Final Project: Deep Learning for Commercial Building System Identification

*CSCI575 Advanced Machine Learning

1st Dylan Wald
Department of Electrical Engineering
Colorado School of Mines
Golden, US
dylanwald@mines.edu

***Abstract*—Commercial buildings contribute to a large portion of CO2 emissions in the United States today, much of which is attributed to their high energy consumption. Many strategies such as Model Predictive Control (MPC) have been investigated to control a building's HVAC system and improve efficiency of the building. However, MPC requires a model accurate enough to apply controls, but not overly complex. One strategy is to use a gray-box model, representing a building as an equivalent electrical circuit. Another strategy is to use a black-box model, which uses only data to learn a model. By combining the two, one can leverage the physical structure of a gray-box model using deep learning (DL) techniques. In this work, three DL techniques: a recurrent neural network, a gated recurrent unit, and a novel, neural state space model are implemented using PyTorch to learn the structure of a reduced order building model and predict the building temperature. Results indicate that the three DL models each perform very well in training and also generalize well to the given data.**

***Index Terms*—buildings, deep learning, RNN, GRU, constrained learning**

I. Introduction

In 2021, buildings represented 27% of the CO2 emissions from the energy sector in the United States [1] and currently, their heating, ventilation and air conditioning (HVAC) systems represent a large portion of their end-use energy consumption [2]. However, these HVAC systems also provide a clear opportunity to improve efficiency of a building through various advanced control techniques.

A popular advanced control technique for building HVAC control is Model Predictive Control (MPC) [3]. In MPC, one must have an accurate model of the system they are trying to control, in this case a commercial building. This may be a difficult task as a large building can be a very complex, coupled system. Furthermore, if a complex model can be obtained, it may not be feasible to implement in a control scheme due to computational complexity. These *purely physics based* models are typically referred to as white-box models [4]–[6].

On the other end, a black-box model referrers to a model that is *purely data-driven*, and uses methods like regression as the underlying model [5], [6]. Here, very little knowledge about the model physics is required. However, in some cases a very large amount of data is needed to develop these models, which may be difficult to obtain. Additionally, these models may not generalize well to data they are not trained on.

In a gray-box modelling approach, a simplified model structure based on physics is generated, but data is used to identify parameters of the model [6]. Hence, the complexities of a white-box model are simplified and the data dimensionality and model generalization problems of a black-box model are mitigated. An example of a gray-box model for a building is a 1-resistor, 1-capacitor (1R1C) equivalent circuit model [4], see Fig. 1. Some issues with this type of gray-box modelling is the fact that 1) the models are very simplified and may miss complexities in the dynamics and 2) they still require the use on complicated estimation techniques, such as an extended Kalman Filter [7], to identify the model parameters.

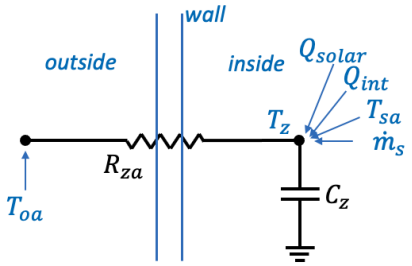

Fig. 1. Equivalent 1R1C model of a building. Figure adapted from [4].

To address point 2) above, researchers in [5], [8], [9] developed a hybrid framework to leverage deep learning (DL) methods to identify gray-box models for commercial buildings. Their work investigates recurrent neural networks (RNNs), gated recurrent units (GRUs), and a novel neural state space model (NSSM), how they are related to traditional dynamical systems, and how they can be used to more efficiently and accurately build gray-box models for buildings. Therefore, in this report, the methods introduced in [5], [8], [9] are recreated and used to identify a 1R1C model for a commercial building using a small data set. Therefore, the *goal* is to build DL models that resemble the 1R1C model such that the building zone temperature is predicted as accurately as possible. Each

method is implemented using the open source framework PyTorch [10]. The results from each method are then compared using different input data normalization techniques.

II. Methods

In order to recreate the methods described in [5], [8], [9], the commercial building dynamical system investigated is first introduced. The succeeding sections then describe in detail the DL models and methods used in this work. Last, the data processing is described.

A. 1R1C Dynamic Model of a Building

From the 1R1C equivalent circuit model shown in Fig. 1, a simple first-order differential equation

$$\begin{aligned}\dot{T}_z = &\frac{1}{R_{za}C_z}(T_{oa} - T_z) \\ &+ \frac{1}{C_z}(Q_{solar} + Q_{int} + \dot{m}_s + T_{sa})\end{aligned} \tag{1}$$

is identified where T_z is the indoor (zone) temperature of the building, $\dot{m}_s$ is the mass flow rate of the HVAC system, T_{sa} is the supply air temperature of the HVAC system, T_{oa} is the outdoor air temperature, Q_{solar} is the solar heat gain through the windows of the building, and Q_{int} is the internal load of the building. As (1) is a first-order differential equation, it can then be put into state space (SS) form. Let T_z be the system **state** (x), $\dot{m}_s$ and T_{sa} be the system **inputs** (u), T_{oa}, Q_{solar}, and Q_{int} be the system **disturbances** (d), and T_z be the system **output** (y). Then, (1) can be represented by the following SS model

$$\begin{aligned}x_{k+1} &= Ax_k + Bu_k + Ed_k \\ y_k &= Cx_k\end{aligned} \tag{2}$$

where A, B, and E are functions of the resistance and capacitance values and are unique to a specific building. Note that, in this model, $C = 1$ and thus $y_k = x_k$ where k is the discrete time step.

B. Recurrent Neural Network

A typical RNN can be represented by the system of equations shown in (3) below

$$\begin{aligned}h_t &= ReLU(x_t W_{ih}^T + h_{t-1} W_{hh}^T) \\ y_t &= h_t\end{aligned} \tag{3}$$

where h_t is the hidden state, x_t is the input to the RNN, y_t is the output of the RNN, W_{ih}^T and W_{hh}^T are weight matrices, and $ReLU$ is the rectified linear activation unit (ReLU) activation function [10], [8] . Immediately, it is evident that there are clear similarities between the RNN structure (3) and the SS model (2). To see these similarities, let $h_t = x_k$, $W_{hh}^T = A$, $x_t = \begin{bmatrix} u_k & d_k \end{bmatrix}^T$, and $W_{ih}^T = \begin{bmatrix} B & E \end{bmatrix}$. The system of equations (3) then becomes

$$\begin{aligned}x_k &= ReLU(x_{k-1}A + \begin{bmatrix} B & E \end{bmatrix} \begin{bmatrix} u_k & d_k \end{bmatrix}^T) \\ y_k &= x_k\end{aligned} \tag{4}$$

The only difference between (2) and (4) is the nonlinear ReLU activation function. Therefore, in a sense, as weight matrices W_{ih}^T and W_{hh}^T are learned, the RNN is essentially learning the 1R1C model for the building. Figure 2 shows a block diagram of (3) and how it relates to the SS model.

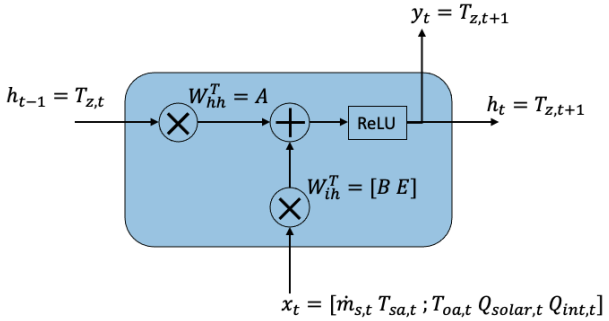

Fig. 2. Individual RNN cell. Figure adapted from [11].

Note that while Fig. 2 displays a single RNN cell, in practice multiple RNN can be stacked to create a "stacked RNN". A stacked RNN is equivalent to adding layers to a neural network. The more layers in a network, the more complex predictions can be made, i.e., the model is deeper [12]. Additionally, a stacked RNN very naturally resembles the propagation of a dynamical system, such as (2), through time. Figure 3 displays a stacked RNN with respect to the system in this work.

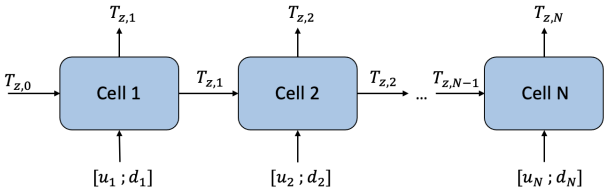

Fig. 3. Stacked RNN framework used in this work.

C. Neural State Space Model

The neural state space model (NSSM) is a novel approach introduced in [8]. This method decouples the different transformations between the states x, inputs u, and disturbances d by directly learning the A, B, and E matrices from (2). The motivation behind this decoupling is to be able to preserve known physical characteristics of the system. The system of equations for this NSSM is shown below in (5)

$$\begin{aligned}\tilde{x}_t &= ReLU(\tilde{W}_A^T h_{t-1}) \\ \tilde{u}_t &= ReLU(W_B^T u_t) \\ \tilde{d}_t &= ReLU(W_E^T d_t) \\ h_t &= \tilde{x}_t + \tilde{u}_t + \tilde{d}_t \\ y_t &= h_t\end{aligned} \tag{5}$$

where

$$\tilde{W}_{A_{i,j}}^T = \frac{exp(A_{i,j})M_{i,j}}{\sum_{k=1}^{n_x} exp(A_{i,j})} \tag{6}$$

and where $M = \mathbf{1} - \epsilon\sigma(M')$, $\epsilon \in [0, 1)$, and $\mathbf{1}$ is a matrix of ones. Equation (6) is a parametrization of the transition matrix A and essentially constrains the eigenvalues of the $\tilde{W}^T_{A_{i,j}}$ to be less than or equal to 1 in the learning process. This forces $\tilde{W}^T_{A_{i,j}}$ to be stable, which is an inherent property of a physical system. According to the authors of [8], by constraining $\tilde{W}^T_{A_{i,j}}$: 1) the stability of the learned dynamics is guaranteed, 2) can prevent exploding or vanishing gradients, and 3) is a clear path to modelling a wide range of physical systems. A block diagram describing the system of equations in (5) is shown below in Fig. 4. A NSSM cell can also be stacked similar to an RNN.

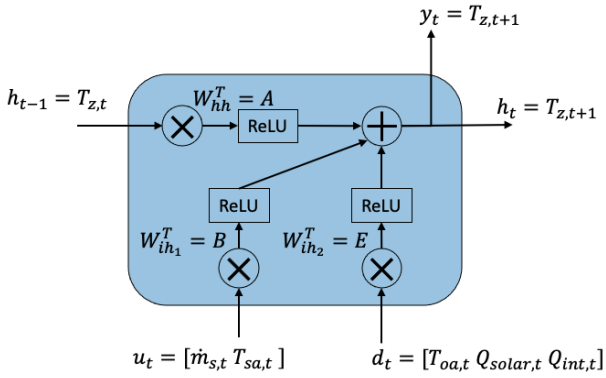

Fig. 4. Individual NSSM cell.

D. Gated Recurrent Unit

A typical RNN such as the one described in Section II-B can suffer from a vanishing or exploding gradient when learning long-term relationships [13]. Some strategies, such as the GRU, have been investigated to mitigate this issue. A GRU is a type of RNN that uses a more complex activation function, or a gated unit. This gated unit contains a "forget" gate and a "update gate" [13]. A typical GRU is represented by the system of equations in (7) [10].

The forget gate, represented by r_t in (7), determines whether or not a previously computed state should be included in the next state's computation. The update gate, represented by z_t in (7), determines how much information form the past state and new inputs is used to compute the next state. The next state is then a linear interpolation between h_{t-1} and $\tilde{h}_t$.

$$\begin{aligned} r_t &= \sigma(W_r^T[h_{t-1}, x_t]) \\ \tilde{h}_t &= tanh(W_h^T[r_t \odot h_{t-1}, x_t]) \\ z_t &= \sigma(W_z^T[h_{t-1}, x_t]) \\ h_t &= (1 - z_t) \odot \tilde{h}_t + z_t \odot h_{t-1} \\ y_t &= h_t \end{aligned} \tag{7}$$

Like in Section II-B, the hidden state corresponds to the system state ($h_t = x_k$) and the cell input corresponds to the system inputs and disturbances ($x_t = \begin{bmatrix} u_k & d_k \end{bmatrix}^T$). In the GRU, the weight matrices do not correspond to (2) as naturally as the RNN and thus the GRU is used in this work to compare the other two methods. A block diagram describing a GRU cell is shown below in Fig. 5. A GRU cell can also be stacked similar to an RNN.

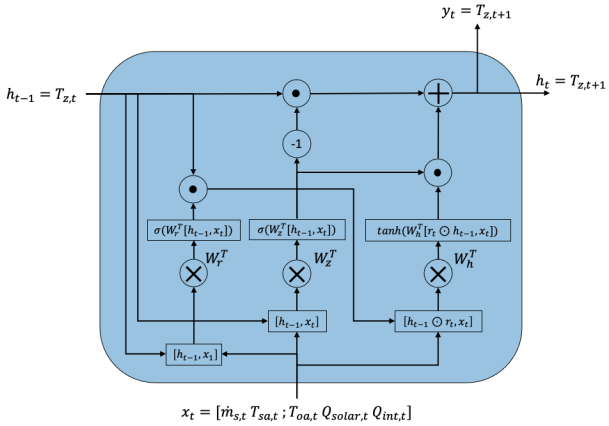

Fig. 5. Individual GRU cell. Figure adapted from [11].

E. Data Preparation

Before the DL methodologies are applied, data is obtained and put into the correct form. In this work, time series data from 20 days with 5 minute resolution was obtained by running an EnergyPlus [14] simulation. EnergyPlus is a high fidelity building simulation tool that can generate high accuracy outputs from a given building input file and weather data. An EnergyPlus building input file, `.idf` format, contains all known details of a building. In this work, an `.idf` file generated in [15], as well as weather data from Golden, CO, is used in an EnergyPlus simulation to produce 20 days' worth of T_z, $\dot{m}_s$, T_{sa}, T_{oa}, Q_{solar}, and Q_{int} data with a 5 minute resolution. Note that the building used to generate the data is an office building with multiple stories and multiple zones per story. However, in this work, only one floor is analyzed where data from each zone is averaged and lumped into one zone.

Upon analyzing the data, it was found that the time series data was drastically different on weekend days than weekdays. For this reason, weekend data was removed from the data set, resulting in 14 total days. Figure 6 below displays the first 10 days of the data set described above. The top plot shows the inputs, the middle plot shows the disturbances, and the bottom plot shows the output (truth) value.

Since the models in Sections II-B - II-D are implemented in PyTorch, the data set must be in the correct format. First, the data set is split into training and testing segments. Roughly using a 70% - 30% split, the training data set is composed of the first 10 days while the test data set is composed of the final 4 days. Note that this data is sequential and thus is not randomized. For sequential learning methods such as RNNs and GRUs, PyTorch requires input data to be a 3-dimensional tensor where the first element is the batch size (or sequence length), the second element is the number of samples, and the third element is the number of features. In this work, each data set has 5 features and each model is

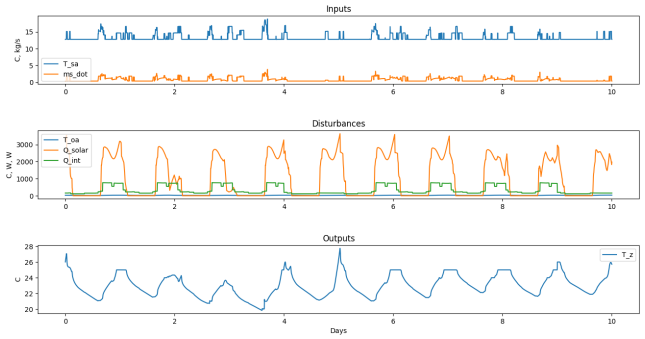

Fig. 6. 10 days of times series data obtained and cleaned from an EnergyPlus simulation. **Top**: input, **middle**: disturbance, and **Bottom**: output data.

trained in 16 element chunks, i.e., a batch size of 16. Thus, the training and testing sets have shapes `(16, 180, 5)` and `(16, 72, 5)`, respectively.

III. Results

To analyze the performance of each DL method presented in Sections II-B - II-D, three different simulations are ran for each model. The three different simulations, Cases 1-3, each use a different type of normalization on the input data and are described in Table I. For Case 1, each input value is normalized via. min-max normalization, individually. In Case 2, all inputs are normalized via. min-max normalization based on the largest input value. In Case 3, there is no normalization on the input data. In each case, the performance is analyzed for the training set and testing set using the learning model and its state space equivalent. The performances are quantitatively compared using the Mean Square Error (MSE) between the predicted indoor temperature and actual indoor temperature and qualitatively compared by plotting the predictions. The parameters used in the training of each model are shown in Table IV of the Appendix.

TABLE I
CASES INVESTIGATED FOR ALL THREE METHODS AND THEIR STATE SPACE EQUIVALENTS.

	Individual Normalization	Max Element Normalization	No Normalization
Case 1	X	-	-
Case 2	-	X	-
Case 3	-	-	X

A. Training Performance

The first step is to train each model. Using theory described in Sections II-A - II-D, each model is built using the PyTorch framework in Python coding language. Then, using the training data prepared in Section II-E and the parameters described in Table IV, each model is trained for each of the three cases described in Table I. Each performance is recorded and tabulated in Table II.

It can be seen that each DL model performs best in Case 1. However, this effect is most evident in the case of the RNN and the NSSM. The GRU seems mostly unaffected by the different normalization schemes. For this reason, the GRU has the best performance on average across all cases. However, while one would expect the NSSM model to perform the best, the best overall performance is from the RNN (in Case 1). This may be due to the fact that each batch size is only 16 time steps. With a larger batch size, the RNN may start to suffer from the exploding or vanishing gradient problem. Additionally, the building model is quite simplified. If the zones were not averaged across an entire story, the building model would be much more complex and thus each DL model would be more complex. It may be the case that other methods such as the NSSM and GRU start to outperform the RNN in with a more complex model.

TABLE II
MSE FOR EACH CASE AND EACH MODEL EVALUATED ON THE TRAINING DATA.

	RNN	NSSM	GRU	SS from RNN	SS from NSSM
Case 1	0.0496	0.0866	0.0933	7.45	12.4
Case 2	0.0521	0.0700	0.0920	1.36	3.39
Case 3	1.04	0.144	0.0949	1.026	0.527

Oppositely, the SS equivalent models tend to perform best in Case 3, i.e., no normalization. This may be because, with no normalization, the inputs and disturbances have much different magnitudes, creating more unique scenarios. If nonlinearities are indeed present, more unique scenarios should allow a linear model, such as the SS models, to generalize better. On average, the RNN SS equivalent model performs best, but the NSSM produces the lowest overall MSE between the two.

Overall, the DL models have a much better performance then their equivalent SS models. As briefly mentioned in the previous paragraph, this is most likely due to the fact that the DL models can handle nonlineariteies that may arise between the state (zone temperature) and the disturbances and inputs acting on the building. The equivalent SS models are restricted to linear dynamics and as a result may miss crucial nonlinearities in the building dynamics. The actual SS equivalent model values are shown in Table V in the Appendix.

To qualitatively analyze the training results, the predicted and true building temperatures are plotted across all training days. Due to the lack of space in the report, only the Case 2 results are shown, see Figs. 7 and 8. Figure 7 displays the predictions of each DL model while Fig. 8 displays the predictions of the equivalent SS models.

It can be seen from Figs. 7 and 8 that, as indicated by the results in Table II, the DL models clearly outperform the equivalent SS models. There is clearly a much larger deviation between the predicted and true T_z values in Fig. 8. Within Fig. 8, the RNN slightly outperforms the NSSM. However, it is much more difficult to see the difference between each model's performance in Fig. 7. Therefore, overall the results indicate that the DL models all perform extremely well on the training data.

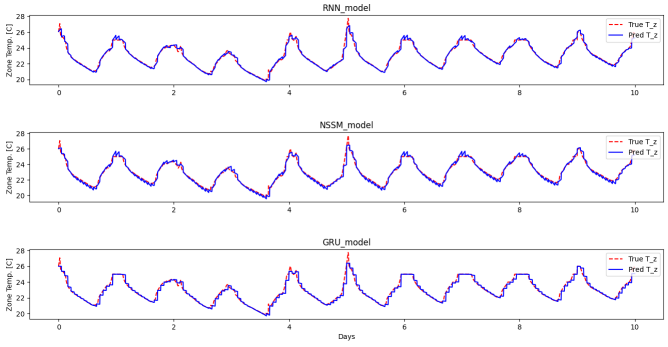

Fig. 7. Predicted vs. true indoor temperature T_z for **Top**: the RNN model, **middle**: the NSSM model, and **bottom**: the GRU model in Case 2 using training data.

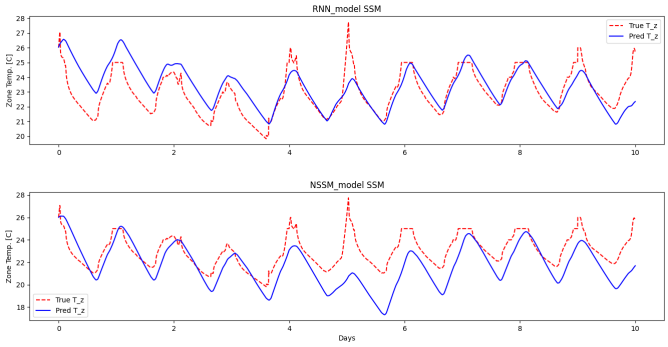

Fig. 8. Predicted vs. true indoor temperature T_z for **Top**: the RNN-equivalent state space model and **bottom**: the NSSM-equivalent state space model in Case 2 using training data.

B. Testing Performance

With training complete, the next step is to evaluate each model's performance on the test set. This will indicate how generalizable the models are. Using the models trained in Section 7, each model for each normalization case is analyzed on the test data. The results are tabulated in Table III.

TABLE III
MSE FOR EACH CASE AND EACH MODEL EVALUATED ON THE TEST DATA.

	RNN	**NSSM**	**GRU**	**SS from RNN**	**SS from NSSM**
Case 1	0.0362	0.0924	0.0704	1.46	12.3
Case 2	0.0420	0.0653	0.0686	1.09	4.33
Case 3	1.36	0.196	0.0719	2.10	1.42

From Table III, it can be seen that the trend in each model's performance resembles the results in Table II. The DL models tend to perform better *with* normalization while the SS equivalents tend to perform better *without* normalization. Also, like in Table II, the GRU seems to be unaffected by the normalization cases.

To to analyze the DL approaches, let us only consider Case 1. From Tables II and III, it can be seen that the RNN and GRU models actually produce better results on the test data. In fact, the RNN returns the smallest overall MSE for any case on test data for Case 1. While the NSSM model performs only slightly worse, the performance is very similar. Based on these results, it can be said that the DL methods are *not* over fitting the training data and that they generalize well in modelling the commercial building.

Now, to analyze the equivalent SS approaches, let us only consider Case 3. From Tables II and III, it can be seen that both the RNN and NSSM equivalent SS models perform slightly worse on the test data. However, while slightly worse, their performances are still very similar to their respective training data performances. In the end, the performance of the equivalent SS models cannot be deemed over or under fitted since they are purely products of their corresponding DL models.

Next, the model performances are qualitatively analyzed. The predicted and true building temperatures are plotted across all test days for Case 2 only and are shown in Figs. 9 and 10. Figure 9 displays the predictions of each DL model while Fig. 10 displays the predictions of the equivalent SS models.

Similar to results in Figs. 7 and 8, the DL models clearly outperform the equivalent SS models on the test data. This is expected from the results in Table III. Also, similarly, the the RNN equivalent model clearly outperforms the NSSM SS equivalent model. There is no visible difference between the DL models, which all predict T_z extremely well.

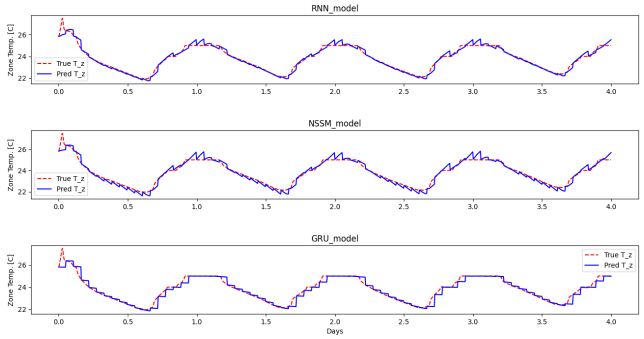

Fig. 9. Predicted vs. true indoor temperature T_z for **Top**: the RNN model, **middle**: the NSSM model, and **bottom**: the GRU model in Case 2 using test data.

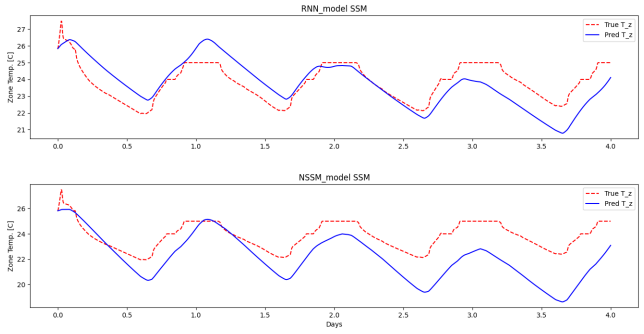

Fig. 10. Predicted vs. true indoor temperature T_z for **Top**: the RNN-equivalent state space model and **bottom**: the NSSM-equivalent state space model in Case 2 using test data.

IV. Conclusions

In this work, a novel hybrid approach developed by authors in [8] is explored. This hybrid approach makes use of physical structure of a gray-box modelling approach and the efficient learning capabilities of a black-box approach to effectively develop controls oriented models for commercial buildings. This work reproduces this hybrid method, i.e. the NSSM, and leverages its capabilities to develop a 1R1C equivalent circuit model of a commercial building. The goal of this model is to accurately predict the zone temperature of the building given inputs and disturbances. Implemented in PyTorch, the NSSM is supplemented by two additional DL methods, an RNN and a GRU, to compare performances. Additionally, the learned weight matrices from the NSSM and the RNN methods are used to develop and test equivalent SS models of the building.

After training each model using 10 days of data and three different normalization techniques, it was found that the DL models perform better when the inputs to the model are normalized while the equivalent SS models perform better with no normalization. Although, in either case, the DL models drastically outperform the equivalent SS models. This may be due to the SS models' limitations in terms of capturing model liniearities. Overall, it was found that the RNN had the best performance of all three models. However, this work uses a small batch size and an extremely simplified model, and thus results may differ with more complex scenarios. While it was found that all three DL models generalize well to the data used in this experiment, in future work should investigate including weekends to see how well the models generalize to drastically different inputs and disturbances such as different seasons and extreme weather events.

Appendix

TABLE IV
Learning Parameters used for the Training of Each Model.

	RNN	**NSSM**	**GRU**
Learning Rate	0.0005	0.0005	0.0006
Epochs	15000	15000	25000
Solver	AdamW	AdamW	AdamW
Cost Function	MSE	MSE	MSE

TABLE V
SS model matrices extracted from DL methods.

RNN			
	A	**B**	**E**
Case 1	$\begin{bmatrix}0.999\end{bmatrix}$	$\begin{bmatrix}0.0195\\-0.0102\end{bmatrix}$	$\begin{bmatrix}-0.016\\0.0829\\0.0161\end{bmatrix}$
Case 2	$\begin{bmatrix}0.996\end{bmatrix}$	$\begin{bmatrix}2.88\\1.16\end{bmatrix}$	$\begin{bmatrix}-0.997\\0.0651\\0.743\end{bmatrix}$
Case 3	$\begin{bmatrix}0.0322\end{bmatrix}$	$\begin{bmatrix}1.41\\-2.50\end{bmatrix}$	$\begin{bmatrix}0.209\\-8.42e-7\\0.00117\end{bmatrix}$
NSSM			
Case 1	$\begin{bmatrix}0.998\end{bmatrix}$	$\begin{bmatrix}0.04\\0.0055\end{bmatrix}$	$\begin{bmatrix}0.043\\0.058\\0.018\end{bmatrix}$
Case 2	$\begin{bmatrix}0.996\end{bmatrix}$	$\begin{bmatrix}4.5\\5.02\end{bmatrix}$	$\begin{bmatrix}4.83\\0.0471\\0.18\end{bmatrix}$
Case 3	$\begin{bmatrix}0.943\end{bmatrix}$	$\begin{bmatrix}2.10\\1.42\end{bmatrix}$	$\begin{bmatrix}0.0102\\2.28e-5\\8.2e-5\end{bmatrix}$

References

[1] C. Delmastro, "Buildings," September 2022. License: CC BY 4.0.

[2] "2018 Commercial Buildings Energy Consumption Survey, Consumption and Expenditures Highlights," December 2022.

[3] A. Afram and F. Janabi-Sharifi, "Theory and applications of HVAC control systems – A review of model predictive control (MPC)," *Building and Environment*, vol. 72, pp. 343–355, Feb. 2014.

[4] H. Harb, N. Boyanov, L. Hernandez, R. Streblow, and D. Müller, "Development and validation of grey-box models for forecasting the thermal response of occupied buildings," *Energy and Buildings*, vol. 117, pp. 199–207, Apr. 2016.

[5] J. Drgoňa, A. R. Tuor, V. Chandan, and D. L. Vrabie, "Physics-constrained deep learning of multi-zone building thermal dynamics," *Energy and Buildings*, vol. 243, p. 110992, 2021.

[6] A. Afram and F. Janabi-Sharifi, "Review of modeling methods for hvac systems," *Applied thermal engineering*, vol. 67, no. 1-2, pp. 507–519, 2014.

[7] R. Chintala, J. Winkler, and X. Jin, "Automated fault detection of residential air-conditioning systems using thermostat drive cycles," *Energy and Buildings*, vol. 236, p. 110691, 2021.

[8] J. Drgona, A. Tuor, and D. Vrabie, "Constrained physics-informed deep learning for stable system identification and control of unknown linear systems," *arXiv preprint arXiv:2004.11184*, 2020.

[9] J. Drgona, A. Tuor, and D. Vrabie, "Learning constrained adaptive differentiable predictive control policies with guarantees," 2022.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[11] A. dprogrammer, "RNN, LSTM & GRU," Apr. 2019.

[12] J. Brownlee, "Stacked Long Short-Term Memory Networks," Aug. 2017.

[13] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[14] M. Wetter, "Energyplus version 7," tech. rep., Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2011.

[15] J. Wang, R. El Kontar, X. Jin, and J. King, "Electrifying high-efficiency future communities: Impact on energy, emissions, and grid," *Advances in Applied Energy*, vol. 6, p. 100095, 2022.