Learning Based Compact Thermal Modeling for Energy-Efficient Smart Building Management

(invited)

Hengyang Zhao*, Daniel Quach*, Shujuan Wang†, Hai Wang§, Haibao Chen¶, Xin Li‡, and Sheldon X.-D. Tan*

* Department of Electrical and Computer Engineering, University of California, Riverside, CA 92521
† Department of Mechanical Engineering, University of California, Riverside, CA 92521
‡ Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213
§ School of Microelectronic and Solid-State Electronics
University of Electronic Science and Technology of China, Chengdu, China 610054
¶ Department of Micro/Nano-electronics, Shanghai Jiao Tong University, Shanghai, China 200240

***Abstract*—In this article, we propose a new behavioral thermal modeling method for fast building performance analysis, which is critical for energy-efficient smart building control and management. The new approach is based on two recurrent neutral network architecture to obtain the compact nonlinear thermal models for complicated building. We start with a more realistic building simulation program, EnergyPlus, from Department of Energy, to model some practical buildings such as office buildings and data centers. EnergyPlus can model the various time-series inputs to a building such as ambient temperature, heating, ventilation, and air-conditioning (HVAC) inputs, power consumption of electronic equipment, lighting and number of occupants in a room sampled in each hour and produce resulting temperature traces of zones (rooms). In this work, we apply two recurrent neural network (RNN) architectures to build the non-linear compact thermal model of the building: one is non-linear state-space RNN architecture (NLSS), which has global feedbacks, and the other one is Elman's RNN architecture (ELNN), which has local feedbacks in each layer. We give a simple formula to calculate the RNN layer number, layer size to configure RNN architecture to avoid overfitting and underfitting problems. A cross-validation based training technique is further applied to improve predictable accuracy of models. Experimental results from a case study of three buildings show that ELNN and NLSS can both build very accurate building thermal models for the 2-zone and 5-zone building cases: both of them have average errors from around 1% to 1.5% for the two buildings. For the more complex 6-zone building case, ELNN outperforms NLSS with maximum errors 16% against 23%. But both methods have 2.2% average errors.**

I. Introduction

It is estimated that building section accounts for about 40% energy consumption in United States. The building section is also responsible for 70% of electricity use. About 50% of the energy consumed in buildings are directly related to space heating, cooling and ventilation [1]. As a result, it is important to control the heating, ventilation and air conditioning (HVAC) system in a more energy-efficient way.

Smart buildings today have sophisticated and distributed control systems as part of a Building Automation System (BAS). The task of a BAS is to maintain building climate within a specified range, control the lighting based on the occupancy schedule, and monitor the system performance and failures. To achieve these tasks, a BAS needs complicated and advanced control techniques such as Extended Kalman Filter (EKF) method to minimize energy consumption of the HVAC system and satisfy the temperature and demand constraints [27].

However, the EKF based control could have high computational requirements depending on the number of control variables, the time horizon for the optimization, the discretization for the control decisions and model complexity. One critical issue in effective EKF control is to have compact nonlinear thermal model of large buildings.

Building performance simulation has been well studied in the past [26]. Simulation programs such as EnergyPlus [7] from U.S. Department of Energy, and TRNSYS [3], which are both based on the first principle of dynamic heat transfer and air flow have been developed. However, for practical buildings, those simulators have very high computational cost due their detailed modeling of the underlying physics and interactions among the components of buildings, which makes them less suitable for MPC based thermal control for smart buildings.

To mitigate this problem, compact thermal modeling methods for buildings have been proposed recently [8], [10], [21]. Existing approaches include the reduced order modeling method using the classic truncated balanced realization method [21], aggregation-based reduction approach, which performs localized reduction (aggregation) so that some network properties can be preserved [8], and the ad-hoc model reduction method, which extracts the basic linear dynamics of thermal behaviors of a building from the EnergyPlus program [10]. All those existing compact modeling approaches can be viewed as the white-box models, which start with accurate and detailed models from the first principle and then perform approximation to obtain compact models via model order reduction, aggregation or ad-hoc dynamics extraction. But those methods suffer from several problems. First, those methods need to know the detailed structures or equations of the thermal systems to start with, which need steep learning curves to learn and extracts. For commercial building

This work is supported in part by NSF grants under No. CCF-1255899, in part by NSF Grant under No. CCF-1527324, and in part by Semiconductor Research Corporation(SRC) Grant under No. 2013-TJ-2417.

978-1-4673-8388-2/15/$31.00 ©2015 IEEE

simulation tools, this will become impossible. Second, they may suffer significant accuracy loss during the reduction process as many assumptions were made such as linear system models [10], ignorance of cooling and exhaust loads [8].

Recently a number of top-down black-box based behavioral modeling technique have been proposed to build thermal models for chips and packages of VLSI systems. Existing works consist of the matrix pencil method [17], [24] and the subspace identification method [9], [28]. The major advantage of such pure behavioral modeling methods are their flexibility and simplicity as no physical restrictions and assumptions are made or required for the models. They are also very accurate as the training is based on accurate data from measurement or detailed numerical simulation. However, existing approaches can not deal with the nonlinearity of the complicated dynamic thermal systems such as building.

In this work, we apply the neural network based modeling technique to build the dynamic thermal models for complicated buildings. The training data is obtained from the EnergyPlus software package [7], which is a suite of algorithms that calculate the energy required to operate a building and its resulting thermal behavior based on numerous considerations ranging from the specifics of the structure, ambient temperature, heating, ventilation, and air-conditioning (HVAC) inputs, power consumption of electronic equipment, lighting, number of occupants in a room, resulting temperature traces of zones (rooms), and other factors that we are interested in. We explore two recently proposed recurrent neural network (RNN) structures to build the time-dependent nonlinear thermal models. One structure is the non-linear state space RNN architecture (NLSS), which has a global feedback from the last hidden layer to the first hidden layer. The other structure is the Elman's RNN architecture (ELNN). Comparing with NLSS, ELNN has a local feedback on each hidden layer, which introduces more internal states and connection weights than NLSS with the same hidden layer number and layer size. We give a simple formula to calculate the RNN layer number, layer size to configure RNN architecture to avoid overfitting and underfitting problems. Experimental results are performed on three buildings with different number of zones. We show that NLSS and ELNN both perform pretty well on the 2-zone and 5-zone buildings: they have similar average errors from around 1% to 1.5%. For the more complex 6-zone building case, ELNN outperforms NLSS with average errors of 2.2% against 3.1%. But both methods have 2.2% average errors.

II. Review of EnergyPlus for energy simulation of building

In this section, we review the EnergyPlus software program, which provide accurate input and output traces from buildings for the new thermal modeling algorithm.

The EnergyPlus software package [7] is a suite of algorithms that calculate the energy required to operate a building and its resulting thermal behavior based on numerous considerations ranging from the specifics of the structure, to heat sources and sinks within the building, and weather. EnergyPlus consists of an integrated solution manager which manages the calculation of the heat balance of various surfaces in the building, the heat balance of the air, and the heat balance on the mechanical systems. The solution to each of these three elements are calculated separately and communicated to each other using the manager at each time-step. Due to its modularity, it's easy to establish links to other program links such as Google SketchUp [2] for 3D building display.

An input data file (IDF) and weather file are needed for the EnergyPlus simulation. The IDF includes all the information of the building such as size, structure, position and the HVAC subsystem etc. The IDF editor in EnergyPlus can be used to change parameters of the building, the schedule of the HVAC subsystem and also the output information. The selected output information will be generated in the spreadsheet file after running the simulation.

Fig. 1 shows the side view and the top view of a data center building with 2 rooms and HVAC modeled in the EnergyPlus. The heat sources for this building can be HVAC, light, occupants, electric equipment, air filtration etc. The room temperature is also affected by the weather (ambient temperature). The room temperature is controlled by the HVAC system with coil and fan as shown in Fig. 2. The air of a room are cooled or heated in the coil and goes back to the zone. The coil itself is a heat exchanger. Besides the air loop, the coil has a water loop coming from the boiler or chiller to control the temperature of the air.

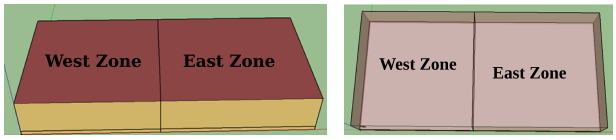


Fig. 1: The 2-zone office building (a) side view (b) top view.

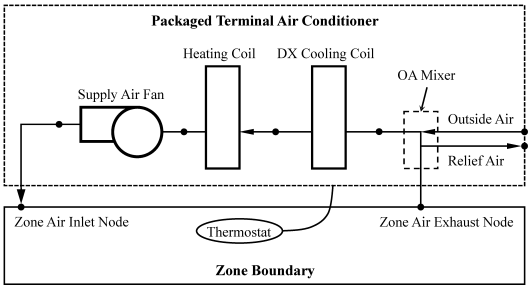


Fig. 2: A HVAC in a zone with coil and fan system.

Fig. 3 shows the simulated temperature changes and input changes over one week from EnergyPlus for a data center building with the 2 zones (rooms) as shown in Fig. 1.

In this particular example, we have 6 inputs for each output, which is the temperature of one zone (room). The 6 inputs are shown in Fig. 3. The main heating sources comes from information technology equipment (ITE) as this is a date center building. ITE is a group of equipment such as computers, monitors, servers, printers, network hubs etc. With the increasing of computing capacity in the data center, ITE becomes a significant power consumer and the cooling of ITE is required. Inputs 5 and 6 comes from HVAC for the cooling.

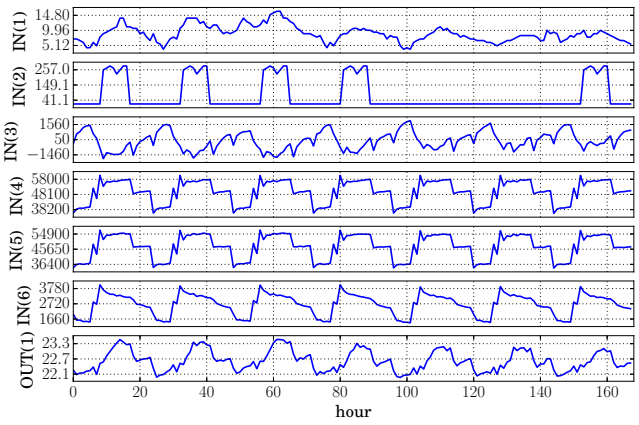


Fig. 3: The input and simulated temperature output data sample in one week from EnergyPlus. IN(1): ambient temperature; IN(2): the total heating rate of lights in the zones; IN(3): the total power generated by electric machines; IN(4): total heat gain rate of the area with information technology equipment (ITE) to the zones; IN(5): air system sensible cooling rate for zones; IN(6): air system sensible cooling rate for plenum; OUT(1): zone operative temperature (hourly).

Notice that ambient temperature is also input for our thermal model.

We want to stress that fundamentally thermal behavior of building systems is typically nonlinear (at least weakly nonlinear) due to the temperature-dependent properties of the building materials and thermal radiation effects [4], [25]. As a result, nonlinear modeling is preferred for accurate temperature control and management.

III. Review of Recurrent Neural Networks

Learning based techniques such as neural networks, which composed of multiple processing layers, can learn representations of data with multiple levels of abstraction. Deep learning techniques with consist of many layers recently have dramatically improve the state-of-the-art in speech recognition and image recognition [22].

Behavioral modeling can be viewed as a supervised learning in neural network. The main advantage of the neural networks is its wide application for both linear and nonlinear systems. The universal approximation capability of feed-forward neural networks (FNN) has been proved in [16] to show that any Borel measurable function can be approximated with any arbitrary accuracy by an FNN using squashing activation functions.

Specifically, let $\boldsymbol{u} = \{u_1, u_2, \cdots, u_p\}$ denote the input $p \times 1$ vector, $\boldsymbol{y} = \{y_1, y_2, \cdots, y_q\}$ denote the output $q \times 1$ vector, a layer-wise structured FNN without bias can be described as

$$\begin{aligned} \boldsymbol{a}_1 &= \boldsymbol{f}_1(\boldsymbol{u}\boldsymbol{W}^{(\mathrm{IN},1)}), \\ \boldsymbol{a}_2 &= \boldsymbol{f}_2(\boldsymbol{a}_1\boldsymbol{W}^{(1,2)}), \cdots, \\ \boldsymbol{a}_i &= \boldsymbol{f}_i(\boldsymbol{a}_i\boldsymbol{W}^{(i-1,i)}), \cdots, \\ \boldsymbol{a}_k &= \boldsymbol{f}_k(\boldsymbol{a}_{k-1}\boldsymbol{W}^{(k-1,k)}), \\ \boldsymbol{y} &= \boldsymbol{a}_k\boldsymbol{W}^{(k,\mathrm{OUT})} \end{aligned} \tag{1}$$

where activation function $\boldsymbol{f}$ is element-wise squashing operator such as sigmoid or hyperbolic tangent function; vector $\boldsymbol{a}_i$ is the intermediate activation result of each layer; matrix $\boldsymbol{W}^{(\cdot,\cdot)}$ is the weighting matrix connecting adjacent layers. FNN with bias requires each activation result vector $\boldsymbol{a}_i$ to be appended with a fixed unit value before it is fed into next level of calculation, and dimension of $\boldsymbol{W}^{(\cdot,\cdot)}$ also needs to be adjusted accordingly.

A. Review of recurrent neural networks

FNN can not work for the time domain models, which depend on the history of the inputs of the models instead of current inputs. As a result, recurrent neural networks (RNN) has been proposed [31].

A recurrent neural network (RNN) is constructed by introducing internal status holders to a memory-less network so that it can deal with time-series data. The internal status holders store outputs of designated neurons and usually function as feedbacks into other neurons. The application of feedback enables RNNs to acquire time-dependent state representations, making them suitable devices for applications like time-dependent non-linear prediction, plant control, etc. [13]

There are many RNN structures proposed by varying the form of global recurrent feedback [11], [13], [23], [29]. Nonlinear autoregressive with exogenous inputs (NARX) model [23] uses history of system input and output as feedback to the multilayer perception. The estimated system output has the form of

$$\begin{aligned} \hat{\boldsymbol{y}}(n+1) = \boldsymbol{\varphi}\big(&\boldsymbol{y}(n), \boldsymbol{y}(n-1), \cdots, \boldsymbol{y}(n-q+1), \\ &\boldsymbol{u}(n), \boldsymbol{u}(n-1), \cdots, \boldsymbol{u}(n-q+1)\big) \end{aligned} \tag{2}$$

where q denotes the order of the unknown system; $\boldsymbol{y}$ and $\boldsymbol{u}$ are respectively the system output and input; $\hat{\boldsymbol{y}}$ denotes the estimated system output, $\boldsymbol{\varphi}$ denotes the non-linear function approximated by the NARX network.

The second RNN model is the non-linear state space model (NLSS) [13], taking the following form:

$$\boldsymbol{x}(n+1) = \boldsymbol{\varphi}\left(\boldsymbol{x}(n), \boldsymbol{u}(n)\right) \tag{3}$$

$$\boldsymbol{y}(n) = \boldsymbol{C}\boldsymbol{x}(n) \tag{4}$$

which essentially is the general nonlinear continuous-time dynamic model useful for modern control theory to study the dynamic behavior.

The third RNN network is Elman's simple recurrent network (ELNN) architecture [11] , which applies local recurrent feedback on each layer of neurons instead of the global feedbacks as in the NLSS model. Similar state space dynamic neural network (SSDNN) has been applied to model non-linear microwave integrated circuits with acceptable errors using both

clean and noisy input data [5], [6]. SSDNN introduces feed-forward neural network into linear state space equation in form of

$$\dot{\boldsymbol{x}}(t) = -\boldsymbol{x}(t) + \tau\boldsymbol{\varphi}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \quad (5)$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) \quad (6)$$

to describe the transient behavior of non-linear circuits. Recurrent multilayer perceptron (RMLP) [29] has been proposed, which uses recurrent feedback between layer groups in a more coarse-grained sense.

B. Review of neural network training

In theoretical aspect, training a neural network represented in equation 1 is equivalent to the optimization problem to minimize cost function (without bias connections, without regulation terms):

$$\boldsymbol{J}\left(\boldsymbol{W}^{(\mathrm{IN},1)}, \boldsymbol{W}^{(1,2)}, \cdots, \boldsymbol{W}^{(k,\mathrm{OUT})}\right) = \sum_{j=1}^{m} \left\|\boldsymbol{y}_j - \hat{\boldsymbol{y}}_j\right\| \quad (7)$$

where $\hat{\boldsymbol{y}}_i$ is neural network output which can be explicitly written in a nested activation form

$$\hat{\boldsymbol{y}} = \boldsymbol{f}_k(\boldsymbol{f}_{k-1}(\boldsymbol{f}_{k-2}(\cdots \boldsymbol{f}_2(\boldsymbol{f}_1(\boldsymbol{u}\boldsymbol{W}^{(\mathrm{IN},1)})\boldsymbol{W}^{(1,2)})\boldsymbol{W}^{(2,3)} \cdots) \boldsymbol{W}^{(k-2,k-1)})\boldsymbol{W}^{(k-1,k)})\boldsymbol{W}^{(k,\mathrm{OUT})} \quad (8)$$

Therefore the neural network training problem can be solved by applying existing optimization method such as gradient decent, Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [14], and the Quasi-Newton method on the cost function $\boldsymbol{J}$. In practice, algorithms with lower computational cost has been developed. Back-propagation algorithm is a widely-used algorithm and has been well studied [15]. It collects errors in weighting matrices $\boldsymbol{W}^{(\cdot,\cdot)}$ in a backward propagation, after the errors of output vectors have been observed in each epoch. Based on the back-propagation algorithm, many improvements have been developed such as the resilient back-propagation (RProp) method [30], which is more adaptive approach, and a further improvement method: RPropMinus [19], which has an overall better performance in reducing average error in late training phase. The back-propagation algorithm family has also been extended to train recurrent neural networks. Back-propagation through time (BPTT) [34] unfolds every network activation of a continuous sequence. Back-propagation through structure (BPTS) [12] delivers more computational efficiency on arbitrary structured networks.

IV. RNN-BASED THERMAL MODELING FOR BUILDING

In this section, we apply the two RNN architectures for building the nonlinear thermal models for building. We also describe the structures of the two networks, how the gold referencing data is computed, and the detailed works on training the networks.

Specifically, we apply a non-linear state space RNN architecture (NLSS) and Elman's RNN architecture (ELNN) for our thermal modeling problem. The reason for choosing the two networks is that the two networks represent the two major RNN architectures (NLSS for global feedback and ELNN for local feedback) so that we can perform some good comparison for the thermal modeling problems.

In the following, we briefly further discuss the two RNN networks. As shown in Fig. 4, the non-linear state space RNN (NLSS) is constructed to achieve a direct representation of equation (3) and (4). By introducing a recurrent connection feeding back from the last hidden layer to the first hidden layer, the internal state vector $\boldsymbol{x}$ is maintained internally by the network. Fig. 5 depicts the second architecture of

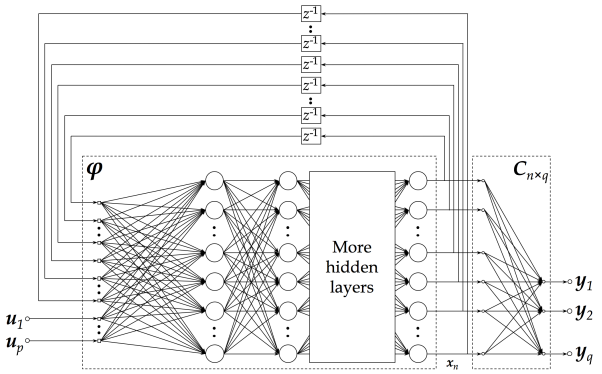


Fig. 4: Non-linear state space recurrent neural network (NLSS) architecture (bias units are not included).

the Elman's network (ELNN) with local feedback on each hidden layer. In comparison to the non-linear state space architecture, given the same number of hidden layers and layer sizes, the Elman's network introduces more internal states and connection weights.

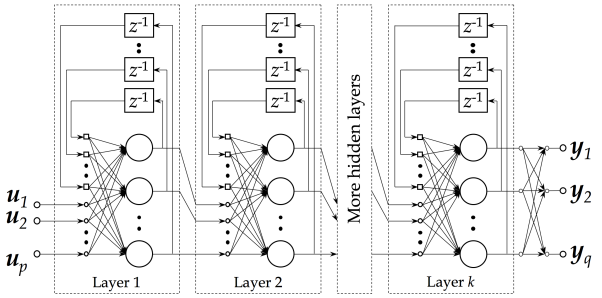


Fig. 5: Elman's recurrent neural network (ELNN) architecture with local feedbacks (bias units are not included).

For neural network based modeling and classification methods, one critical issue is to find the most suitable architecture and proper training to avoid overfitting and underfitting problems. If we select a more complicated structure than needed for the system, we may model excessive noise instead of the true system dynamics, this is called overfitting. On the other hand, if we use an over simplified model than required for the system, we may run into an underfitting problem.

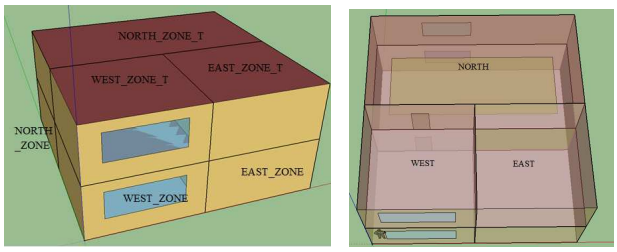
To mitigate these problems, we first select the proper architectures in terms of number of neurons per layer and layers based on a simple rule after some experiments with the building modeling problems for the two networks. For the Elman's network in Fig. 5, we use two hidden layers with local feedbacks. The number of neurons in first hidden layer is determined by the number of input: $N_{\mathrm{ELNN1}} = \frac{1}{5}N_{\mathrm{IN}} + 5$;

Sequence configuration						NLSS (%)		ELNN (%)	
1–2	3–4	5–6	7–8	9–10	11–12	Avg.	Max.	Avg.	Max.
T	T	T	T	T	V	2.2	23	2.6	21
T	T	T	T	V	T	8.0	74	3.2	33
T	T	T	V	T	T	3.9	28	2.2	16
T	T	V	T	T	T	4.9	81	2.5	23
T	V	T	T	T	T	4.5	25	2.2	24
V	T	T	T	T	T	4.5	26	2.5	31

TABLE I: 6-fold cross validation configuration (T: training, V: validation) and relative error statistics.

the number of neurons in second hidden layer is determined by the number of output: $N_{\text{ELNN2}} = 2 \times N_{\text{OUT}}$. N_{IN} and N_{OUT}, denoting the number of input and output of Elman's network, (some building examples are in Table II). For the non-linear state space RNN depicted in Fig. 4, we use four hidden layers with sizes $N_{\text{NLSS1}} = N_{\text{NLSS2}} = N_{\text{NLSS3}} = \frac{1}{2}N_{\text{IN}} + 15$ and $N_{\text{NLSS4}} = N_{\text{OUT}}$.

Second, for a given data set, we perform the cross validation to find the best models. We use the 6-zone office building example, which is shown in Fig. 6, to illustrate this. Before the cross validation, we have a high-variance problem using fixed portion of data to perform training and validation, in which the average and maximum validation error went up to 3.5% and 44%.



(a) (b)

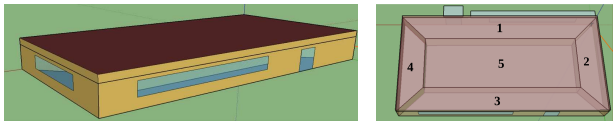
Fig. 6: The 6-zone office building (a) side view (b) top view

Cross validation is applied using non-linear state space RNN. Specifically, we group the 12 sequences of experiment data into 6 groups with two continuous sequences in each to perform a 6-fold cross validation. As shown in Table I, in each run of cross validation, we pick a single group to do validation and rest of groups to train the RNN. From this table, we obtain the best NLSS model when we pick sequence 11–12 as the validation data with average relative error 3.5% and maximum relative error 23%; and the best ELNN model when using sequence 7–8 as the validation data.

V. Numerical Results and Discussions

The proposed RNN based modeling method has been implemented in Python, using the popular PyBrain [32], NumPy [33], SciPy [20] and Matplotlib [18] for building the RNN, and for manipulating and visualizing the input data. We run EnergyPlus on three different buildings to get the referencing data which will later be compared with the outputs of our RNN models to observe errors.

In addition to the data center building with two zones in Fig. 1 and the office building with 6 zones in Fig. 6, we also have the third example: the office building with 5 zones shown in Fig. 7.



(a) (b)

Fig. 7: The 5-zone office building (a) side view (b) top view.

For the 2-zone and 5-zone building, we do two different simulations on each: one simulation considering only ambient temperature and HVAC (Heating, Ventilation, Air-Conditioning) factors, and the other simulation considering all factors like occupants, equipment factors, etc. Since the 6-zone building uses different air system other than HVAC, we only include the EnergyPlus simulation and RNN modeling considering all factors. As shown in Table II, we use one year's simulation data generated by EnergyPlus to train and validate the RNN models. In each case, input factors (such as ambient temperature, HVAC and equipment factor) and output (temperature of each zone) are simulated and sampled in 1-hour time steps by EnergyPlus. We group the data points by month to get 12 sequences for cross-validation based training and validation as mentioned earlier.

	#Input	#Output	#Samples	#Sequences
2-zone (all factors)	11	2	24×365	12
2-zone (HVAC only)	3	2	24×365	12
5-zone (all factors)	26	5	24×365	12
5-zone (HVAC only)	6	5	24×365	12
6-zone	55	6	24×365	12

TABLE II: Number of Inputs and outputs generated by EnergyPlus for RNN training and validation.

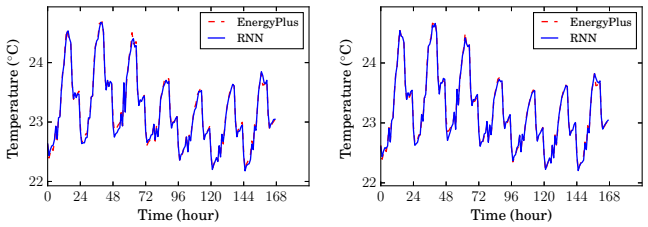
In our work, we use following the error notations. We use *Absolute error* to denote the value $|t_{\text{EP}} - t_{\text{RNN}}|$ and *Relative error* to denote the value $\frac{1}{t_{\text{EP}}}|t_{\text{EP}} - t_{\text{RNN}}|$, where t_{EP} and t_{RNN} are zone temperature generated by EnergyPlus and the proposed RNNs. We calculate the average *(avg)* and maximum *(max)* for each case for the two error terms mentioned above.

First, for the 2-zone building shown in Fig. 1, we train the two non-linear state space network (NLSS) and Elman's local-feedback neural network (ELNN) with sizes of layers configured according to the rules mentioned before. Among the 12 sequences of data in each month of a year, we use months 1–3, 5–7, 9–11 for training and months 4, 8, 12 for validation. A sample validation curve data of one week is shown in Fig. 8, where all input factors are considered.

As shown in Table III, average relative errors of NLSS and ELNN are 0.11% and 0.10% and maximum relative errors are 0.94% and 1.1% respectively. We also did experiment on the 2-zone building with all input factors forced to zero except HVAC. In this HVAC-only case, the average relative errors of NLSS and ELNN are 1.2% and 0.98% and maximum relative errors are 7.0% and 6.5%, shown in Fig. 9.

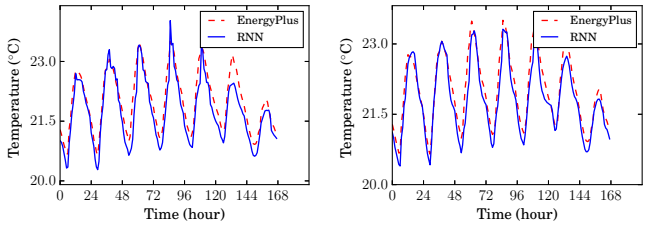
		2-zone (all)		2-zone (HVAC)		5-zone (all)		5-zone (HVAC)		6-zone	
		NLSS	ELNN	NLSS	ELNN	NLSS	ELNN	NLSS	ELNN	NLSS	ELNN
Training	Absolute error (avg) (°C)	0.025	0.022	0.20	0.19	0.24	0.19	0.050	0.14	0.36	0.31
	Absolute error (max) (°C)	0.30	0.30	1.4	2.1	4.9	4.4	0.84	1.8	3.1	5.4
	Relative error (avg) (%)	0.11	0.090	1.0	0.94	1.1	0.84	0.23	0.62	1.6	1.4
	Relative error (max) (%)	1.28	1.3	8.5	12	17	16	4.26	9.6	13	17
Validation	Absolute error (avg) (°C)	0.026	0.022	0.23	0.20	0.31	0.23	0.057	0.16	0.48	0.53
	Absolute error (max) (°C)	0.21	0.24	1.4	1.34	4.6	4.4	2.2	2.2	4.8	4.1
	Relative error (avg) (%)	0.11	0.10	1.2	0.98	1.4	1.0	0.26	0.71	2.2	2.2
	Relative error (max) (%)	0.94	1.1	7.0	6.5	20	16	9.6	12	23	16

TABLE III: Error statistics of the two RNN architectures (NLSS: non-linear state space RNN with global feedback; ELNN: Elman's network with local feedbacks).



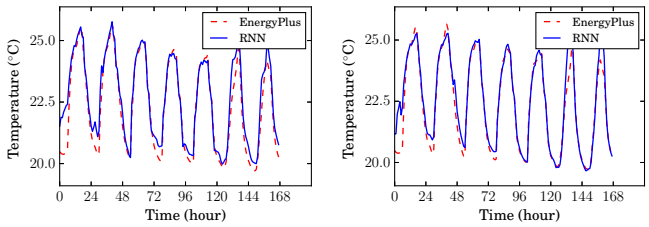
(a) Non-linear state space RNN (b) Elman's RNN

Fig. 8: Validation errors for 2-zone building model (all factors).



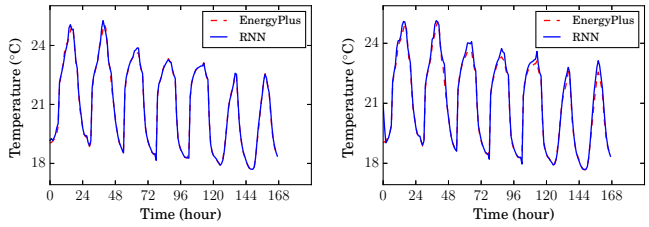
(a) Non-linear state space RNN (b) Elman's RNN

Fig. 9: Validation errors for 2-zone building model (HVAC only).



(a) Non-linear state space RNN (b) Elman's RNN

Fig. 10: Validation errors for 5-zone building model (all factors) validation.



(a) Non-linear state space RNN (b) Elman's RNN

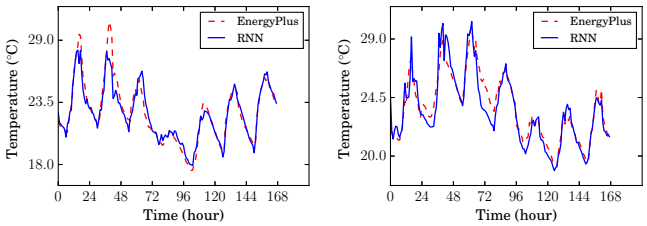
Fig. 11: Validation errors for 5-zone building model (HVAC only).

For the 5-zone building (floor plan is shown in Fig. 7) case, we use same data subset as in 2-zone to perform RNN training and validation. To overcome the overfitting problem and reduce validation error, we applied regularization on connection weights, which is implemented by adjusting the `weightdecay` parameter to limit the growth speed of connection weights in the training phase. As shown in Table III, in the case where all factors are considered, average relative errors of NLSS and ELNN are 1.4% and 1.0% and maximum relative errors are 20% and 16% respectively. In the HVAC-only case, the average relative errors of NLSS and ELNN are 0.26% and 0.71% and maximum relative errors are 9.6% and 12%. The representing curves sampled in one week are shown in Fig. 10 and Fig. 11.

For the 6-zone building shown in Fig. 6, we perform cross-validation on both NLSS and ELNN. Among the training/validation errors shown in Table I, the maximum relative errors in validation are 23% for NLSS and 16% for ELNN, reduced from 34% and 37% in the baseline configuration using a simple partitioning of training/validation data set.

As shown in Table III, the NLSS and ELNN using cross validation delivers relative average error of 2.2% and 2.2% respectively, the relative maximum error of 23% and 16% respectively. The representing curves for the 6-zone building sampled in one week are shown in Fig. 12a and Fig. 12b respectively.

In summary, the performance of the two RNN networks are quite similar especially in the 2-zone and 5-zone building examples. But for the 6-zone building example, which is more difficult with larger errors, we find that ELNN network performs better than NLSS network with maximum errors 16% against 23%. But both methods have 2.2% average errors.



(a) Non-linear state space RNN (b) Elman's RNN

Fig. 12: Validation errors for 6-zone building model.

VI. Conclusion

In this article, we have proposed a recurrent neutral network (RNN) based behavioral thermal modeling method for energy-efficient smart building control and management. The model is trained by using the data generated from realistic building simulation program, EnergyPlus, which can simulate some practical buildings such as office buildings and data centers. In this work, we have applied and studied two recurrent neural network architectures to build the non-linear compact thermal model for the building: one is non-linear state space RNN architecture (NLSS), which has global feedbacks, and the other one is Elman's RNN architecture (ELNN), which has local feedbacks in each layer. We derived a simple formula to calculate the RNN layer number, layer size to configure RNN architecture to avoid overfitting and underfitting problems. A cross-validation based training technique has been further applied to improve the accuracy. Experimental results from a case study of three buildings show that ELNN and NLSS can both build very accurate building thermal models for the 2-zone and 5-zone building cases: both of them have average errors from around 1% to 1.5% for the two buildings. And for the more complex 6-zone building case, ELNN outperforms NLSS with maximum errors 16% against 23%. But both methods have 2.2% average errors.

References

[1] Building Energy Data Book of DOE. [Online]. Available: http://buildingsdatabook.eren.doe.gov

[2] "Google SketchUp," http://www.sketchup.com.

[3] TRNSYS – A transient systems simulation program. [Online]. Available: http://www.trnsys.com/

[4] T. Bergman, A. Lavine, F. P. Incropera, and D. P. DeWitt, Fundamentals of Heat and Mass Transfer, 7th ed. New York: John Wiley & Sons, 2011.

[5] Y. Cao, R. Ding, and Q. Zhang, "A new nonlinear transient modelling technique for high-speed integrated circuit applications based on state-space dynamic neural network," in Microwave Symposium Digest, 2004 IEEE MTT-S International, vol. 3. IEEE, 2004, pp. 1553–1556.

[6] Y. Cao, R. Ding, and Q.-J. Zhang, "State-space dynamic neural network technique for high-speed IC applications: modeling and stability analysis," Microwave Theory and Techniques, IEEE Transactions on, vol. 54, no. 6, pp. 2398–2409, 2006.

[7] D. B. Crawley, L. K. Lawrie, F. C. Winkelmann, W. F. Buhl, Y. J. Huang, C. O. Pedersen, R. K. Strand, R. J. Liesen, D. E. Fisher, M. J. Witte, and J. Glazer, "EnergyPlus: creating a new-generation building energy simulation program," Energy and Buildings, vol. 33, no. 4, pp. 319–331, Apr. 2001.

[8] K. Deng, P. Barooah, P. G. Mehta, and S. P. Meyn, "Building Thermal Model Reduction via Aggregation of States," in American Control Conference, June 30-July 02, 2010, Baltimore, MD, Baltimore, MD, Jun. 2010, pp. 5118–5123.

[9] T. Eguia, S. X.-D. Tan, R. Shen, D. Li, E. H. Pacheco, M. Tirumala, and L. Wang, "General parameterized thermal modeling for high-performance microprocessor design," IEEE Trans. on Very Large Scale Integration (VLSI) Systems, 2011.

[10] B. Eisenhower and I. Mezic, "Extracting Dynamic Information from Whole-building Energy Models," in Proc. of Conference on Dynamics for Design (DFD 2012), Chicago, IL, Apr. 2012, pp. 3–10.

[11] J. L. Elman, "Finding structure in time," Cognitive science, vol. 14, no. 2, pp. 179–211, 1990.

[12] C. Goller and A. Kuchler, "Learning task-dependent distributed representations by backpropagation through structure," in Neural Networks, 1996., IEEE International Conference on, vol. 1. IEEE, 1996, pp. 347–352.

[13] S. Haykin and N. Network, "A comprehensive foundation," Neural Networks, vol. 2, no. 2004, 2004.

[14] M. T. Heath, Scientific Computing: An Introductory Survey. McGraw-Hill, 1997.

[15] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in Neural Networks, 1989. IJCNN., International Joint Conference on. IEEE, 1989, pp. 593–605.

[16] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," Neural networks, vol. 2, no. 5, pp. 359–366, 1989.

[17] Y. Hua and T. Sarkar, "Generalized pencil of function method for extracting poles of an em system from its transient responses," IEEE Trans. on Antennas and Propagation, vol. 37, no. 2, pp. 229–234, Feb. 1989.

[18] J. D. Hunter, "Matplotlib: A 2d graphics environment," Computing in science and engineering, vol. 9, no. 3, pp. 90–95, 2007.

[19] C. Igel and M. Hüsken, "Empirical evaluation of the improved rprop learning algorithms," Neurocomputing, vol. 50, pp. 105–123, 2003.

[20] E. Jones, T. Oliphant, P. Peterson et al., "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed 2015-07-23]. [Online]. Available: http://www.scipy.org/

[21] D. Kim and J. E. Braun, "Reduced-Order Building Modeling For Application to Model-Based Predictive Control," in 5th National Conference of Innational Building Performance Simulation Association (IBPSA-USA), Aug. 2012, pp. 554–561.

[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, May 2015.

[23] I. Leontaritis and S. A. Billings, "Input-output parametric models for non-linear systems part i: deterministic non-linear systems," International journal of control, vol. 41, no. 2, pp. 303–328, 1985.

[24] D. Li, S. X.-D. Tan, E. H. Pacheco, and M. Tirumala, "Parameterized architecture-level dynamic thermal models for multicore microprocessors," ACM Trans. Des. Autom. Electron. Syst., vol. 15, no. 2, pp. 1–22, 2010.

[25] Z. Liu, S. X.-D. Tan, H. Wang, A. Gupta, and S. Swarup, "Compact nonlinear thermal modeling of packaged integrated systems," in Proc. Asia South Pacific Design Automation Conf. (ASPDAC), January 2013, pp. 157–162.

[26] A. Malkawi and G. Augenbroe, Eds., Advanced Building Simulation. Spon Press, Jun. 2004.

[27] M. Massoumy, Q. Zhu, C. Li, F. Meggers, and A. Sangiovanni-Vincentelli, "Co-design of Control Algorithm and Embedded Platform for Building HVAC Systems ," in International Conference on Cyber Physical Systems (ICCPS), Philadephia, PA, Apr. 2013, pp. 61–70.

[28] P. V. Overschee and B. D. Moor, "N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems," Automatica, vol. 30, no. 1, pp. 75–93, 1994.

[29] G. V. Puskorius, L. Feldkamp, L. Davis et al., "Dynamic neural network methods applied to on-vehicle idle speed control," Proceedings of the IEEE, vol. 84, no. 10, pp. 1407–1420, 1996.

[30] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in Neural Networks, 1993., IEEE International Conference on. IEEE, 1993, pp. 586–591.

[31] R. Rojas, Neural networks: a systematic introduction. Springer Science & Business Media, 2013.

[32] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber, "PyBrain," Journal of Machine Learning Research, vol. 11, pp. 743–746, 2010.

[33] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," Computing in Science & Engineering, vol. 13, no. 2, pp. 22–30, 2011.

[34] P. J. Werbos, "Backpropagation through time: what it does and how to do it," Proceedings of the IEEE, vol. 78, no. 10, pp. 1550–1560, 1990.