feature vector. If the models utilizing a max pooling layer generalize well, it indicates that the max pooling layer removes features that are not relevant for the particular task (in this case prediction of energy). Reducing the size of the feature vector by max pooling makes the deep learning model invariant to irrelevant features. However, understanding the induced invariance with respect to the building design input features is limited. Examples of such understanding are spatial invariance in images [42] and phase invariance for time-series data [43]. More research needs to be done to understand the type of invariance created by the pooling layer.

In this study, the CNN utilized has only one max pooling layer. The reason for this limitation is due to the small size of the feature maps generated by the utilized *DM*. The convolutional layer receives the *DM* of size $M \times 2$ and outputs a feature map of size $C \times 2$, where C is the number of convolutional operations in a layer and 2 is the number of similar features within a parameter group. The max pooling layer receives this feature map and outputs a reduced feature map to a size of $C \times 1$. Hence, adding more max pooling layers will not have any effect on the model. If the size of the feature map increases, the number of pooling layers could be increased. Identifying other *DM* configurations will be conducted in future research.

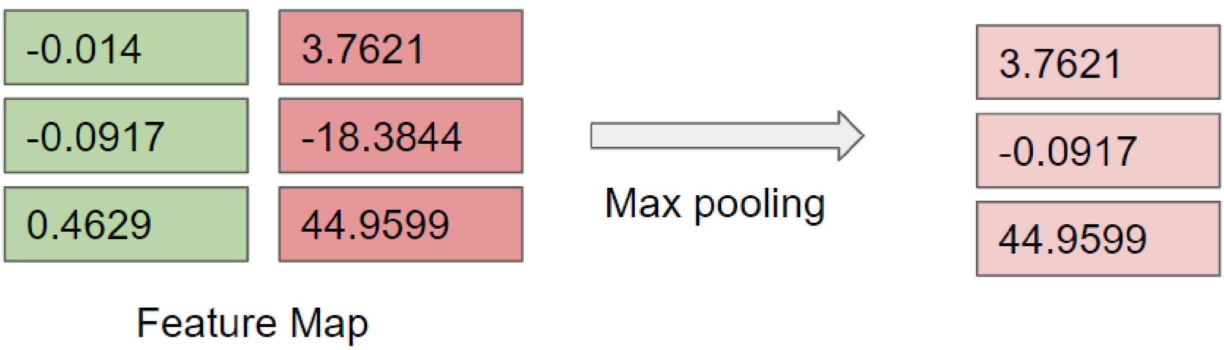

Figure 5 Illustration of max pooling

2.2.4 Description of activation functions

Suitable activation functions for an NN model varies for different data types. Some examples of activation functions are sigmoid, hyperbolic tangent, and ReLU. In this study, the ReLU activation is used together with an FC layer. The convolutional layer utilizes the PReLU activation as it offers a better performance than the ReLU activation. Equation 2 shows the ReLU activation, where negative values are made zero. Equation 3 shows the PReLU activation, where the negative values are multiplied by alpha (a), which is learned during the training process.

$$ReLU(X) = max(0, X) \qquad (2)$$

$$PReLU(X) = max(0, X) + a \times min(0, X) \quad (3)$$

3 Methodology for Evaluating Deep Learning for Design Stage Energy Predictions

The following methodology is applied to evaluate the feature extraction capability of deep learning methods for a satisfactory model generalization and to obtain an initial understanding of features learned by the deep learning model:

1. Benchmarking the performance of deep learning models against a simple NN on test design cases.
2. Kernel-PCA is utilized to analyze the characteristics of the features that results in model generalization.

3. Evaluating early design decisions using building performance simulation (BPS) and ML models.

This section starts by describing the generated data, which is followed by the methods for developing and evaluating deep learning models.

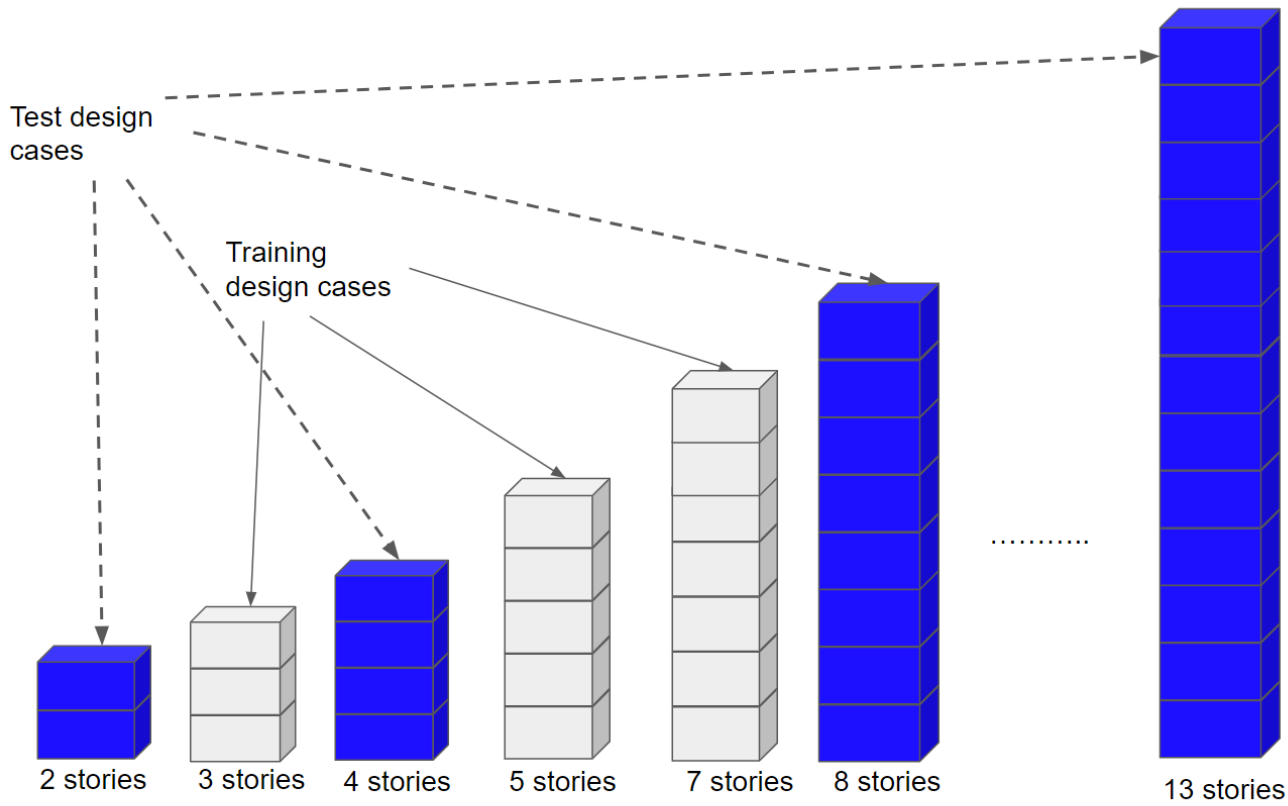

Figure 6 Training and test design cases

3.1 Description of training and test data

3.1.1 Design context

The early design stage decision support could be in the form of a what-if analysis [44], [45]. Some potential questions are “What if we increase the window area?”, “What if we reduce the efficiency of the HVAC system but increase the insulation level?”, and “What if we reduce the floor area per story and add an additional floor?”. To perform such analysis effectively, the utilized ML model provides predictions, which ensure that the decision taken on its predictions are valid as the design progresses. Therefore, test cases are created to analyze the reliability of design decisions taken from ML models on unseen designs. Furthermore, the training data provide the possibility of performing early what-if analyses and capture enough non-linearity to evaluate the robustness of the model on unseen test cases. Model generalization on more complex data will be performed in the future.

3.1.2 Parametric simulation model

The training data are design cases, which a model developer anticipate as potential design options evaluated by the designer. In contrast, the test data can be considered as design options evaluated by the designer. Training and test data are generated through parametric simulations in EnergyPlus version 8.7. The training data (gray blocks in Figure 6) come from design options of a 3-, 5-, and 7-story buildings. The test data (blue blocks in Figure 6) are obtained from the design options of 2-, 4-, 8-, 9-, 10-, 11-, 12-, and 13-story buildings, respectively. Building design options with 2 and 13 stories are later referred to as extreme test cases as they are in the boundaries of the test cases. From the generated data, the peak and annual energy demand data are extracted.

The models simulate an office building design located in Brussels. Assumptions in the models are (1) a fixed HVAC system, which is a variable air volume system with chillers and a gas boiler; (2) 100% occupancy and lighting and equipment gains between 9:00 and 17:00; (3) 50% occupancy at opening (8:00) and closing (18:00) hours; (4) 50% lighting usage after opening hours (8:00–18:00); and (3) room heating and cooling set points of 20/25 during opening hours and 16/28 after opening hours. Because the main objective of this study is to evaluate the deep learning model's ability to extract general features for better generalization, the assumptions in the models should not have an impact on the conclusions.

Table 1 presents the design parameters and sampling ranges utilized in the parametric simulation. The samples are generated using the Sobol sequence method, which is a quasi-random low-discrepancy sequence method. For the 3-, 5-, and 7-story buildings, 1500 design options are generated, resulting in a total training sample size of 4500. Similarly, for each test design case (see Figure 6), 1500 design options are generated. It can be noted from Figure 6 that only the 4-story building falls in the interpolation region of training design space. Other test design cases are outside the training design space.

Table 1 Design parameter ranges in the parametric simulation

	Units	Minimum	Maximum
Length (l)	m	20	80
Width (w)	m	20	80
Height (h)	m	3	6
Overhang length (l_{oh})[2]	m	0	6
Window to wall ratio (WWR)[2]		0.01	0.95
Orientation (α)	Degree	−180	180
Wall U-value (U_{wall})	$W/(m^2 \cdot K)$	0.41	0.78
Window U-value (U_{win})	$W/(m^2 \cdot K)$	0.5	2
Ground floor U-value (U_{floor})	$W/(m^2 \cdot K)$	0.41	0.86
Roof U-value (U_{roof})	$W/(m^2 \cdot K)$	0.19	0.43
Window g-value (g_{win})		0.1	0.9
Floor heat capacity (c_{floor})	$J/(kg \cdot K)$	900	1200
Infiltration air change rate (n_{air})	h^{-1}	0.2	1
Number of floors (n_{floor})		3, 5, 7	
Lighting heat gain (Q'_{light})	W/m^2	5	11
Equipment heat gain (Q'_{equip})	W/m^2	10	15
Chiller coefficient of performance (COP)		3	6
Boiler efficiency (η_{Boiler})		0.7	1
Chiller type		Electric reciprocating chiller	Electric screw chiller
Boiler pump type		Constant flow	Variable flow

3.2 Training and testing of deep learning architectures

Different ML model architectures with different input parameter configurations are trained and tested to identify conditions for conventional ML and deep learning model generalization. This section describes (1) the different input parameter configurations utilized in model development, (2) input parameter

[2] Varies differently in all orientations

configurations assigned to each ML model architecture, and (3) ML model selection and evaluation process.

3.2.1 Model input parameter configurations

Table 2 indicates three configurations of model input parameters utilized in the evaluation process. These three input parameter configurations are designed to show the importance of feature engineering and selection for conventional ML model generalization and to understand conditions under which deep learning extracts generalizable features from data.

Table 2 Model input parameter configurations

Configuration number	Description of group	Reference in text as
1	Design inputs are listed in Table 1.	Actual inputs (Act ip)
2	Certain design inputs from Table 1 are transformed using formulas given in Table 3. Non-transformed parameters are utilized as in category 1.	Feature engineered inputs (FE ip)
3	All design inputs together with feature engineered inputs.	Act + FE ip

Table 3 summarizes the formulas used to transform design parameters (i.e., feature engineering). In the feature engineering process, features/input parameters, which interact with other design parameter or other environmental factors, are identified. The building area is a feature that captures the interaction between building length (*l*) and width (*w*). On the other hand, transformations such as *HF*s capture the interaction between the building and its environment. For example, *HF*s through the wall capture the interaction between wall area, insulation level, and outdoor weather conditions (T_o) of the building's environment and indoor temperature (T_i). Weather conditions utilized to perform these transformations are average summer and winter conditions for the cooling and heating models. The indoor temperature is assumed to be 25 °C for the cooling model and 20 °C for the heating model.

Table 3 Formulas for feature engineering

Design parameters (Actual inputs)	Transformed inputs (Feature engineered inputs)	Units
Length (*l*)	Building area (*BA*)	m^2
Width (*w*)	$l \times w \times n_{floors}$	
Height (*h*)	Building volume (*BV*)	m^3
Number of floors (n_{floors})	$l \times w \times h \times n_{floors}$	
U-value	Heat flow (*HF*)	W
of wall, window, floor, roof	U-value × Area × (T_o – T_i)	
Window g-value	Solar gain (*SG*)	W
	Area × g_{win} × average solar radiation	
Infiltration air change rate	Infiltration gain (*IG*)	W
	Air specific heat capacity × density × air volume × (T_o – T_i)	

3.2.2 Model architectures and corresponding input configuration

The global model architecture is presented in this section, and hyperparameters in each layer are tuned during the training process. Table 4 indicates the trained model architectures and their input configuration. A simple NN is the reference ML model architecture for the deep learning model architectures. Therefore, the simple NN is trained with all input configurations. Benchmarking of deep learning models against simple NNs with actual inputs is performed to examine if the deep learning model can extract good features. Additionally, benchmarking against a simple NN with feature engineered inputs is performed to determine the quality of the extracted features.

The multilayer NN is evaluated to understand the feature extraction capability of the deep learning model. Hence, input configuration 1, i.e., actual input, is provided. The CNN is evaluated to understand its ability to extract good features from similar input parameters. Hence, input configuration 3 (Act + FE inputs) is provided.

Table 4 Model architecture and input configuration

Model architecture	Number of hidden layers	Model input configuration	Reference in text
Simple NN	1 FC layer	Act ip	Simple NN – Act ip
		FE ip	Simple NN – FE ip
		Act + FE ip	Simple NN – Act + FE ip
Multilayer NN	2 FC layers	Act ip	Multilayer NN – 2 layers
	3 FC layers		Multilayer NN – 3 layers
	4 FC layers		Multilayer NN – 4 layers
CNN	1 Convolution layer 1 FC layer	Act + FE ip	CNN – 1+1 layers
	2 Convolution layers 1 FC layer		CNN – 2+1 layers
	3 Convolution layers 1 FC layer		CNN – 3+1 layers

Simple and multilayer NNs require the inputs to be in a vector form. However, a CNN requires a matrix input. Table 5 presents the *DM* structure used for the CNN. Each design parameter (also referred to as actual inputs), wherever possible, is paired with its equivalent transformation or a design parameter. The objective of the grouping is to bring similar parameters together, which allows the convolutional layer to learn an equivalent parameter weight (see Figure 4). Equivalent transformations capture the effect of changes in one over another parameter. Examples of equivalent transformation are building length (l) to building area (*BA*) and U-values to *HF*. Similar design parameters are parameters that have similar effects on the energy consumption. Examples are lighting gain (Q'_{light}) and equipment gain (Q'_{equip}). Within the current feature space, if a parameter does not have an equivalent transformation or a similar design parameter, it is not paired with any other parameter (i.e., Feature 2 is zero). Orientation (α) is an example of a parameter that is not paired with any other parameter. Other potential arrangements of the data structure need to be researched further.

Table 5 Input data structure (i.e., DM) of a design option for CNN

Parameter group	Feature 1	Feature 2
1	Length (l)	Building area (*BA*)
2	Width (w)	Building area (*BA*)
3	Height (h)	Building volume (*BV*)
4	Number of floors (n_{floors})	*0*
5	Orientation (α)	*0*
6	Overhang length (l_{oh})	Window to wall ratio (*WWR*)
7	Window g-value	Solar gain

8	U-value	Heat flow (*HF*)
9	Floor heat capacity	*0*
10	Infiltration air change rate (n_{air})	Infiltration gain
11	Lighting heat gain (Q'_{light})	Equipment heat gain (Q'_{equip})
12	Chiller COP / Boiler efficiency	Chiller type / Boiler pump type

3.2.3 Computational environment

The simple NN and deep learning model are developed using the PyTorch library in Python [46]. Models are trained on NVIDIA Quadro M1000M, which has 512 CUDA cores and 2 GB memory. The training time[3] in Intel Core i7 processors takes approximately 5.3 min. In contrast, the training time in a graphical processing unit (GPU) is approximately 2 min. Training the deep learning model in this GPU is ~3 times faster than in a central processing unit.

3.2.4 Model selection and evaluation

All model architectures are trained using the ADAM optimization algorithm. The learning rate to update the model weights is $1e^{-4}$. Model overfitting is addressed through an L2 regularization penalty of 0.01. The optimization algorithm needs 10000 epochs for obtaining satisfactory convergence.

During the training process, the model performance is evaluated through the coefficient of determination (R^2) and mean absolute percentage error (MAPE) on the CV data. The CV data are a subset of training data, which has not been used in the training process. In this study, 20% of the training data are randomly selected to form the CV data. Model hyperparameters such as the number of hidden units are tuned until the CV error is low. The hyperparameter combination that resulted in a low CV error is used to train the final model.

The model generalization is evaluated based on the prediction accuracy in test design cases (see Figure 6). A model architecture is considered to have generalized when the R^2 is higher than 0.9 and MAPE is lower than 15%. Models meeting the abovementioned evaluation criteria are considered to have a satisfactory performance. Similarly, models that do not meet the above criteria are considered to have a poor performance.

3.3 Kernel-PCA for analyzing the effect of features

Using kernel-PCA, the effects of actual inputs, feature engineered inputs, and features extracted by deep learning models on model generalization are analyzed. To make the features extracted by deep learning model comparable with features received by a simple NN, the features from the $n-1$ hidden layer are analyzed. Kernel-PCA reduces the high-dimensional input/features to a two-dimensional input space. Dimensionality reduction makes input features with different dimensions comparable. For instance, models with actual inputs have 24 inputs, while models with feature engineered inputs only have 14 inputs.

The reduced two-dimensions from kernel-PCA are the 1st and 2nd principal components. The 1st principal component represents the highest variance in the input/feature space. The 2nd principal component is orthogonal to the 1st principal component and represents the second highest variance in

[3] Training time estimated for CNN – 2+1 layers

the feature space. The following methodology is utilized to analyze the effect of features on model generalization:

1. The kernel for kernel-PCA is selected based on its ability to reconstruct actual design inputs. To obtain comparable low-dimensional reductions, both feature engineered inputs and features extracted by deep learning models utilize the same kernel as actual design inputs. In this study, the radial basis function kernel is selected, as it has the lowest reconstruction error.
2. A training design case represented by different input configurations, i.e., actual inputs, feature engineered inputs, and features extracted by deep learning models, are reduced into two dimensions.
3. Test design cases represented by different input configurations are reduced to two dimensions using eigenvectors determined for training design case with different inputs.
4. Visualizing the principal components of training and test design cases along with information on floor area and energy provides us with insights on the characteristics of features for generalization.

3.4 Evaluating early design decisions using building performance simulation (BPS) and ML models

The objective of this section is to illustrate the evaluation of an early design case using the ML model and BPS. The evaluation is performed for an 8-story building design located in Brussels. The design process (reflection of what-if analysis) illustrated in this study has three stages. In each stage, the following are conducted:

Stage 1: Initial estimate of energy.

Stage 2: Decision on south and north window to wall ratio (*WWR*) is made.

Stage 3: Designers decide whether to change the window g-value or insulation level.

The methodology used to evaluate the ML models and BPS for the early design process takes the following criteria into consideration:

1. Estimate energy demands from the BPS and ML models with best test data performance. Comparing the energy demand estimates from the BPS and ML models shows the reliability of decisions taken from both approaches.
2. Estimate the time required to make a prediction from each model. The time required to estimate energy allows quantifying the suitability or effort required for steering early stage design through BPS and ML.
3. Visualize the principal components of the evaluated design to understand the reason for a prediction. The principal components are estimated using the same eigenvectors determined in Section 3.3.

4 Results

4.1 Performance of model architectures

In this section, the performance of heating and cooling models with different architectures on CV and test data is presented. The CV data are used to tune the number of hidden units/convolution operations in each layer, while the test data show the generalization of model architecture. Generalization refers to the validity of models beyond the training design cases, assuming that test design cases are within the data distribution.

4.1.1 CV data performance

Table 6 lists the heating model's hyperparameters obtained after manual tuning while Table 7 provides the corresponding CV errors. For peak heating predictions, the R^2 and MAPE range between 0.98 and 0.99 and 7.07% and 9.87%, respectively, indicating that all architectures have a satisfactory performance on the CV data. For total heating predictions, the R^2 and MAPE range between 0.94 and 0.97 and 15.65 and 26.48%, respectively. The deep learning architecture has a better CV data performance compared to the simple NN.

The data indicated that the simulated design cases are cooling dominated, which is the result of the utilized HVAC system configuration and internal gains. The cooling dominance, in turn, made a lot of similar designs to have significantly different energy demands caused by complex interactions within the building. Hence, the utilized features (in simple NNs) are not able to segregate similar design options effectively, resulting in the poor prediction quality from simple NNs on total heating predictions. The good performance of deep learning models indicates that the extracted features can segregate similar design options effectively.

Table 6 Heating model hyperparameters

Model architecture	Number of input parameters	Hidden unit per layer	Number of output parameters
Simple NN with actual inputs	24	40	2
Simple NN with feature engineered inputs	14	40	
Simple NN with actual and feature engineered inputs	30	40	
Multilayer NN - 2 layers	24	30, 25	
Multilayer NN - 3 layers	24	30, 30, 20	
Multilayer NN (4 layers)	24	30, 30, 25, 20	
CNN - 1+1 layers	30	30, 25	
CNN - 2+1 layers	30	30, 30, 20	
CNN - 3+1 layers	30	30, 30, 25, 20	

Table 7 Heating model hyperparameters and CV errors on heating demand predictions

Model architecture	Coefficient of determination (R^2)		Cross-validation MAPE (%)	
	Peak	Total	Peak	Total
Simple NN with actual inputs	0.98	0.95	9.87	23.83
Simple NN with feature engineered inputs	0.98	0.94	7.94	25.54
Simple NN with actual and feature engineered inputs	0.99	0.95	7.47	23.31
Multilayer NN - 2 layers	0.99	0.97	7.56	17.98
Multilayer NN - 3 layers	0.98	0.96	9.16	18.95
Multilayer NN - 4 layers	0.99	0.97	7.37	15.65
CNN - 1+1 layers	0.98	0.94	7.89	26.48

CNN - 2+1 layers	0.99	0.97	7.07	17.31
CNN - 3+1 layers	0.98	0.97	7.61	19.13

Table 8 presents the cooling model hyperparameters obtained after manual tuning while Table 9 indicates the CV errors. The R^2 and MAPE for peak cooling predictions range between 0.97 and 0.99 and 5.77 and 14.15%, respectively. For total cooling predictions, the R^2 and MAPE range between 0.97 and 0.99 and 5.78 and 13.21%, respectively, indicating that all architectures have a satisfactory performance on the CV data.

Table 8 Cooling model hyperparameters

Model architecture	Number of input parameters	Hidden unit per layer	Number of output parameters
Simple NN with actual inputs	24	25	
Simple NN with feature engineered inputs	14	25	
Simple NN with actual and feature engineered inputs	30	25	
Multilayer NN - 2 layers	24	30, 20	
Multilayer NN - 3 layers	24	30, 25, 20	2
Multilayer NN - 4 layers	24	30, 25, 20, 20	
CNN - 1+1 layers	30	30, 25	
CNN - 2+1 layers	30	30, 30, 20	
CNN - 3+1 layers	30	30, 25, 20, 20	

Table 9 Cooling model hyperparameters and CV errors on cooling demand predictions

Model architecture	Coefficient of determination (R^2)		Cross-validation MAPE (%)	
	Peak	Total	Peak	Total
Simple NN with actual inputs (Act ip)	0.97	0.98	14.15	13.21
Simple NN with feature engineered inputs (FE ip)	0.98	0.98	8.15	7.8
Simple NN with actual and feature engineered inputs (Act + FE ip)	0.97	0.97	12.59	13.21
Multilayer NN (2 layers)	0.98	0.97	11.42	12.52
Multilayer NN (3 layers)	0.98	0.99	8.75	8.21
Multilayer NN (4 layers)	0.99	0.99	5.77	5.78
CNN (1+1 layers)	0.98	0.99	8.68	7.52
CNN (2+1 layers)	0.99	0.98	7.41	7.50
CNN (3+1 layers)	0.99	0.99	6.74	6.22

4.1.2 Performance of ML models on test design cases

Figure 7 shows the performance of the heating models on the test design cases. As defined in Section 3.2.4, the performance of a model is satisfactory when R^2 and MAPE are higher than 0.9 and lower than 15%, respectively. Models that do not meet these performance criteria are considered to have poor performance. It can be noted from Figure 7 that the performance of the different architectures is not consistent in the different test cases.

The 4-story building falls within the interpolation zone of the training design cases. It can be noted from Figure 7 that in general, all model architectures perform well for the 4-story building. As the test cases move far away from the training design cases, the performance starts to reduce. The amount of performance reduction depends on the model architecture. The reason for performance reduction is due to the difference is thermal behavior captured in the training design cases when compared to test design cases. However, results show that utilization of appropriate ML model features and model architecture reduces the prediction error (i.e. increase in ML model performance). Finally, the 2-story building cases have a poorer performance than the 8-story building cases. However, both cases are close to the training design case. The reason for the poorer performance on the 2-story building is the absence of an intermediate floor, which influences both the top and bottom floor's thermal behavior independently.

For peak heating energy prediction, the performance of all model architectures is satisfactory for the 4- and 8-story buildings. In addition, the performance of specific model architectures is satisfactory in the other design cases. For the other design cases, the following architectures have satisfactory performances:

- Simple NN with FE inputs and Act + FE input,
- Multilayer NN with 4 hidden layers, and
- All CNN architectures.

It can also be noted that models with FE input parameters (both simple NNs and CNNs) consistently have better performances than models with only actual design inputs, indicating the significance of having features engineering with physical equations. Finally, the satisfactory performance of the selected deep learning architectures indicates that they can automatically extract generalizable features from data.

For total heating energy prediction, most of the models have an R^2 above 0.9. However, the overall error in predictions is higher, which is reflected in high MAPE values. The multilayer NN with 4 hidden layers and CNN with 2 convolutional layers have better performances compared to other architectures. The reason for the poorer performance of the other architectures is due to the complexity of data. The complexity is caused by similar design options having different total heating energy consumptions, which is the result of interactions within the building. The satisfactory performance of deep learning models indicates that the extracted features can segregate design options effectively.

Figure 8 shows the performance of cooling models on the test design cases. For peak cooling energy prediction, all model architectures have satisfactory performances on the 4-, 8-, and 9-story buildings. For other test design cases, the selected model architectures also performed well. The selected architectures are the simple NN with FE inputs and all CNN architectures. The satisfactory performance of the CNN on all design cases indicates that convolutional layers can extract good features from data. It should also be noted that the simple NN with actual and FE inputs has a poor performance in extreme test cases, highlighting the importance of feature selection. A similar trend is observed for the total cooling energy predictions.

In general, the CNN generalizes better than the simple NN with actual inputs. Depending on the architecture of the CNN, the reduction in MAPE varies. For peak heating demand predictions, the average reduction in MAPE ranged between 7.1% and 8%. Similarly, the average reduction in MAPE for the total heating predictions ranged between 1.4% and 9%. For cooling energy demand predictions, the reduction in MAPE for peak predictions ranged between 10.9% and 13.7%, and for the total demand predictions, the reduction in MAPE ranged between 10.8% and 15%. However, when comparing the CNN with the simple NN with feature engineered inputs, the overall reduction in MAPE ranged between 0% and 8%.

For the simple NN, manual feature engineering and selection play a crucial role in model generalization. Deep-learning model architectures can extract good features that extend the reusability of the model in complex datasets. Within the evaluated deep learning architectures, the proposed CNN architecture results in a better model generalization.

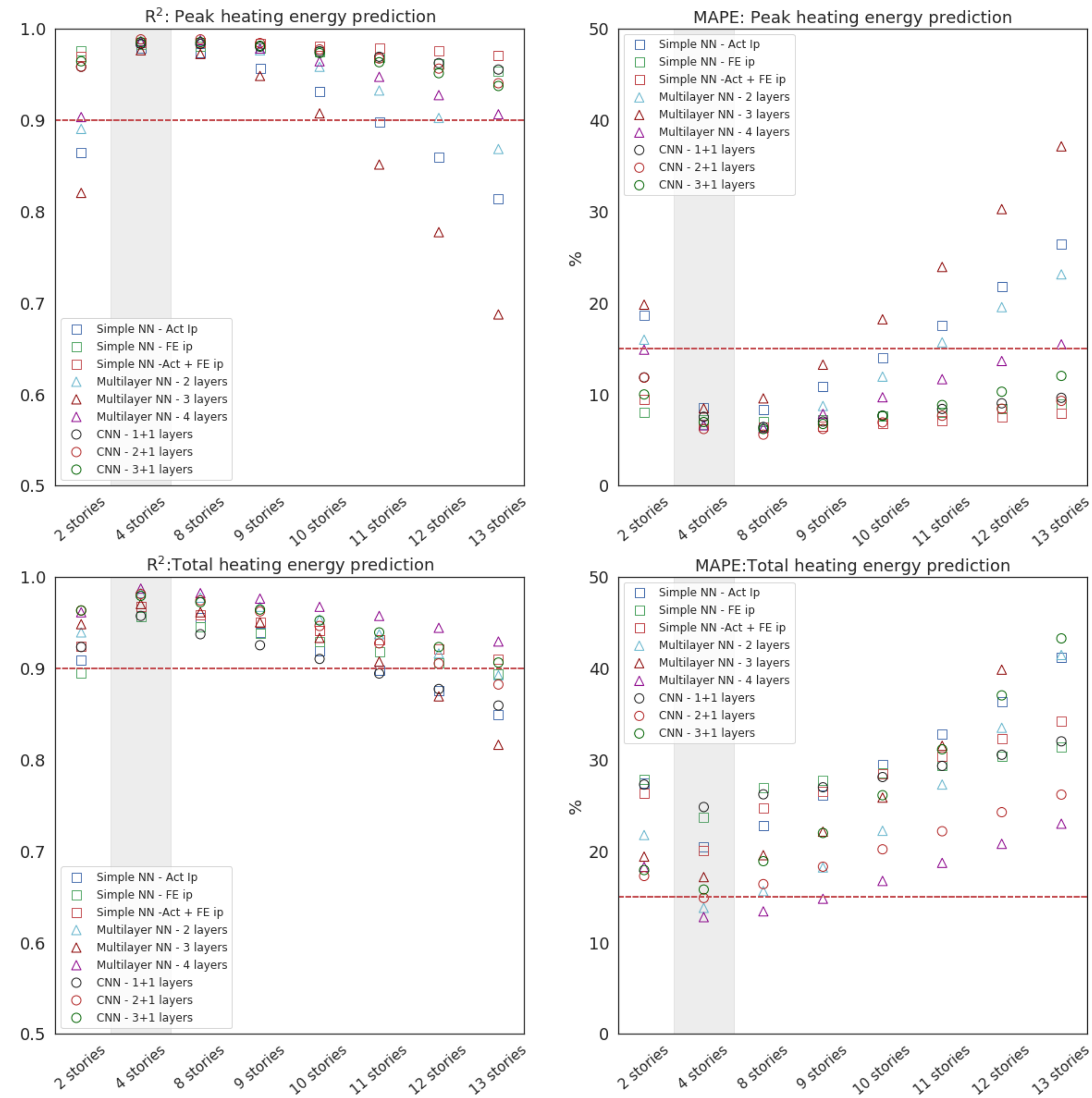

Figure 7 Performance of heating models on test design cases

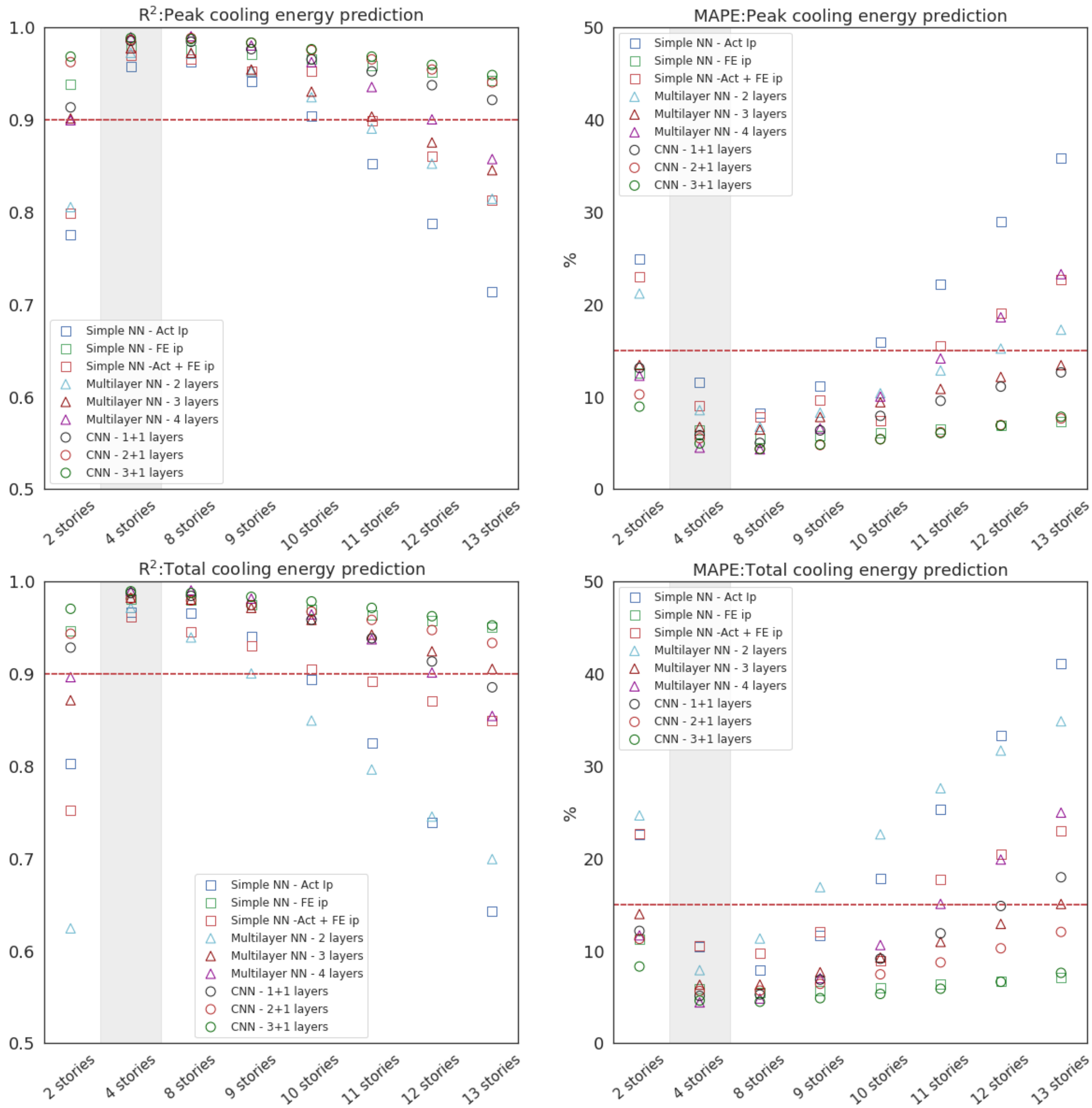

Figure 8 Performance of cooling model in test design cases

4.2 Effect of features on model generalization

In supervised learning, the models learn to identify the relationship between input and output variables. Input features determine the data distribution for a simple NN while for deep learning, the model determines the data distribution by hierarchically extracting features from input features. In this section, the effects of actual inputs, feature engineered inputs, and features extracted by the deep learning models on model generalization are analyzed. The data distributions generated by training and test design cases are referred to as training and test design spaces.

High input dimensional features are reduced to two dimensions using the kernel-PCA. The total heating demand and total floor area information are overlaid on the principal components from the kernel-PCA. The total heating demand is used to show the effect of features on model generalization, as simple NNs with all input configurations have a higher test data error compared to deep learning models. The total floor area captures information on increasing the number of floors. Only the 2- and 13-story buildings are presented in this section as the effects of features on the other test cases lie between these design cases.

4.2.1 Kernel-PCA on training design space

Figure 9 shows the 1st and 2nd principal components from the kernel-PCA of the training design space obtained through actual inputs, feature engineered inputs, and features extracted by the deep learning models. In Figure 9, information of the total heating demand is represented through purple to yellow gradient, and the total floor area is represented through black to white gradient. Models with actual and feature engineered inputs have equivalent features. Example of equivalent feature is the use of building area instead of building length and width as model input. Figure 9a shows six clusters: they represent buildings with 3, 5, and 7 stories with two types of boiler pumps. From Figure 9b, it can be noted that feature engineering has transformed six clusters into two clusters. The two clusters represent the type of boiler pump. For each cluster in Figure 9b, the building area and energy consumption increase as we move from the bottom to the top of the graph. The deep learning models have also learned to group similar designs together as the conventional feature engineering method. The multilayer NN features have buildings with area and energy gradients that move from right to left. Similarly, the CNN features have a gradient that moves from the right to the left.

Figure 7 shows that deep learning models generalize better in predicting total heating energy demand than simple NNs with feature engineering. The reason for the poorer performance of the simple NN is the poor segregation of the total heating energy clusters by feature engineered inputs (see Figure 9b) compared with feature learning by deep learning models (see Figure 9c and d). For other response variables such as cooling energy (not included in this study), feature engineered inputs resulted in satisfactory segregation of energy clusters, resulting in a satisfactory performance.

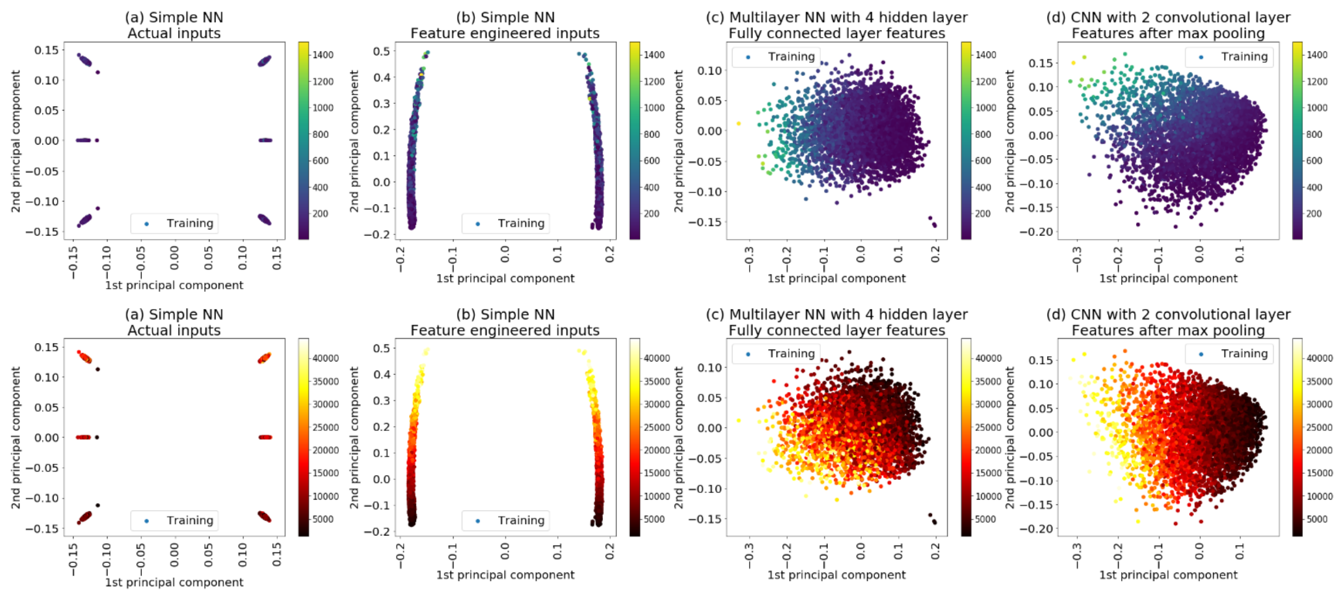

Figure 9 Principal component from kernel-PCA of training design space for actual inputs, multilayer NN feature, feature engineered inputs, and CNN features: (top) overlay with information of total heating demand (W); (bottom) overlay with information of total floor area (m^2)

4.2.2 Kernel-PCA of training and test design space

In this section, two test design cases are analyzed. The analyzed test design spaces are from the 2- and 13-story buildings, which are at the extremes of the test cases. Figure 10 shows the kernel-PCA of the 2-story building compared to the training design space whereas Figure 11 shows the kernel-PCA of the 13-story building compared to the training design space. The top row graphs have the test cases in orange and overlaid with the energy gradient of the training design space. The bottom row graphs have test cases with the floor area gradient, and the training design space is in blue.

For simple NNs with actual inputs, it can be noted from Figure 10a and Figure 11a that the test design cases fall outside the training design space. Feature engineering helps the simple NN (see Figure 10b and Figure 11b) to identify similar design options within the training design space. The multilayer NN extracts features that can identify similar designs within the training design space. Furthermore, in Figure 11c, it can also be noted that certain design cases from the 13-story building fall outside the training design space. For CNNs, in Figure 11d, the 13-story building mostly falls outside the training design space. However, the generalization of the CNN is similar to the multilayer NN (see Figure 7 bottom), indicating that features that locate the design space in the appropriate region of the data distribution result in a satisfactory model generalization.

From Figure 10 and Figure 11 it can also be noted that general ML models for design can be developed when features provided or learned can identify similar design options within the data distribution. The features can be either provided through manual feature engineering/selection or extracted through a deep learning model. Hence, the characteristics of features extracted automatically or provided manually for model generalization are as follows:

- can identify similar design options within the data distribution, and
- identified similar design is mapped to appropriate response variables.

More research should be conducted to identify the training process that can incorporate these conditions during training, thereby resulting in general and reliable ML models.

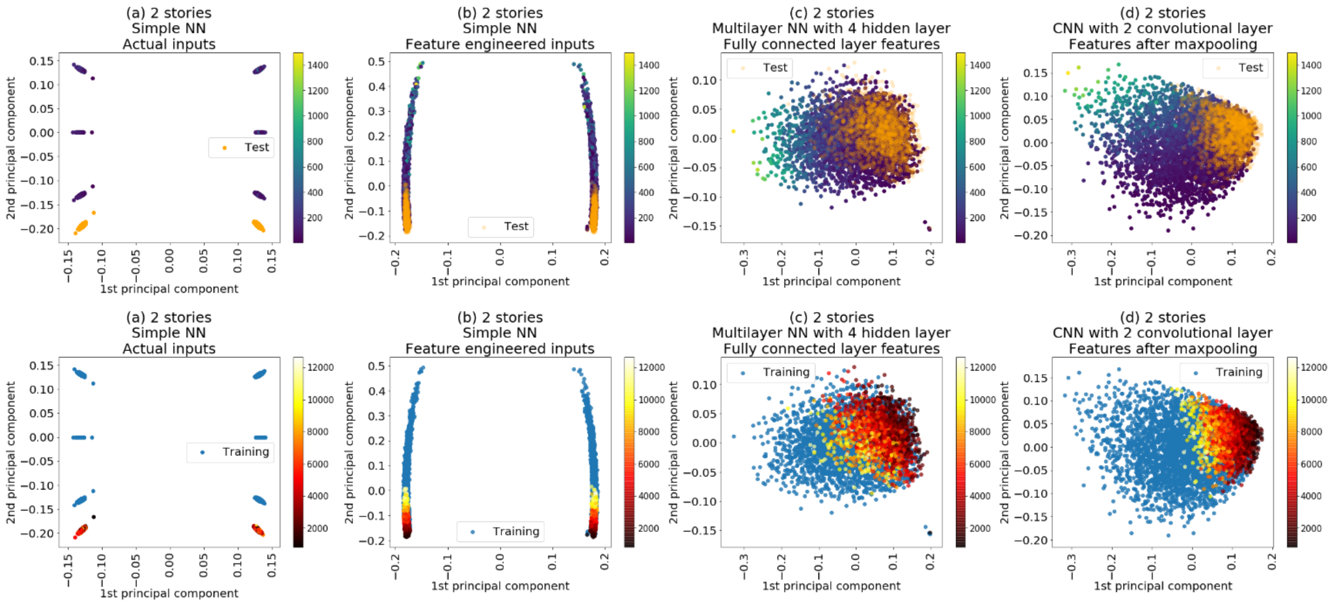

Figure 10 Principal component from kernel-PCA of training and 2-story test design space for actual inputs, multilayer NN feature, feature engineered inputs, and CNN features. (top) Orange cluster is the 2-story design space and training design space overlay with information of total heating demand (W). (bottom) Blue cluster is the training design space and 2-story design space overlay with information of total floor area (m^2)

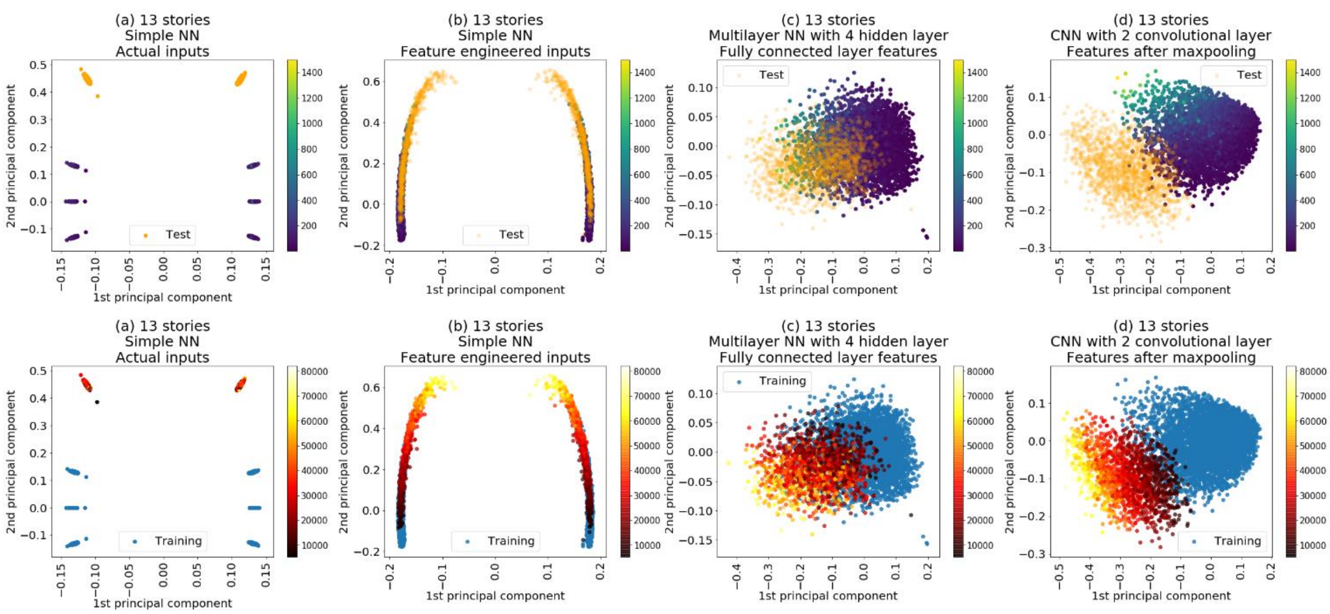

Figure 11 Principal component from kernel-PCA of training and 13-story test design space for actual inputs, multilayer NN feature, feature engineered inputs, and CNN features. (top) Orange cluster is the 13-story design space and training design space overlay with information of total heating demand (W). (bottom) Blue cluster is the training design space and 13-story design space overlay with information of total floor area (m^2)

4.3 Evaluation of design cases with BPS and ML models

In this section, energy estimates from BPS and ML models are evaluated for a design case to understand the reliability of decisions taken based on each approach and the effort required to obtain the energy estimates. Figure 12 shows the design process utilized in this study. The design decision process is for an 8-story building located in Brussels. The length and width of the 8-story building are 50 m and 60 m, respectively. The design decision process is covered in three stages. In each stage, the following action or decision is taken:

- Stage 1: The initial estimate of energy is obtained for the 8-story building with a length and width of 50 m and 60 m, respectively. All other technical specifications are assigned randomly (see Table 10), as the main object of this section is to evaluate a design process with ML models.
- Stage 2: The decision on the south and north *WWR* is taken. The south *WWR* has been decided as 0.5 and that of the north as 0.9.
- Stage 3: Designers are thinking whether to change the window g-value or insulation level. As a first option, designers evaluate a window with a g-value of 0.5 (U-value is 1.4 W/(m^2·K)). In the second option, designers evaluate a window with U-value of 0.9 W/(m^2·K) (g-value is 0.78).

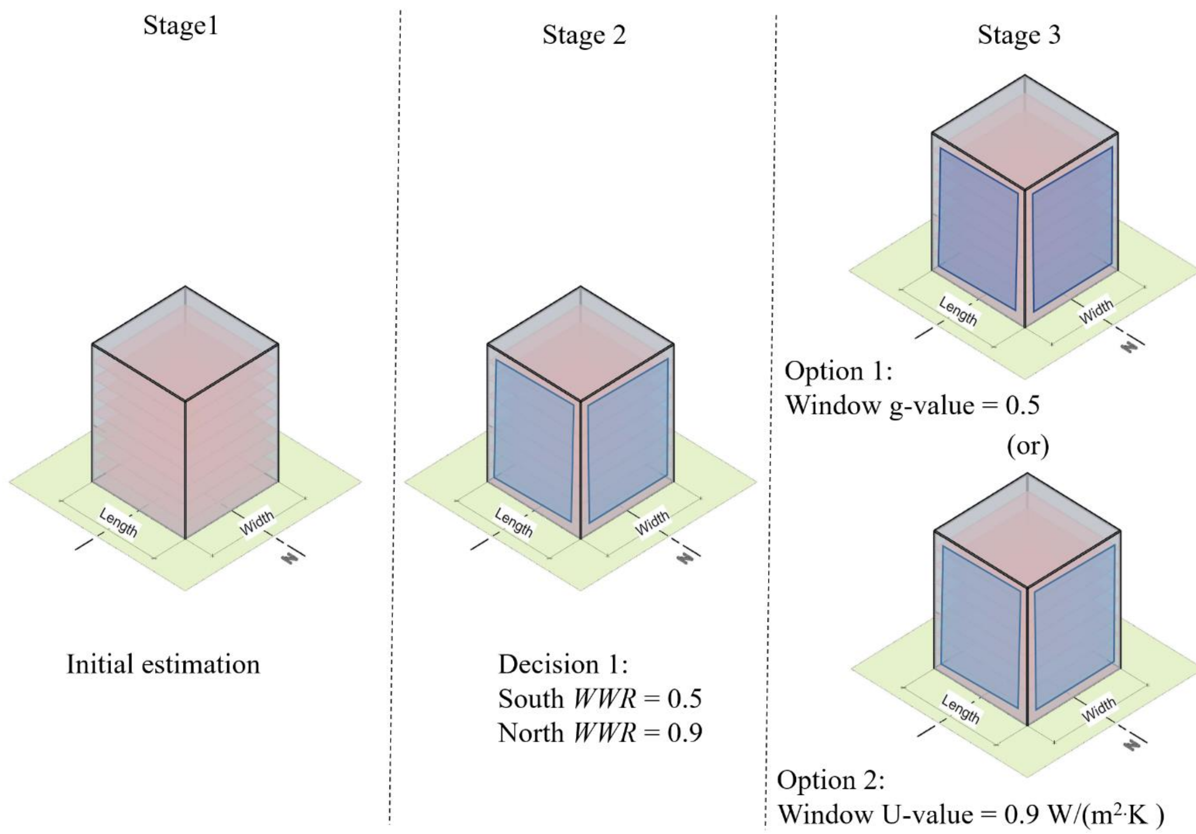

Figure 12 Case for illustrating design decisions with ML model and BPS

Table 10 Design parameters used to make the initial estimation

	Units	Stage 1: Initial estimation
Length (l)	m	50
Width (w)	m	60
Height (h)	m	4
Overhang length (l_{oh}) [2]	m	0
Window to wall ratio (*WWR*)[4]		S = 0.9, N = 0.3, E = 0.6, W = 0.9
Orientation (α)	Degree	0
Wall U-value (U_{wall})	W/(m^2·K)	0.55

[4] Varies differently in all orientations

Window U-value (U_{win})	W/(m^2·K)	1.4
Ground floor U-value (U_{floor})	W/(m^2·K)	0.44
Roof U-value (U_{roof})	W/(m^2·K)	0.32
Window g-value (g_{win})		0.78
Floor heat capacity (c_{floor})	J/(kg·K)	1107
Infiltration air change rate (n_{air})	h^{-1}	0.8
Number of floors (n_{floor})		8
Lighting heat gain (Q'_{light})	W/m^2	6
Equipment heat gain (Q'_{equip})	W/m^2	12
Chiller *COP*		3.9
Boiler efficiency (η_{Boiler})		0.95
Chiller type		Electric reciprocating chiller
Boiler pump type		Constant flow

The ML models used are simple NN with FE inputs and CNN, as these methods have a better generalization. With the CNN architecture, heating demand predictions are performed using CNN with 2 convolutional layers, and cooling demand predictions are performed using CNN with 3 convolutional layers. Figure 13 and Figure 14 show the heating and cooling demands estimated through the BPS and ML models. For heating demand predictions, the simple NN has an error range of −4% to 8% for peak predictions and 3% to 10% for total demand predictions, while the CNN has an error range of −2% to 8% for peak predictions and −5% to −14% for total demand predictions. Similarly, for cooling demand predictions, the simple NN has an error range of −4% to −12% for peak predictions and 1% to −8% for total demand predictions. The CNN has an error range of −5% to −12% for peak predictions and −4% to 4% for total demand predictions. It can be noted from Figure 13 and Figure 14 that both simple NN and CNN have similar performances. However, the advantage of CNN is the elimination of feature selection during model development, which saves time.

It can be observed from the peak heating predictions in Figure 13 (left) that the relationship learned by the ML model is not similar to that of the BPS. Therefore, taking decision on the size of heating system may not be accurate. However, by observing the total heating demand predictions from Figure 13 (right), the designer can choose Option 2 as it offers the lowest total heating demand compared with Option 1. The decision to choose Option 2 taken through ML predictions is consistent with the decision taken with BPS. Figure 14 shows the cooling demand predictions. It can be noted from Figure 14 that the changes observed in the cooling energy demand from the ML models and BPS are similar. Looking at Figure 14, the designer can select Option 1. By comparing the total heating and cooling demands, it can be observed that the design is cooling dominated and Option 1 can be chosen as it offers greater energy savings. This design decision is consistent with the use of BPS or ML models.

The advantage of ML models over BPS is the computation time required to obtain the heating and cooling energy demand. Performing one simulation using BPS takes ~2 min. Similar results can be obtained from ML models in less than 1 s. The high computation speed of the ML models together with their ability to take similar design decisions make them suitable for early design stage predictions.

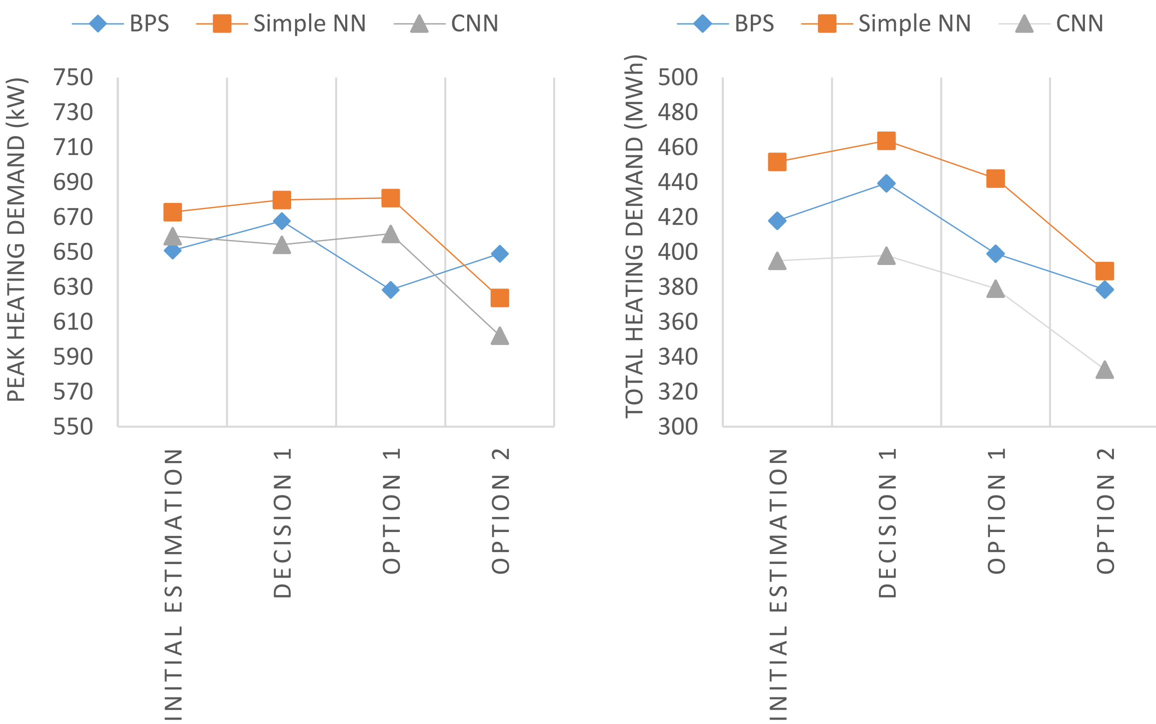

Figure 13 Estimation of heating demand from BPS and ML models: (left) peak heating demand and (right) total heating demand

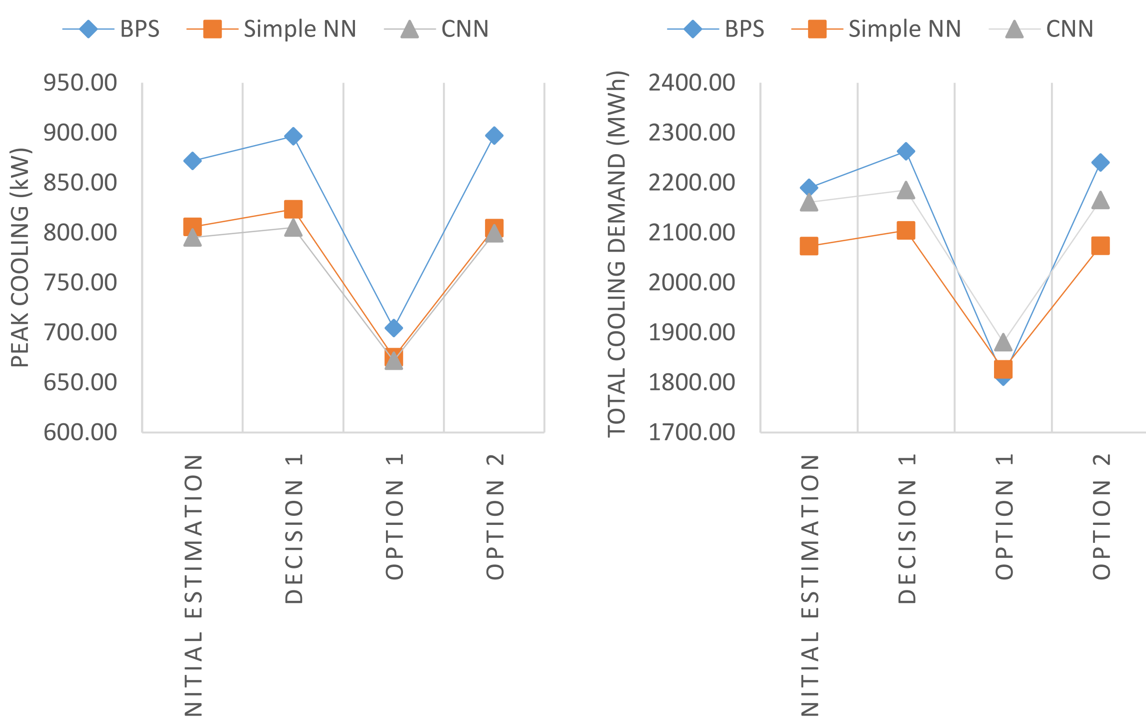

Figure 14 Estimation of cooling demand from BPS and ML models: (left) peak cooling demand and (right) total cooling demand

Figure 15 shows the location of the evaluated design options in the heating data distribution. Figure 15 (top) is overlaid with information of total heating demand within the data distribution, whereas Figure 15 (bottom) is overlaid with information of peak heating demand within the data distribution. Figure 15a shows the data distribution, which is the result of feature engineering and selection for a simple NN and Figure 15b shows the data distribution determined by the features extracted by the CNN. The location of design options within the cooling model is similar to observations present within the heating model; hence, they are not shown in this study. The red point in Figure 15 (top, b) shows the initial design option that falls in the data distribution region of 200 MWh to 400 MWh. The CNN predicts a total heating demand of 395 MWh. Similarly, Decision 1, i.e., the green point (approximately on top of red point) in Figure 15 (top, b) falls in the data range of 200 MWh to 400 MWh. The CNN predicts a total heating demand of 398 MWh. The movement of design options with the simple NN with feature engineered inputs (see Figure 15a) shows a similar pattern as observed in the CNN. Finally, such visualizations enables justification of a prediction.

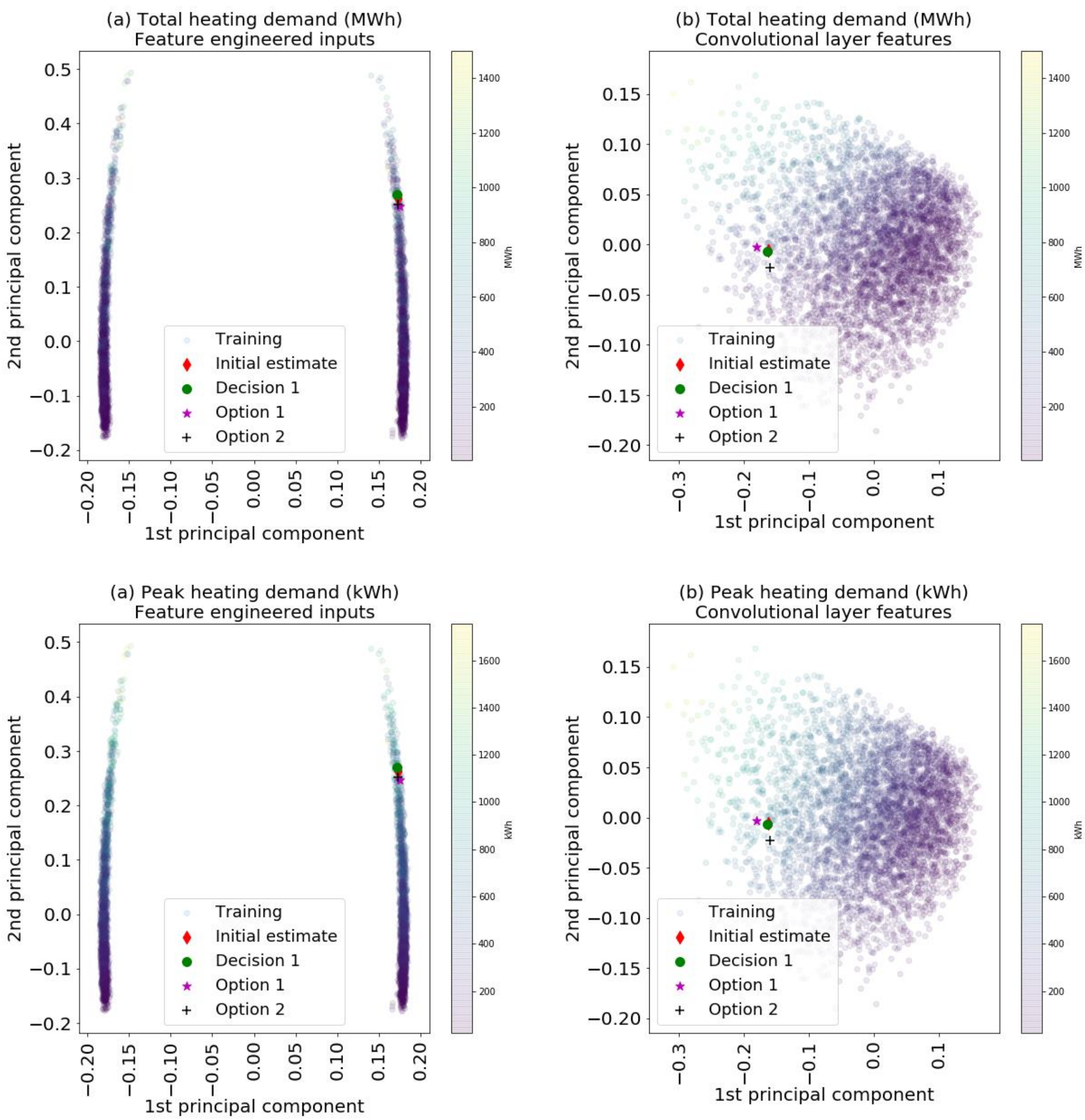

Figure 15 Location of design options with respect to the heating data distribution: (top) overlaid with total heating demand information and (bottom) overlaid with peak heating demand information

5 Discussion

Developing an ML model with a satisfactory generalization performance is crucial for the effective utilization of ML models in the design stage performance analysis. Results indicate that manual feature engineering and selection play a vital role in extending the model reusability of simple NNs. In addition, deep learning model architectures could extract features from data, which extends their reusability in design. Irrespective of the use of simple or more advanced ML methods, for an ML model to generalize in unseen design, it should be able to identify similar design options within the data distribution.

Although most resulting ML models support decisions well as shown in Figure 13, there are some models that represent relationships that are not in alignment with the BPS simulation and lead to deviations in the decision process (see Figure 13 (left)). Nonetheless, the prediction error in specific design options are within acceptable ranges. Hence, such deviations can be mitigated by introducing prediction intervals within the ML prediction process. Prediction intervals provide information on uncertainties present within an ML model prediction, allowing for predictions with high uncertainty to be viewed critically. Except for some deviations in peak heating predictions, evaluations of specific design options show that other parameters have learned appropriate relationships. Incorporating prediction intervals for these parameters can improve the reliability of decisions made using the ML models. More research on methods of incorporating design stage prediction intervals needs to be done.

The evaluated design cases are limited to typical design cases. The reason for this limitation is that the primary objective of the paper is to propose and obtain an initial understanding of deep convolutional learning methods for early building design performance evaluation. Furthermore, by limiting to typical design cases, intuition on the working of deep learning methods for building design evaluation is obtained (see Figure 15). Based on this intuition, appropriate *DM* to extract features from data for more complex design cases can be derived. Further research on extending the current models to more complex early design case will be performed.

Nevertheless, the proposed ML models are reliable for typical early design options. Hence, for evaluating complex building designs, architects and engineers can use the (rough) predictions from the current models along with their experience to make an appropriate design decision. Even though the prediction for complex design is rough, the high computational speed of the deep learning model facilities the discussion between engineers and architects; reducing the need for rule-of-thumb knowledge.

The current ML models are reliable for typical early design stage decisions. Further research will be necessary to extend the current models to different design stage performance predictions. Research to extend ML models to other design stages can incorporate two different strategies. The first strategy will be to develop flexible components (based on component-based ML approaches presented in [7]) using a deep learning architecture to emulate data from a more detailed BPS. The advantages would be that all information required for training can be obtained from parametric simulation models and domain knowledge allowing for the development of ML models for quick design stage feedback. The drawback of using BPS data is the occurrence of model errors present within the collected data. Model error is the result of model simplification made by simulation tool like EnergyPlus and assumptions of a model developer. Such errors in data reduce the effectiveness of ML models. Therefore, methods to collect data from BPS for ML needs to be researched further. The second strategy can be the development of deep learning models from smart city data with real building energy consumption. Such models can potentially lower the performance gap for the design stage energy evaluation. One challenge to overcome with real building consumption data is missing information from key factors such as building occupancy.

In this study, feature engineering is performed using physical equations of *HF*. Simple NNs learning on features with physical significance generalize better than simple NNs with only design information. Within the deep learning model, CNNs generalize better than multilayered NNs, where CNN requires both design and physical information, indicating that feature engineering is still a relevant step in the model development process. However, the feature selection process can be eliminated, as the convolutional layer filters out irrelevant features, improving the model development process for multiple design performance indicators, because identifying and selecting such features for multiple response variables could be a time-consuming and expensive process.

For total heating demand prediction, deep learning models generalized better than simple NNs. This indicates that for complex data, deep learning methods can identify better features than manual feature engineering and selection. Within the deep learning architecture, the CNN architecture performed consistently better than multilayer NNs. Further research will be required to further understand CNNs for design stage predictions.

The *DM* utilized in this study resulted in a satisfactory model generalization. However, it is possible to derive other *DMs* with better generalization, for example, the use of hourly *HF* information instead of static *HF* information. Further research will be performed to explore other potential *DMs*.

CNNs utilize max pooling to reduce the size of the feature map (i.e., output of a convolutional layer). The current research results show that reducing the size of the feature map does not influence the model generalization. This indicates that max pooling removes features that are not related to the response variable (i.e., energy prediction). Furthermore, reducing the size of the feature map through max pooling creates an information bottleneck that induces invariance (i.e., insensitivity to irrelevant features) within a model. Based on the current results, it is not clear which aspect of input features is contributing to the generation of unrelated features. Identification of such characteristics of max pooling will provide an idea on non-relevant input features.

The kernel-PCA shows that the extracted features identify similar design options within the data distribution and mapping the similar design option to the right response variable. These characteristics of extracted features allow the deep learning model to generalize well in unseen design cases. Furthermore, methods such as kernel-PCA can be utilized for (1) steering the feature engineering and selection process even before the training process and (2) diagnosing features extracted by the deep learning model, potentially increasing the efficiency of model development. Further research will be necessary to understand the deep learning model process.

6 Conclusion

General ML models enable reliable and quick predictions, which aid in the effective design decision-making process. General ML models are ones that generalize in all possible unseen design cases. Developing such models using conventional methods requires considerable knowledge in both building performance analysis and ML. Knowledge on building performance analysis is required for manual feature engineering and selection, while knowledge on ML enables an effective development of ML models. The study shows that deep learning methods can indeed automatically learn features that results in the general model, thereby reducing the need for feature selection. Feature extraction capability of deep learning makes it easier to develop ML models for a wide range of design performance parameters.

The ML model generalization through conventional ML methods rely on manual feature engineering and selection, while deep learning models extract features automatically from data resulting in a similar or better generalization. In both cases, model generalization is dependent on the feature's ability to identify similar design options within the data distribution. The need for ML to identify similarity within the data distribution makes ML model predictions top-down. For example, energy demand predictions from ML is based on energy demand of a similar design option. In contrast, BPS models make predictions based on a bottom-up approach, in which energy demand prediction results from hierarchical interactions (such as *HF*s) within the model. However, both approaches are prone to biases, which can mislead the designer. The quality of BPS prediction depends on the quality of inputs and model complexity. The quality of ML model prediction depends on the quality of the data utilized in the model development and quality of input features engineered, indicating that making decision from both the BPS and ML models can remove potential model-based biases. Hence, an ensemble of BPS and ML models can be a potential direction for model development, making BPS and ML methods complimentary technologies rather than competing ones. However, the computational efforts required to make predictions from ML and BPS are different. Hence, intelligent ensemble methods that can exploit the strengths of ML are necessary. Finally, based on the current research results, the designer can rely on the ML models for a quick assessment of the design and design strategy and moves toward BPS for a more detailed analysis. This will enable a model-driven design decision-making process, rather than reliance on rule-of-thumb knowledge.

7 Acknowledgments

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 by ERC Advanced Grant E-DUALITY (787960), KU Leuven C1, FWO G.088114N.

8 References

[1] G. Zapata-Lancaster and C. Tweed, "Tools for low-energy building design: an exploratory study of the design process in action," *Archit. Eng. Des. Manag.*, vol. 12, no. 4, pp. 279–295, 2016.

[2] C. Bleil de Souza, "Contrasting paradigms of design thinking: The building thermal simulation tool user vs. the building designer," *Autom. Constr.*, vol. 22, pp. 112–122, Mar. 2012.

[3] S. Attia, E. Gratia, A. De Herde, and J. L. M. Hensen, "Simulation-based decision support tool for early stages of zero-energy building design," *Energy Build.*, vol. 49, pp. 2–15, Jun. 2012.

[4] P. Shiel, S. Tarantino, and M. Fischer, "Parametric analysis of design stage building energy performance simulation models," *Energy Build.*, vol. 172, pp. 78–93, Aug. 2018.

[5] M. N. Hamedani and R. E. Smith, "Evaluation of Performance Modelling: Optimizing Simulation Tools to Stages of Architectural Design," *Procedia Eng.*, vol. 118, pp. 774–780, 2015.

[6] L. Van Gelder, P. Das, H. Janssen, and S. Roels, "Comparative study of metamodelling techniques in building energy simulation: Guidelines for practitioners," *Simul. Model. Pract. Theory*, vol. 49, pp. 245–257, 2014.

[7] P. Geyer and S. Singaravel, "Component-based machine learning for performance prediction in building design," *Appl. Energy*, vol. 228, pp. 1439–1453, Oct. 2018.

[8] S. Singaravel, J. Suykens, and P. Geyer, "Deep-learning neural-network architectures and methods: Using component-based models in building-design energy prediction," *Adv. Eng. Informatics*, vol. 38, pp. 81–90,

Oct. 2018.

[9] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[10] S. M. C. Magalhães, V. M. S. Leal, and 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.*, vol. 151, pp. 332–343, 2017.

[11] F. Ascione, N. Bianco, C. De Stasio, G. M. Mauro, and 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.*, vol. 146, pp. 200–219, 2017.

[12] J. Yang, H. Rivard, and R. Zmeureanu, “On-line building energy prediction using adaptive artificial neural networks,” *Energy Build.*, vol. 37, no. 12, pp. 1250–1259, Dec. 2005.

[13] B. B. Ekici and U. T. Aksoy, “Prediction of building energy consumption by using artificial neural networks,” *Adv. Eng. Softw.*, vol. 40, no. 5, pp. 356–362, May 2009.

[14] A. Kusiak and G. Xu, “Modeling and optimization of HVAC systems using a dynamic neural network,” *Energy*, vol. 42, no. 1, pp. 241–250, Jun. 2012.

[15] Z. Hou, Z. Lian, Y. Yao, and X. Yuan, “Cooling-load prediction by the combination of rough set theory and an artificial neural-network based on data-fusion technique,” *Appl. Energy*, vol. 83, no. 9, pp. 1033–1046, 2006.

[16] A. Chari and S. Christodoulou, “Building energy performance prediction using neural networks,” *Energy Efficiency*, pp. 1–13, 2017.

[17] J. Yao, “Prediction of Building Energy Consumption at Early Design Stage Based on Artificial Neural Network,” *Adv. Mater. Res.*, vol. 108, pp. 580–585, May 2010.

[18] A. Lazrak *et al.*, “Development of a dynamic artificial neural network model of an absorption chiller and its experimental validation,” *Renew. Energy*, vol. 86, pp. 1009–1022, 2016.

[19] A. H. Neto and F. A. S. Fiorelli, “Comparison between detailed model simulation and artificial neural network for forecasting building energy consumption,” *Energy Build.*, vol. 40, no. 12, pp. 2169–2176, Jan. 2008.

[20] S. Paudel *et al.*, “A relevant data selection method for energy consumption prediction of low energy building based on support vector machine,” *Energy Build.*, vol. 138, pp. 240–256, 2017.

[21] F. Zhang, C. Deb, S. E. Lee, J. Yang, and K. W. Shah, “Time series forecasting for building energy consumption using weighted Support Vector Regression with differential evolution optimization technique,” *Energy Build.*, vol. 126, pp. 94–103, 2016.

[22] B. Dong, C. Cao, and S. E. Lee, “Applying support vector machines to predict building energy consumption in tropical region,” *Energy Build.*, vol. 37, no. 5, pp. 545–553, 2005.

[23] Q. Li, Q. Meng, J. Cai, H. Yoshino, and A. Mochida, “Applying support vector machine to predict hourly cooling load in the building,” *Appl. Energy*, vol. 86, no. 10, pp. 2249–2256, Oct. 2009.

[24] H.-X. Zhao and F. Magoulès, “Feature Selection for Predicting Building Energy Consumption Based on Statistical Learning Method,” *J. Algorithm. Comput. Technol.*, vol. 6, no. 1, pp. 59–77, 2012.

[25] G. K. F. Tso and K. K. W. Yau, “Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks,” *Energy*, vol. 32, no. 9, pp. 1761–1768, Sep. 2007.

[26] C. Zhang, L. Cao, and A. Romagnoli, “On the feature engineering of building energy data mining,” *Sustain. Cities Soc.*, vol. 39, pp. 508–518, May 2018.

[27] T. Catalina, J. Virgone, and E. Blanco, "Development and validation of regression models to predict monthly heating demand for residential buildings," *Energy Build.*, vol. 40, no. 10, pp. 1825–1832, Jan. 2008.

[28] I. Jaffal and C. Inard, "A metamodel for building energy performance," *Energy Build.*, vol. 151, pp. 501–510, Sep. 2017.

[29] K. Amasyali and N. Gohary, "A review of data-driven building energy consumption prediction studies," *Renew. Sustain. Energy Rev.*, vol. 81, pp. 1192–1205, Jan. 2018.

[30] C. Fan, F. Xiao, and Y. Zhao, "A short-term building cooling load prediction method using deep learning algorithms," *Appl. Energy*, vol. 195, pp. 222–233, 2017.

[31] D. L. Marino, K. Amarasinghe, and M. Manic, "Building energy load forecasting using Deep Neural Networks," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 7046–7051.

[32] C. Li *et al.*, "Building Energy Consumption Prediction: An Extreme Deep Learning Approach," *Energies*, vol. 10, no. 10, p. 1525, Oct. 2017.

[33] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling, "Deep learning for estimating building energy consumption," *Sustain. Energy, Grids Networks*, vol. 6, pp. 91–99, Jun. 2016.

[34] B. Zhong, X. Xing, P. Love, X. Wang, and H. Luo, "Convolutional neural network: Deep learning-based classification of building quality problems," *Adv. Eng. Informatics*, vol. 40, pp. 46–57, Apr. 2019.

[35] C. Lu, Z. Wang, and B. Zhou, "Intelligent fault diagnosis of rolling bearing using hierarchical convolutional network based health state classification," *Adv. Eng. Informatics*, vol. 32, pp. 139–151, Apr. 2017.

[36] W. Fang *et al.*, "A deep learning-based approach for mitigating falls from height with computer vision: Convolutional neural network," *Adv. Eng. Informatics*, vol. 39, pp. 170–177, Jan. 2019.

[37] W. Fang, L. Ding, B. Zhong, P. E. D. Love, and H. Luo, "Automated detection of workers and heavy equipment on construction sites: A convolutional neural network approach," *Adv. Eng. Informatics*, vol. 37, pp. 139–149, Aug. 2018.

[38] B. Doshi-Velez, Finale and Kim, "Towards a rigorous science of interpretable machine learning," *arXiv Prepr.*, 2017.

[39] S. Singaravel, P. Geyer, and 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.

[40] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. 2016.

[41] A. Achille and S. Soatto, "Emergence of invariance and disentanglement in deep representations," in *2018 Information Theory and Applications Workshop, ITA 2018*, 2018, vol. 18, pp. 1–34.

[42] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *20th International Conference on Artificial Neural Networks (ICANN)*, 2010, vol. 6354 LNCS, no. PART 3, pp. 92–101.

[43] J. Oh, J. Wang, and J. Wiens, "Learning to Exploit Invariances in Clinical Time-Series Data using Sequence Transformer Networks," *Proc. Mach. Learn. Res.*, vol. 85, pp. 1–15, 2018.

[44] C. J. Hopfe and J. L. M. Hensen, "Uncertainty analysis in building performance simulation for design support," *Energy Build.*, vol. 43, no. 10, pp. 2798–2805, Oct. 2011.

[45] T. Østergård, R. L. Jensen, and S. E. Maagaard, "Building simulations supporting decision making in early design - A review," *Renewable and Sustainable Energy Reviews*, vol. 61. Pergamon, pp. 187–201, 01-Aug-

2016.

[46] A. Paszke *et al.*, "Automatic differentiation in pytorch," 2017.

Conflict of Interest and Authorship Conformation Form

Please check the following as appropriate:

- All authors have participated in (a) conception and design, or analysis and interpretation of the data; (b) drafting the article or revising it critically for important intellectual content; and (c) approval of the final version.
- This manuscript has not been submitted to, nor is under review at, another journal or other publishing venue.
- The following authors have affiliations with organizations with direct or indirect financial interest in the subject matter discussed in the manuscript:

Author's name	Affiliation
Sundaravelpandian Singaravel	KU Leuven
Johan Suykens	KU Leuven
Philipp Geyer	KU Leuven