Applied Energy 340 (2023) 121071

Contents lists available at ScienceDirect

Applied Energy

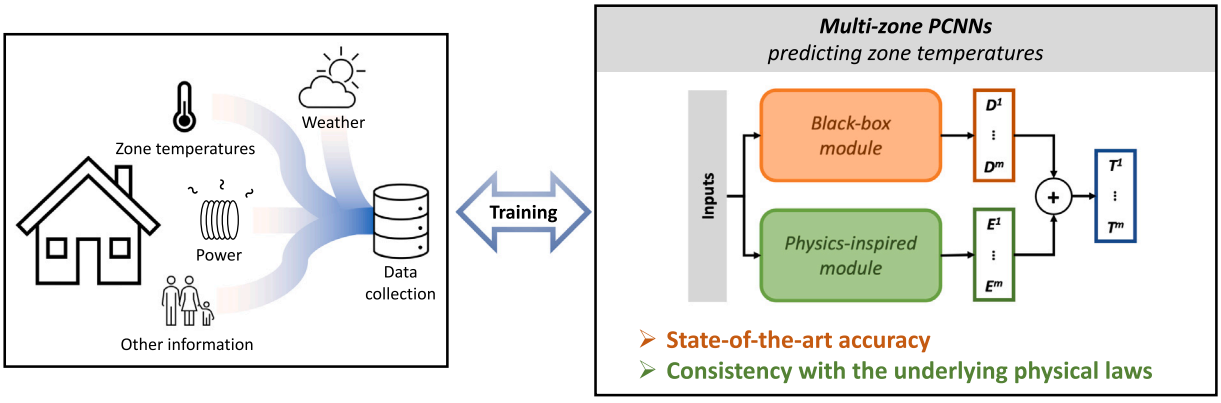
journal homepage: www.elsevier.com/locate/apenergy

Towards scalable physically consistent neural networks: An application to data-driven multi-zone thermal building models

L. Di Natale [a,b,*], B. Svetozarevic [a], P. Heer [a], C.N. Jones [b]

[a] Urban Energy Systems Laboratory, Swiss Federal Laboratories for Materials Science and Technology (Empa), 8600 Dübendorf, Switzerland
[b] Laboratoire d'Automatique, EPFL, 1015 Lausanne, Switzerland

GRAPHICAL ABSTRACT



ARTICLE INFO

Dataset link: https://gitlab.nccr-automation.ch/loris.dinatale/multi-zone-pcnns

Keywords:
Neural Network
Physical consistency
Building modeling
Deep learning
Physics-inspired

ABSTRACT

With more and more data being collected, data-driven modeling methods have been gaining in popularity in recent years. While physically sound, classical gray-box models are often cumbersome to identify and scale, and their accuracy might be hindered by their limited expressiveness. On the other hand, classical black-box methods, typically relying on Neural Networks (NNs) nowadays, often achieve impressive performance, even at scale, by deriving statistical patterns from data. However, they remain completely oblivious to the underlying physical laws, which may lead to potentially catastrophic failures if decisions for real-world physical systems are based on them. Physically Consistent Neural Networks (PCNNs) were recently developed to address these aforementioned issues, ensuring physical consistency while still leveraging NNs to attain state-of-the-art accuracy, and applied to zone temperature modeling.

In this work, we scale PCNNs to model the temperature dynamics of buildings with several connected thermal zones and propose a thorough comparison with classical gray-box and black-box methods. More precisely, we design three distinct PCNN extensions with different levels of information sharing between the modeled zones, thereby exemplifying the modularity and flexibility of the architecture, and formally prove their physical consistency. In the presented case study, PCNNs are shown to achieve state-of-the-art accuracy, even outperforming classical NN-based models despite their constrained structure. Our investigations furthermore provide a clear illustration of NNs achieving seemingly good performance while remaining completely physics-agnostic, which can be misleading in practice. While this performance comes at the cost of computational complexity, PCNNs on the other hand show accuracy improvements of 17–35% compared to all other physically consistent methods, paving the way for scalable physically consistent models with state-of-the-art performance.

* Corresponding author at: Urban Energy Systems Laboratory, Swiss Federal Laboratories for Materials Science and Technology (Empa), 8600 Dübendorf, Switzerland.
E-mail address: loris.dinatale@empa.ch (L. Di Natale).

https://doi.org/10.1016/j.apenergy.2023.121071
Received 23 December 2022; Received in revised form 25 March 2023; Accepted 31 March 2023
Available online 5 April 2023

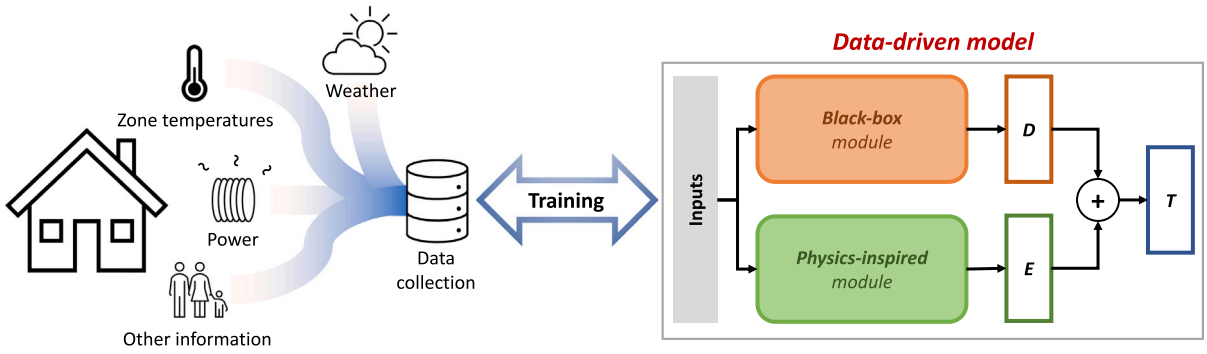


Fig. 1. General pipeline of data-driven building thermal modeling frameworks.

1. Introduction

Under the pressing issue of climate change, there is a worldwide effort to decrease our global energy consumption. Being responsible for a large part of the final energy consumption and greenhouse gas emissions [1], buildings are a primary target in that trend, with space heating and cooling being the main identified culprits [2]. Interestingly, we can intervene at any stage of the life of a building to decrease its energy intensity, either designing and constructing more efficient new structures [3], retrofitting old edifices [4], or designing smart controllers to minimize the energy consumption of the current building stock [5]. However, while decreasing the energy consumption of buildings is the main goal of advanced control algorithms, this cannot be done at the expense of the comfort of the inhabitants, who require the temperature to stay within a comfortable range [6]. This calls for accurate building temperature models to close the sim2real gap of advanced control algorithms [7,8]. Indeed, Model Predictive Control (MPC) uses a model to predict the impact of possible power input sequences and choose the optimal one [9], for example, and Reinforcement Learning (RL) control policies have to be trained in simulation prior to their deployment [6,10].

1.1. Towards data-driven methods

Since the evolution of the temperature in a thermal zone is governed by the laws of thermodynamics, the most natural way to model it is to write down the corresponding Ordinary Differential Equations (ODEs) and then use custom solvers or discretization schemes to propagate them through time, such as in [11,12]. To alleviate the engineering burden of constructing the ODEs describing building temperature dynamics, allow more complex structures to be modeled, and accelerate the entire pipeline, custom modeling tools are often used in practice, such as EnergyPlus, Modelica, TRNSYS, or IDA ICE [13–15]. Such detailed simulation tools however still require expert knowledge and access to many design parameters that are often not directly available [16], which makes them infamously hard to calibrate [17]. Moreover, solving the complex underlying ODEs to simulate each time step can entail a significant computational burden at runtime [18].

In recent years, owing to the growing amount of data collected in buildings, researchers started to leverage data-driven methods, bypassing the cumbersome procedures and expert knowledge required to set up classical physics-based models [19]. This gave rise to so-called gray-box or black-box frameworks, both of which use historical data for calibration or training purposes, as pictured in Fig. 1.

1.2. Gray- and black-box models

When a control-oriented thermal building model is designed, typically for MPC, data is in most cases used to identify the parameters of a simplified physics-based model [20], usually a low-order Resistance-Capacitance (RC) model, such as in [16,21–24]. These models are particularly popular due to their ease of implementation, interpretability, close ties to the underlying physics, and because they often give rise to linear dynamics. The latter characteristic is indeed particularly desirable in MPC applications since an appropriate choice of objective function then renders the optimization problem to solve at each time step convex and hence tractable. Nonetheless, the parameter identification procedure of RC models is generally nontrivial and sensitive to the data quality [25,26], which partially explains why low-order models often perform better than complex ones [27,28]. Alternatively or additionally, when data is available, the residual errors of an often simplified model can be fit to improve its performance, such as in [29].

While the aforementioned gray-box methods only require limited domain knowledge since simplified ODEs are used, which ODEs to choose is not always clear [30] and wrong choices might hinder the performance of the identified model. To avoid this pitfall, when enough data is available, fully data-driven black-box models might be used, such as AutoRegressive models with eXogenous inputs (ARX) or Neural Networks (NNs). Indeed, they do not rely on any prior knowledge of the system to model since they directly extract patterns from data to explain and predict the behavior of the system. Consequently, black-box methods are often easier and faster to deploy, more flexible, and thus often more scalable than their gray-box counterparts [31,32]. Furthermore, since they do not have to follow a predefined underlying architecture, black-box methods are generally more expressive, being capable of capturing unknown nonlinear dynamics, and hence usually perform better [33].

1.3. Neural networks and their inconsistencies

With the recent advances in Deep Learning (DL), NN-based solutions are gaining in popularity to represent unknown and potentially highly nonlinear dynamics, making them state-of-the-art solutions for time series modeling [34]. Unsurprisingly, given the broad range of applications of NNs, researchers have already applied them to model building thermal dynamics, for example in [5,35,36]. When applying NNs to physical systems, one should however keep their well-known generalization issue in mind [37], which is partially caused by the *underspecification* plaguing DL applications [38]. This can indeed be particularly problematic for NNs modeling physical processes since they are physics-agnostic and might hence find unrealistic solutions [4]. If the training data set does not span all the operating conditions of the system, there is thus a risk for NNs to fail to generalize in a meaningful manner to new conditions, an issue that is typically expected for thermal building models since the collected data sets are generally inherently incomplete [39].

Deep NNs are indeed able to learn *shortcuts* [40], which means they might fit the training data well without fundamentally understanding the problem, hence failing to generalize. They for example attain superhuman performance on image recognition tasks, and yet fail when undistinguishable noise is added [37] or the background

changes [41,42]. They are also able to generate captions without ever looking at the corresponding images [43], or detect pneumonia from X-ray scans only by looking at hospital-specific tokens and each hospital's pneumonia prevalence [44].

While these are only a few examples, they clearly indicate how NNs can find ways to perform extremely well without fundamentally solving the task at hand. These flawed models are however unable to generalize and cannot be deployed in real-world applications since we have no means to know how they will react to new conditions. While tasks such as object recognition and captioning might be hard to characterize in general, the case of physical system modeling is different. Indeed, we often know the underlying physical laws and can hence impose constraints on NNs that help them understand the task at hand.

1.4. Physics-inspired neural networks

To incorporate some knowledge of the underlying physics in NN training procedures and promote desired system properties, counterbalancing the aforementioned brittleness of classical NNs, researchers recently proposed to design *Physics-informed* or *Phyiscs-inspired* NNs (PiNNs) [45]. While many works modify the loss function of NNs to steer the learning towards physically meaningful solutions [46,47], these schemes cannot provide any guarantee about the final model respecting the desired constraints. More systematic approaches hence directly alter the networks' architecture to ensure the underlying physical laws are followed *by design*, i.e., at all times, such as in [39,48–50]. Additionally, since the desired properties are hard-coded in such models, the loss function does not need to be altered, which avoids common pitfalls of classical PiNNs, such as the difficult trade-off between the accuracy and the physical consistency of the model, which can also increase the amount of data needed [51]. These specific NN architectures ensuring some physical properties are philosophically related to the celebrated convolutional [52], recurrent [53], or graph [54] NN architectures, which were designed to capture spatial, timely, or neighboring relationships in the input data, respectively.

Despite the recent popularity of the field, to the best of the authors' knowledge, PiNNs were only applied to thermal building modeling in [55–58]. Gokhale et al. relied on the classical PiNN framework, augmenting the loss function of their NNs and creating latent states to include some physical intuition in otherwise standard networks [55], while Nagarathinam et al. designed a specific PiNN architecture for building control [56]. On the other hand, Drgoňa et al. used NNs to replace the matrices in linear models of building dynamics, which allowed them to enforce the stability and dissipativity of the learned system by constraining the eigenvalues of one of the NNs [57]. However, these works cannot provide guarantees about the physical consistency of their solutions in general beyond stability and dissipativity.

On the contrary, the Physically Consistent Neural Networks (PC-NNs) developed in previous work were theoretically proven to always yield physically consistent temperatures predictions, outperformed a classical gray-box model, and attained an accuracy on par with pure black-box models, but were limited to single-zone temperature modeling [39]. PCNNs are composed of a physics-inspired and a black-box module running in parallel, the former ensuring compliance with the underlying physical laws and the latter capturing unmodeled and potentially highly nonlinear dynamics. In a concurrent line of work, so-called PC-NODEs leveraged Irreversible port-Hamiltonian systems to ensure satisfaction of the first and second law of thermodynamics *by design*, relying on a similar training procedure as PCNNs [59]. Despite being applied to the modeling of a three-zone building, however, this framework considered preprocessed solar gains as inputs instead of raw irradiation measurements, removing most of the nonlinear dynamics. In other words, PC-NODEs propose an alternative version of the physics-inspired module, but have yet to be applied to more complex case studies where significant nonlinearities are present.

1.5. Contribution

In this work, we propose three different extensions of single-zone PCNNs [39] to model entire buildings, exemplifying how one can use the modularity of PCNNs to either expand the physics-inspired or black-box module. We then show that they all retain the desired physical consistency, both theoretically and with a numerical analysis, and investigate their performance on a three-zone building case study. Through extensive comparisons with several classical and state-of-the-art gray- and black-box methods, we demonstrate the ability of multi-zone PCNNs to leverage NNs to be more expressive than physically grounded gray-box methods and even outperform purely black-box NNs in terms of accuracy. Our investigations also clearly illustrate the phenomenon of shortcut learning on building temperature data, with NNs able to fit the data very accurately despite being completely oblivious to the underlying physics, which can be misleading in practice. Altogether, these experiments prove the effectiveness of the proposed PCNNs as thermal building models, alleviating the need for any engineering overhead, following the underlying physical laws, and reaching state-of-the-art accuracy.

The remainder of this paper is structured as follows. Section 2 first defines the notion of physical consistency and recalls the main principles behind PCNNs before the proposed multi-zone architectures are introduced and theoretically analyzed in Section 3. Section 4 then details the case study, implementation considerations, and baseline models. Finally, the results are analyzed in Sections 5 and 6 concludes the paper.

2. Background

In this section, we recall a few prerequisites required to understand the methods proposed in this work, clarifying some definitions, formally introducing the notion of physical consistency, and recalling the design of single-zone PCNNs.

2.1. Definitions

Two zones are said to be *adjacent* if they share at least one common wall in a building, and the collection of zones adjacent to a given zone z form its *neighborhood* $\mathcal{N}(z)$. Note that we consider a zone to be included in its own neighborhood, i.e., $z \in \mathcal{N}(z)$. Similarly, a zone is connected with the outside if it comprises at least one external wall. To generalize the notion of neighborhood, we define the *n-hop neighborhood* $\mathcal{N}^n(z)$ as the set of zones that can be reached in n steps from zone z, moving to an adjacent zone at each step. Note that, by definition, we have $\mathcal{N}^1(z) = \mathcal{N}(z)$, and $y \in \mathcal{N}^n(z) \iff z \in \mathcal{N}^n(y)$.

Throughout this work, we assume the building to be *connected*, i.e., there is no zone (or group of zones) isolated from the rest. This assumption is trivial in practice as one can easily train several separate models if this condition is not met.

2.2. Physical consistency

In this paper, we deem the temperature model of a building $\mathcal{B}$ with m thermal zones to be *physically consistent* if the following conditions are met for each zone $z \in \mathcal{B}$:

$$\frac{\partial T^z_{k+i}}{\partial u^z_{k+j}} > 0 \qquad \forall 0 \leq j < i, \tag{1}$$

$$\frac{\partial T^z_{k+i}}{\partial T^{out}_{k+j}} > 0 \qquad \forall 0 \leq j < i, \tag{2}$$

$$\frac{\partial T^z_{k+i}}{\partial T^y_{k+j}} > 0 \qquad \forall 0 \leq j < i,\ \forall y \in \mathcal{N}^{i-j}(z), \tag{3}$$

where T^z_k is the temperature in zone z at time step k, u its heating/cooling power input, and T^{out} represents the outside temperature.

For example, (1) implies that applying more heating power u^z_{k+j} at time step $k+j$ leads to higher temperatures T^z_{k+i} for all subsequent time steps $i > j$. In other words, heating a zone has the expected and intuitive impact of increasing its temperature, following the laws of thermodynamics. Note that cooling powers are defined to be negative in this work, hence inducing lower temperatures when more cooling is applied, as expected. Similarly, (2) ensures that higher ambient temperatures induce higher temperatures inside, and (3) guarantees that higher temperatures in zone $y \in \mathcal{N}^n(z)$ lead to higher temperatures in zone z after n steps.

Remark 1 (*Generalization of the Approach*)**.** Note that the definition of physical consistency proposed in (1)–(3) can easily be extended for applications where additional criteria need to be met by the learned model, to enforce physically consistent temperature predictions with respect to solar gains, for example. Interestingly, these conditions can also be seamlessly adapted to other fields beyond building modeling where simple physical rules can be encoded in a similar fashion. One can then construct a PCNN architecture following the principles presented in Sections 2.3 and 3 to ensure the learned model respect these desired criteria.

2.3. Single-zone PCNNs

Conceptually, PCNNs are composed of a black-box and a physics-inspired module running in parallel to compute the next output at each step, as depicted on the right of Fig. 1. The former captures potentially complex nonlinearities while the latter ensures that predefined rules are respected, which typically represent physical laws and can be encoded by conditions similar to the ones proposed in (1)–(3). In the case of PCNNs modeling the evolution of the temperature in a single thermal zone z while respecting the criteria in (1)–(3), they can be mathematically described as follows [39]:

$$D_{k+1} = D_k + f(x_k, D_k), \tag{4}$$

$$E_{k+1} = E_k + a_h \max\{u_k, 0\} + a_c \min\{u_k, 0\} - b(T_k - T^{out}_k) - \sum_{z' \in \mathcal{N}(z)} c_{z'}(T_k - T^{z'}_k), \tag{5}$$

$$T_{k+1} = D_{k+1} + E_{k+1}, \quad D_k = T(k), \quad E_k = 0, \tag{6}$$

where $D \in \mathbb{R}$ represents the evolution of the black-box module based on a freely parametrized function $f : \mathbb{R}^{d+1} \to \mathbb{R}$, typically composed of NNs, and $E \in \mathbb{R}$ is the energy accumulator, i.e., the physics-inspired module. The latter is influenced by the power inputs $u \in \mathbb{R}$ and heat transfers to the outside and adjacent zones. On the other hand, $x \in \mathbb{R}^d$ regroups all the exogenous inputs that are not included in E, such as time information and solar irradiation . Finally, the constants a_h, a_c, b, and $\{c_{z'}\}_{z' \in \mathcal{N}(z)}$ capture the impact of heating, cooling, and heat losses to the outside and the neighboring zones on the modeled zone temperature, respectively. Applying (4)–(6) recursively over the prediction horizon, starting from the measured temperature $T(k)$ at time k, PCNNs can predict the evolution of the temperature while satisfying the criteria in (1)–(3). We refer the reader to the original paper for additional details [39].

One important key to the effectiveness and generality of PCNNs comes from the fact that all the parameters a_h, a_c, b, $\{c_{z'}\}_{z' \in \mathcal{N}(z)}$, and f are learned simultaneously using automatic BackPropagation Through Time (BPTT) [39]. As mentioned in Remark 1, the very generic structure of PCNNs can also be applied to model complex phenomena beyond thermal modeling, typically where only part of the physics is well understood. Indeed, it is always possible to adapt the structure of the physics-inspired module, which might also include nonlinearities, let the black-box module capture completely unknown dynamics in parallel, and seamlessly learn everything simultaneously in an end-to-end pipeline.

Remark 2 (*Consistency with Respect to Initial Conditions*)**.** Since we want to model several zones in this paper, the condition (3) is different from the one enforced in [39]. In particular, it includes the case $y = z$, $j = 0$: we want the derivative of any zone temperature with respect to its initial temperature to be positive to avoid spurious effects. This was not considered in the original paper on single-zone modeling and requires a slight modification of (4), as discussed in the following section.

3. Modeling the thermal behavior of buildings

This section proposes three different extensions of the single-zone PCNN architecture, depicted in Fig. 2, to model an entire building and then discusses their physical consistency.

3.1. Extensions to multi-zone PCNNs

While the PCNN architecture described in Section 2.3 was shown to work well for single-zone modeling, this work proposes three possible extensions of this framework to simultaneously capture the evolution of the temperature in several interconnected zones exchanging energy, i.e., in a whole building. The only additional information required is the *topology* of the modeled building, i.e., we assume to know which zones are adjacent and which have an external wall, and then learn its thermal behavior from data without engineering overhead.

Remark 3 (*Topology*)**.** If the topology is unknown, one can also assume each pair of zones to be adjacent and every zone to have an external wall and then learn to put non-existing connection parameters to zero from data.

3.1.1. X-PCNNs: learning several single-zone PCNNs

The most natural and straightforward extension is to separately learn one PCNN for each of the zones to model, as depicted in Fig. 2(a). Since this method involves duplicating the original structure for each zone and fitting them independently, we will refer to the final model of the building as the *X-PCNN* architecture. Mathematically, for a given zone z, we can write the corresponding equations as follows:

$$D^z_{k+1} = D^z_k + f^z(x^z_k), \tag{7}$$

$$E^z_{k+1} = E^z_k + a^z_h \max\{u^z_k, 0\} + a^z_c \min\{u^z_k, 0\} - b(T^z_k - T^{out}_k) - \sum_{y \in \mathcal{N}(z)} \tilde{c}^z_y (T^z_k - T^y_k), \tag{8}$$

$$T^z_{k+1} = D^z_{k+1} + E^z_{k+1}, \quad D^z_k = T^z(k), \quad E^z_k = 0, \tag{9}$$

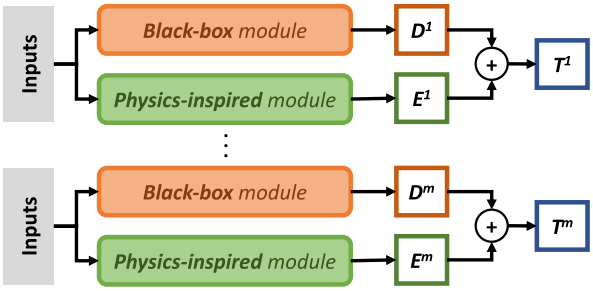
where the superscript z denotes zone-dependent information and all the variables have the same meaning as in Section 2.3.

To retain the physical consistency of this model, however, one needs to ensure that $\tilde{c}^z_y = \tilde{c}^y_z$ for each pair of adjacent zones y and z, so that the amount of energy flowing from z to y always equals the amount of energy received by y from z, and vice versa. Since each zone is modeled and trained separately in this case, such a condition cannot be imposed during the learning phase, and we thus rely on a heuristic to correct the parameters and enforce this desired property *a posteriori*. Once the models have been trained, for every pair of adjacent zones z and y, we compute the average value identified by both PCNNs and define:

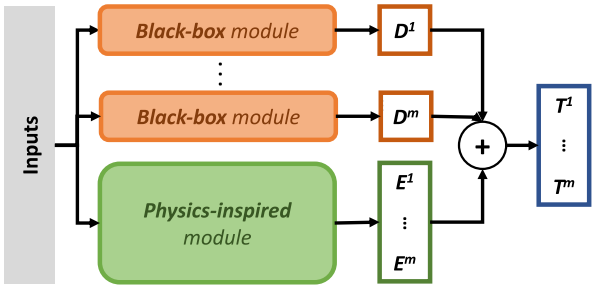
$$c^{zy} = c^{yz} = \frac{\tilde{c}^z_y + \tilde{c}^y_z}{2}. \tag{10}$$

For every zone z, we then replace $\tilde{c}^z_y$ with c^{zy} in (8) for all $y \in \mathcal{N}(z)$.

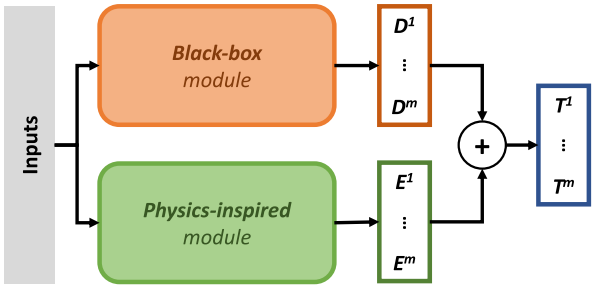
Remark 4 (*Independence of f^z from D^z*)**.** Note the difference between (7) and (4), with D^z not appearing in f^z in (7). Following Remark 2, this ensures that condition (3) is respected at all times, as analyzed in Section 3.2 and Remark 6.



(a) **X-PCNN**: the temperature of each of the m zones is predicted separately.



(b) **M-PCNN**: the physics-inspired module is shared but multiple black-box modules are learned, one for each zone.



(c) **S-PCNN**: both the black-box and physics-inspired modules are shared by all the zones.

Fig. 2. The three PCNN architectures proposed in this work, with different levels of information sharing between the modeled thermal zones.

3.1.2. M-PCNNs: sharing the physics-inspired module

To avoid the hand-crafted correction (10), which might significantly impact the parameters learned by each PCNN, one can fuse all the physics-inspired modules together, again leveraging our prior knowledge of the underlying physical laws. This gives rise to the so-called *M-PCNN* architecture, pictured in Fig. 2(b), where distinct black-box modules are assigned to each zone, but the physics-inspired module is shared and outputs a vector $\boldsymbol{E} \in \mathbb{R}^m$ containing the energy accumulated in each zone at each step:

$$\begin{aligned} \boldsymbol{E}_{k+1} &= \boldsymbol{E}_k + \boldsymbol{a}_h \odot \max\{\boldsymbol{u}_k, \boldsymbol{0}\} \\ &\quad + \boldsymbol{a}_c \odot \min\{\boldsymbol{u}_k, \boldsymbol{0}\} \\ &\quad - \boldsymbol{b} \odot (\boldsymbol{T}_k - \boldsymbol{T}_k^{out}) - \boldsymbol{\Delta T}_k, \\ \boldsymbol{E}_k &= \boldsymbol{0}, \end{aligned} \tag{11}$$

where the bold notations correspond to vectorized quantities in $\mathbb{R}^m$, one dimension for each zone, i.e., $\boldsymbol{a_h} = [a_h^1, \dots, a_h^m]^T$, and similarly for $\boldsymbol{a_c}$, $\boldsymbol{b}$, and $\boldsymbol{u_k}$, and $\odot$ stands for the element-wise product of two vectors. Since there is a unique ambient temperature impacting all the zones, we furthermore define $\boldsymbol{T^{out}} = [T^{out}, \dots, T^{out}]^T \in \mathbb{R}^m$. Finally, $\boldsymbol{\Delta T}_k \in \mathbb{R}^m$ corresponds to energy transfer between each zone and its neighborhood:

$$\boldsymbol{\Delta T}_k^z = \sum_{y \in \mathcal{N}(z)} c^{zy}(T_k^z - T_k^y), \qquad \forall z \in \mathcal{B} \tag{12}$$

where the subscript z denotes the zth entry of a vector. By definition, we know $c^{zy} = c^{yz}$ if y and z are adjacent since both represent the same heat transfer coefficient, which is easily enforced during training since all the zones are now modeled simultaneously, avoiding the *a posteriori* correction (10) required for X-PCNNs.

Each dimension of $\boldsymbol{T} \in \mathbb{R}^m$, i.e., the temperature in each zone z, is then computed as the sum of the physics-inspired and black-box modules, as before:

$$T_{k+1}^z = D_{k+1}^z + E_{k+1}^z, \tag{13}$$

$$\begin{aligned} D_{k+1}^z &= D_k^z + f^z(x_k^z), \\ D_k^z &= T^z(k). \end{aligned} \tag{14}$$

3.1.3. S-PCNNs: sharing both modules

To reduce the computational complexity of the model and introduce parameter sharing between the zones — which are typically similar in the same building —, we propose a third architecture, dubbed *S-PCNN*, where both the black-box and physics-inspired modules are shared. In practice, this means that the black-box module, typically consisting of NNs, now has m outputs corresponding to the main dynamics of each of the zones, as pictured in Fig. 2(c). Using the vectorized notations as before, i.e., $\boldsymbol{D} \in \mathbb{R}^m$, we can write the equations of this architecture as follows:

$$\boldsymbol{D}_{k+1} = \boldsymbol{D}_k + \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_k), \tag{15}$$

$$\begin{aligned} \boldsymbol{E}_{k+1} &= \boldsymbol{E}_k + \boldsymbol{a}_h \odot \max\{\boldsymbol{u}_k, \boldsymbol{0}\} \\ &\quad + \boldsymbol{a}_c \odot \min\{\boldsymbol{u}_k, \boldsymbol{0}\} \\ &\quad - \boldsymbol{b} \odot (\boldsymbol{T}_k - \boldsymbol{T}_k^{out}) - \boldsymbol{\Delta T}_k, \end{aligned} \tag{16}$$

$$\begin{aligned} \boldsymbol{T}_{k+1} &= \boldsymbol{D}_{k+1} + \boldsymbol{E}_{k+1}, \\ \boldsymbol{D}_k &= \boldsymbol{T}(k), \\ \boldsymbol{E}_k &= \boldsymbol{0}, \end{aligned} \tag{17}$$

where the physics-inspired module is the same as for the M-PCNN but we now only have one shared nonlinear function $\tilde{\boldsymbol{f}} : \mathbb{R}^{d'} \to \mathbb{R}^m$ transforming the inputs $\tilde{\boldsymbol{x}} \in \mathbb{R}^{d'}$. Throughout this work, we only consider external inputs that are shared by all the zones, i.e. $d' = d$ and $\tilde{\boldsymbol{x}} := x^z, \ \forall z \in \mathcal{B}$.

Remark 5 (*Zone-Dependent Inputs*)**.** If some measurements differ zone by zone, one can either stack them in a vector $\tilde{\boldsymbol{x}} = [(x^1)^\top, \dots, (x^m)^\top]^\top$ and use (15) as is or for example design a shared function $\tilde{\boldsymbol{f}} : \mathbb{R}^d \to \mathbb{R}$ and modify (15) to $\boldsymbol{D}_{k+1}^z = \boldsymbol{D}_k^z + \tilde{\boldsymbol{f}}(x_k^z), \forall z \in \mathcal{B}$.

3.2. Thermodynamical consistency

Relying on the transformation (10) ensuring that heat transfer coefficients between each adjacent zones are equal in the corresponding single-zone PCNNs, one can vectorize the X-PCNN physics-inspired module (8), putting the parameters of each zone in vectors, to get:

$$\begin{aligned}\boldsymbol{E}_{k+1} = \boldsymbol{E}_k &+ \boldsymbol{a}_h \odot \max\{\boldsymbol{u}_k, \boldsymbol{0}\} \\ &+ \boldsymbol{a}_c \odot \min\{\boldsymbol{u}_k, \boldsymbol{0}\} \\ &- \boldsymbol{b} \odot (\boldsymbol{T}_k - \boldsymbol{T}_k^{out}) - \boldsymbol{\Delta T}_k,\end{aligned} \tag{18}$$

using the definition of $\boldsymbol{\Delta T}$ from (12). As can be seen directly, this expression is the same as the ones describing physics-inspired modules of the M-PCNN and S-PCNN architectures in (11) and (16). This means all the proposed multi-zone PCNNs rely on the same physical model at inference time, with however possibly different parameter values learned during training. This is intuitively expected since they all model the same thermal effects and hence have to follow the same physical principles.

Similarly, we can rewrite the black-box modules of the X-PCNN and M-PCNN architectures in vectorized form as:

$$\boldsymbol{D}_{k+1} = \boldsymbol{D}_k + \bar{\boldsymbol{f}}(\bar{\boldsymbol{x}}_k), \tag{19}$$

where $\bar{\boldsymbol{f}} = [f^1(x^1), \dots, f^m(x^m)]^T$ and $\bar{\boldsymbol{x}} = [(x^1)^\top, \dots, (x^m)^\top]^\top$ groups the different inputs.

For the three proposed architectures, putting (18) and (19) together, we hence get:

$$\begin{aligned}\boldsymbol{T}_{k+1} &= \boldsymbol{D}_{k+1} + \boldsymbol{E}_{k+1} \\ &= \boldsymbol{T}_k + \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{a}_h \odot \max\{\boldsymbol{u}_k, \boldsymbol{0}\} \\ &\quad + \boldsymbol{a}_c \odot \min\{\boldsymbol{u}_k, \boldsymbol{0}\} \\ &\quad - \boldsymbol{b} \odot (\boldsymbol{T}_k - \boldsymbol{T}_k^{out}) - \boldsymbol{\Delta T}_k, \\ \boldsymbol{D}_0 &= \boldsymbol{T}(k), \\ \boldsymbol{E}_0 &= \boldsymbol{0},\end{aligned} \tag{20}$$

where $\boldsymbol{f}(\boldsymbol{x}_k)$ stands for $\tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_k)$ or $\bar{\boldsymbol{f}}(\bar{\boldsymbol{x}}_k)$ for S-PCNNs, respectively X- and M-PCNNs. The only structural difference between the three proposed models (once the heat transfer coefficients of the X-PCNN have been adjusted) hence comes from the form of $\boldsymbol{f}(\boldsymbol{x})$. Remarkably, however, this does not impact their physical consistency, as demonstrated in the following two propositions.

Proposition 1 (*Heat Propagation*). *Independently of the structure of* $\boldsymbol{f}$ *and* $\boldsymbol{x}$*, any model of the form* (20) *satisfies:*

$$\frac{\partial \boldsymbol{T}_{k+i}^z}{\partial \boldsymbol{T}_{k+j}^y} \geq 0 \qquad \forall z, y \in \mathcal{B},\ \forall 0 \leq j < i, \tag{21}$$

with equality if and only if $y \notin \mathcal{N}^{(i-j)}(z)$*, as long the following conditions hold:*

$$b^z + \sum_{y \in \mathcal{N}(z)} c^{zy} < 1, \qquad \forall z \in \mathcal{B}, \tag{22}$$

$$c^{zy} > 0, \qquad \forall z \in \mathcal{B},\ \forall y \in \mathcal{N}(z). \tag{23}$$

Proof. See Appendix A.1. □

In words, Proposition 1 means that heat propagates from any zone y to all the other zones z as physically expected, i.e., higher temperatures in a given zone y will lead to higher temperatures in all the other zones after $(i-j)$ steps.

Proposition 1 can then be used to prove the following proposition stating that heating or cooling any zone ultimately increases, respectively decreases, the temperature in the whole building through heat transfers and that higher and lower ambient temperatures also impact the building as expected.

Proposition 2 (*Physical Consistency with Respect to Inputs*). *Independently of the structure of* $\boldsymbol{f}$ *and* $\boldsymbol{x}$*, any model of the form* (20) *satisfies:*

$$\frac{\partial \boldsymbol{T}_{k+i}^z}{\partial \boldsymbol{u}_{k+j}^y} \geq 0, \qquad \forall z, y \in \mathcal{B},\ \forall 0 \leq j < i, \tag{24}$$

with equality if and only if $y \notin \mathcal{N}^{(i-j-1)}(z)$*, and*

$$\frac{\partial \boldsymbol{T}_{k+i}^z}{\partial \boldsymbol{T}_{k+j}^{out}} > 0, \qquad \forall z \in \mathcal{B},\ \forall 0 \leq j < i, \tag{25}$$

as long as (22)–(23) *hold and:*

$$a_h^z, a_c^z, b^z > 0, \qquad \forall z \in \mathcal{B}. \tag{26}$$

Proof. See Appendix A.2. □

Corollary 1 (*Physical Consistency of PCNNs*). *Independently of the structure of* $\boldsymbol{f}$ *and* $\boldsymbol{x}$*, any model of the form* (20) *respects the physical consistency criteria* (1)–(3) *if:*

$$b^z + \sum_{y \in \mathcal{N}(z)} c^{zy} < 1, \qquad \forall z \in \mathcal{B}, \tag{27}$$

$$a_h^z, a_c^z, b^z, c^{zy} > 0, \qquad \forall z \in \mathcal{B},\ \forall y \in \mathcal{N}(z). \tag{28}$$

Proof. Assuming that (27) and (28) hold, we can apply Propositions 1 and 2. Setting $z = y$ in (24) and recalling that any zone is in its own neighborhood — which implies strict positiveness of (24) —, PCNNs satisfy (1). The satisfaction of (2) directly follows from the second part of Proposition 2. Finally, according to Proposition 1, (21) is strictly positive if and only if $y \in \mathcal{N}^{(i-j)}(z)$, satisfying (3). □

This corollary thus proves that each of the proposed PCNN architectures remains physically consistent as long as all the parameters $\boldsymbol{a_h}$, $\boldsymbol{a_c}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ are small positive constants. Note that this makes intuitive sense since all these parameters correspond to inverses of resistances and capacitances, hence small positive numbers, in real buildings. Interestingly, these conditions can easily be enforced during the training procedure without modifying the classical backpropagation through time algorithm, hence allowing us to rely on well-developed tools to train our models, as detailed in Section 4.2.

Remarkably, the strength of our approach lies in the fact that all the models will remain consistent whatever the structure of $\boldsymbol{f}$ is, being shared or not,[1] composed of NNs or other nonlinearities. This gives the user complete freedom in the design of the black-box module without jeopardizing the physical consistency of the model. Similarly, all the parameters of the physics-inspired module might for example be time-varying or computed as nonlinear functions of external inputs without impacting the consistency of the model as long as they stay small and positive at all times.

Remark 6 (*Inputs of* $\boldsymbol{f}$). While the structure of $\boldsymbol{f}$ does not impact the validity of Propositions 1 and 2, its inputs do. In particular, $\boldsymbol{f}$ has to be independent of $\{T, u, T^{out}\}$ for the first step of the proofs of both propositions to hold in general (Appendix A). If $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{D})$ for example, it would modify the case $z = y$ in Eq. (40) in Appendix A.1, and the satisfaction of (22) would then not be sufficient to guarantee the required nonnegativity of the partial derivatives in (21).

Remark 7 (*A Control Perspective*). The multi-zone PCNNs (20) are power input-affine. This makes such models interesting in control applications, typically for MPC schemes aimed at decreasing the energy consumption of buildings.

[1] This is the main difference between the M-PCNN and S-PCNN architectures in our case, for example.

Fig. 3. NEST, Duebendorf, Switzerland, with UMAR highlighted in white © Zooey Braun, Stuttgart.

4. Case study

To assess the quality of the multi-zone PCNN architectures detailed in Section 3, we carry out an extensive performance analysis on a case study, where the objective is to predict the temperature dynamics over three-day-long horizons with 15 min time steps. This section presents the building where the data was collected, implementation details, and then introduces the other gray- and black-box methods used as benchmarks.

4.1. Data set

The data used in this work was collected in NEST, a vertically integrated district composed of several units [60], located in Duebendorf, Switzerland, and pictured in Fig. 3. In this paper, we focus on the Urban Mining and Recycling (UMAR) unit, circled in white in Fig. 3, an apartment with two bedrooms and a living room in between them. We are thus modeling three thermal zones arranged in a line in this work, i.e., Zone 1 is connected to Zone 2, and Zone 2 is also connected to Zone 3, and each of them has at least one external wall. In heating mode, all the zones are heated by letting hot water flow through ceiling panels. During the cooling season, on the other hand, cold water can flow through the panels to cool down the zones.

We rely on three years of data collected between May 2019 and May 2022 and preprocessed as explained in [39, App. C]. This involved downsampling the data to 15 min intervals, smoothing the time series, and disaggregating the thermal power consumption of UMAR into the consumption of each zone. Besides these computed thermal power inputs, the data set also contains measurements of the temperature in each zone and outside, the horizontal solar irradiation on-site, and the status of the system, i.e., if it is in heating or cooling mode. To facilitate the learning process of the NNs in the black-box modules, we completed the data with additional time information, i.e., the day of the week and the sine and cosine transformations of the time of the day and the month [39, App. D]. Finally, we split the data in a training and a validation set, respectively denoted $\mathcal{D}_t$ and $\mathcal{D}_v$, containing possibly overlapping time series of up to three days of data, as detailed in Appendix B.

4.2. Implementation details

To ensure the physical consistency of the proposed multi-zone PCNNs, i.e., to fulfill the conditions (27) and (28), we parametrize the log-value of each parameter, i.e., we learn $\tilde{a}_h^z, \tilde{a}_c^z, \tilde{b}^z, \tilde{c}^{zy}$, $\forall z \in \mathcal{B}$, $\forall y \in \mathcal{N}(z)$ and define:

$$s = s_0 \exp(\tilde{s}), \qquad \forall s = \{a_h^z, a_c^z, b^z, c^{zy}\} \tag{29}$$

where s_0 is the initial value of the parameter, defined using the same rules of thumb as in [39]. Starting from $\tilde{s} = 0$, PCNNs hence learn to scale the initial value s_0 instead of modifying it directly, which is more numerically stable and ensures that s stays small enough, while the exponential function keeps all the parameters positive at all times. Note that these parameters are learned simultaneously to the parameters in the black-box module: when backpropagation is used to update the parameters of the NNs, we also leverage the propagated gradients to update the parameters of the physics-inspired module.

Remark 8 (*Upperbound on s*). While $\tilde{b}^z$ or $\tilde{c}^{zy}$ can in principle grow uncontrollably and lead to a violation of the necessary condition (27), this was not an issue in our experiments. Nonetheless, one can always introduce bounds on the learned values $\tilde{s}$ if required, typically leveraging activation functions like the sigmoid or hyperbolic tangent to control the range of learned values.

In this paper, each function $\boldsymbol{f}$ has the same encoder-LSTM-decoder architecture, which is repeated when several modules are required for X-PCNNs and M-PCNNs. Both the encoder and decoder are feedforward NNs with 32 hidden units and the LSTM comprises two layers with dimension 64 and is followed by a normalization layer. This architecture was selected over larger ones since we did not observe any significant decrease in performance. The input of each black-box module, $\boldsymbol{x} \in \mathbb{R}^6$, gathers the solar irradiation on a horizontal surface and the time information. NNs share a common learning rate of 5e−4, manually selected small enough to ensure stable convergence, and a batch size of 4096, to maximize the utility of the Graphical Processing Units (GPUs). Every model minimizes the Mean Square Error (MSE) over a given batch of data B:

$$\mathcal{L}_{data} = \frac{1}{|B|} \sum_{s \in B} \left[\frac{1}{l_s} \sum_{k=0}^{l_s-1} \left[\frac{1}{m} \sum_{z=1}^{m} \xi_k^{z,s} \right] \right], \tag{30}$$
$$\xi_k^{z,s} = \left(\boldsymbol{T}_{k+1}^{z,s} - T^{z,s}(k+1) \right)^2,$$

where the subscript s corresponds to which time series of length l_s the predictions and measured temperatures are taken from. To then validate the performance of a model, we rely on the Mean Absolute Error (MAE), where:

$$\xi_k^{z,s} = |\boldsymbol{T}_{k+1}^{z,s} - T^{z,s}(k+1)|,$$

in (30) and the Mean Absolute Percentage Error (MAPE), with:

$$\xi_k^{z,s} = \frac{|\boldsymbol{T}_{k+1}^{z,s} - T^{z,s}(k+1)|}{T^{z,s}(k+1)}.$$

The PCNNs are implemented in PyTorch [61] and were trained on NVIDIA P100 GPUs. The code and data are available on https://gitlab.nccr-automation.ch/loris.dinatale/multi-zone-pcnns.

4.3. Benchmark models

To analyze the performance of the proposed PCNN architectures, we perform an extensive ablation study and compare them to state-of-the-art gray- and black-box methods. An overview of all the models used in this work, and whether they are physically consistent, can be found in Table 1. Note that the linear and LSTM models correspond to learning only the physics-inspired module of S-PCNNs, respectively the black-box one. Furthermore, *Res-cons* is equivalent to fitting both modules of S-PCNNs sequentially, showcasing the importance of learning all the parameters of PCNNs simultaneously to attain state-of-the-art accuracy.

4.3.1. Linear gray-box model

First, it makes intuitive sense to investigate the accuracy of the physics-inspired module of PCNNs on its own, leading to the following linear gray-box model, hereafter referred to as the *Linear* model:

$$\boldsymbol{T}_{k+1} = \boldsymbol{T}_k + \boldsymbol{a}_h \odot \max\{\boldsymbol{u}_k, \boldsymbol{0}\}$$

Table 1
Physical consistency, MAE, and MAPE of the methods investigated in this work.

Category	Model	Phys. cons.	MAE	MAPE
Gray-box	*Linear*	✓	1.79	7.5%
	Res	✗	1.79	7.7%
	Res-cons	✓	1.50	6.4%
Black-box	*ARX*	✗	1.68	7.1%
	ARX-KF	✗	1.35	5.6%
	LSTM	✗	1.27	5.5%
	PiNN	✗	1.37	5.8%
PCNNs	***X-PCNN* (Ours)**	✓	**1.17**	**4.9%**
	M-PCNN (Ours)	✓	1.25	5.3%
	S-PCNN (Ours)	✓	1.22	5.1%

$$+ \boldsymbol{a}_c \odot \min\{\boldsymbol{u}_k, \boldsymbol{0}\} \\ - \boldsymbol{b} \odot (\boldsymbol{T}_k - \boldsymbol{T}_k^{out}) - \boldsymbol{\Delta T}_k + \boldsymbol{e} \odot \boldsymbol{Q}_k^{win}, \tag{31}$$

where $\boldsymbol{Q}_k^{win}$ gathers the solar irradiation on the windows of each zone in a vector, engineered from the measured irradiation on a horizontal surface. Since there is no black-box module taking care of the impact of the sun on building temperatures in this model, we indeed need to include it manually. This can be done efficiently for UMAR but does not generalize to arbitrary buildings, limiting the applications of such linear models, as detailed in Appendix C. As for the other heat gains, $\boldsymbol{e}$ gathers the trainable scaling parameters reflecting the impact of solar gains on each zone temperature in a vector.

Since the classical least squares parameter identification gave rise to physically inconsistent parameters, we chose to identify $\boldsymbol{a}_h, \boldsymbol{a}_c, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{e}$ for each zone using Bayesian Optimization (BO), as detailed in Appendix D. As for X-PCNNs, the heat transfer coefficients between two adjacent thermal zones were then averaged based on (10).

4.3.2. Residual models

A natural extension of the aforementioned linear model is to consider *residual models*, where the idea is to fit the errors of the linear model predictions with a black-box module to improve its performance. Assuming the linear model in (31) to provide predictions $\hat{\boldsymbol{T}}_{k+1}$, a residual model fits a function $f_{Res} : \mathbb{R}^{d'+2m+1} \to \mathbb{R}^m$, typically modeled with NNs, to the residual errors, i.e., it minimizes:

$$\mathcal{L}_{Res} = \frac{1}{|B|} \sum_{s \in B} \left[\frac{1}{l_s} \sum_{k=0}^{l_s-1} \left[\frac{1}{m} \sum_{z=1}^{m} (\epsilon_k^{z,s})^2 \right] \right], \tag{32}$$
$$\epsilon_k^{z,s} = f_{Res}^z(\boldsymbol{T}_k^s, \boldsymbol{x}_k^s, \boldsymbol{u}_k^s, \boldsymbol{T}_k^{out,s}) - \left(\boldsymbol{T}^{z,s}(k+1) - \hat{\boldsymbol{T}}_{k+1}^{z,s} \right),$$

and then predicts temperatures as follows:

$$\boldsymbol{T}_{k+1} = \hat{\boldsymbol{T}}_{k+1} + f_{Res}(\boldsymbol{T}_k, \boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{T}_k^{out}). \tag{33}$$

This model is dubbed *Res* in the rest of this paper, and f_{Res} has the same encoder-LSTM-decoder structure as the proposed PCNNs for fair comparisons.

Remarkably, such residual models cannot be ensured to respect the underlying physical laws in general since f_{Res} is not independent of zone temperatures, power inputs, and ambient temperatures. Since they are composed of a physics-inspired base model and a black-box module running in parallel, as the proposed PCNN architectures, we can indeed use similar arguments to prove their physical consistency (see Remark 6). Consequently, we also investigate the performance of a physically consistent residual model in this work, dubbed *Res-cons*, where the black-box function learning the residuals only depends on $\boldsymbol{x}$, as PCNNs. This model hence fits a function $f_{Res\text{-}cons} : \mathbb{R}^{d'} \to \mathbb{R}^m$ to the residuals, trained similarly to its physically inconsistent counterpart, with the following temperature predictions:

$$\boldsymbol{T}_{k+1} = \hat{\boldsymbol{T}}_{k+1} + f_{Res\text{-}cons}(\boldsymbol{x}_k). \tag{34}$$

Note that residual models first fit the base model to the data and then use black-box methods to fit the residual errors while PCNNs learn both modules together. This also implies that the physics-inspired module reflects the main dynamics of residual models while it only ensures the physical consistency of PCNN architectures, letting more expressive functions like NNs capture the main system dynamics.

4.3.3. Autoregressive model with exogenous inputs

As a first black-box method, we analyze the performance of an ARX model, where autoregressive lags of the states and inputs are used to predict the next state:

$$\boldsymbol{T}_{k+1} = \boldsymbol{\alpha}_0 \boldsymbol{T}_k + \boldsymbol{\alpha}_1 \boldsymbol{T}_{k-1} + \ldots + \boldsymbol{\alpha}_\delta \boldsymbol{T}_{k-\delta} \\ + \boldsymbol{\beta}_0 \hat{\boldsymbol{x}}_k + \boldsymbol{\beta}_1 \hat{\boldsymbol{x}}_{k-1} + \ldots + \boldsymbol{\beta}_\delta \hat{\boldsymbol{x}}_{k-\delta}, \\ \hat{\boldsymbol{x}}_k = [\boldsymbol{u}_k, \boldsymbol{T}_k^{out}, \boldsymbol{Q}_k^{sun}]^T, \tag{35}$$

where $\boldsymbol{Q}_k^{sun} \in \mathbb{R}$ is the solar irradiation measurement on a horizontal surface, and the parameters $\boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_\delta \in \mathbb{R}^{m \times m}$, $\boldsymbol{\beta}_0, \ldots, \boldsymbol{\beta}_\delta \in \mathbb{R}^{m \times (m+2)}$ are identified through least square regression using the `scikit-learn` library [62]. For a fair comparison, we set $\delta = 11$, i.e., we use information from the last 3 h to define the next temperatures, similarly to the warm start period of the proposed PCNN architectures.

For comparison purposes, we also implemented an advanced ARX model relying on the `statsmodels` package [63], which includes Kalman smoothing and filtering operations out-of-the-box and sets $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_\delta = 0$ so that only current information on external inputs is used. Since the identification procedure was harder in that case, we identified each zone z separately, with:

$$\hat{\boldsymbol{x}}_k^z = [\boldsymbol{u}_k^z, \boldsymbol{T}_k^{out}, \boldsymbol{Q}_k^{sun}, \boldsymbol{T}_k^{y_1}, \boldsymbol{T}_k^{y_2}, \ldots, \boldsymbol{T}_k^{y_{|\mathcal{N}(z)|}}]^T,$$

where $y_1, \ldots, y_{|\mathcal{N}(z)|} \in \mathcal{N}(z)$ are the zones adjacent to z, and the final model is dubbed *ARX-KF*. Note that ARX models cannot be enforced to be physically consistent in general.

Remark 9 (*Kalman Filtering and Smoothing*)**.** Note that these operations could be included for any other model as well, potentially impacting their performance. In this work, we focus on methods working on unfiltered data, typically involving NNs, but we also provide this *ARX-KF* as an example of what can be achieved with existing toolboxes on a laptop compared to NN-based methods that might require access to GPUs for training.

4.3.4. Neural network models

As another natural ablation of PCNNs, we also investigate the quality of the black-box module alone. Instead of treating the power inputs and temperatures in a separate module, all the inputs are fed in the black-box function $f_{LSTM} : \mathbb{R}^{d'+2m+1} \to \mathbb{R}^m$, leading to the *LSTM* model:

$$\boldsymbol{T}_{k+1} = \boldsymbol{T}_k + f_{LSTM}(\boldsymbol{T}_k, \boldsymbol{u}_k, \boldsymbol{x}_k, \boldsymbol{T}_k^{out}). \tag{36}$$

As expected, such classical NN-based methods are naturally physically inconsistent and might fail to capture the underlying physical laws even if they fit the data well (Section 1).

Finally, we also compare PCNNs to a standard physics-informed NN, hereafter the *PiNN* model, again relying on the same architecture as the black-box modules of PCNNs and the LSTM model. However, as is classically done, its loss function is modified to steer the learning toward physically meaningful solutions, with:

$$\mathcal{L}_{PiNN} = \mathcal{L}_{data} + \lambda \mathcal{L}_{phys}, \tag{37}$$

where λ is a tuning hyperparameter. Since the purpose of this additional loss term $\mathcal{L}_{phys}$ is to capture physical inconsistencies and penalize them, we naturally design it to bias the model towards solutions satisfying the desired properties (1) and (2). Consequently, we penalize negative gradients of the final predicted temperatures, i.e., at time l_s,

with respect to control inputs and ambient temperatures observed along the horizon:

$$\mathcal{L}_{phys} = \frac{1}{|B|} \sum_{s \in B} \left[\frac{1}{l_s} \sum_{k=0}^{l_s-1} \left[\frac{1}{m} \sum_{z=1}^{m} g_k^{z,s} \right] \right], \tag{38}$$

$$g_k^{z,s} = \sum_{y=1}^{m} \left[r\left(-\frac{\partial \boldsymbol{T}_{l_s}^{z,s}}{\partial \boldsymbol{u}_k^{y,s}} \right) \right] + r\left(-\frac{\partial \boldsymbol{T}_{l_s}^{z,s}}{\partial T_k^{out,s}} \right), \tag{39}$$

where $r(x) = \max\{x, 0\}$, also known as the Rectified Linear Unit (ReLU) function. Since we are interested in physically consistent models in this work, we empirically fixed $\lambda = 100$, which ensures the loss term is dominated by the physical inconsistencies, thereby steering the PiNN towards interesting solutions.

Note that, in building temperature modeling, one can also augment the outputs of NNs to predict not only zone temperatures but also the temperatures of their respective thermal mass, for example, and then penalize deviations of the latter from the predictions of a physics-based model in $\mathcal{L}_{phys}$ to incorporate prior knowledge in PiNNs [58]. However, this requires access to a physics-based model, introducing engineering overhead. Moreover, it can enforce unwanted biases since the physics-based model might be inaccurate and steer PiNN predictions away from the truth. Consequently, in this work, we penalize the gradients of the temperature predictions instead, according to our definition of physical consistency in Section 2.2, which bypasses the need for a physics-based model and only relies on measured quantities while still incorporating knowledge about the underlying laws of physics in PiNNs.

Remark 10 (*Computational Complexity*)**.** To ensure a model is following the underlying physical laws at all times, one should check the gradients throughout the prediction horizon, and not only for the last predictions, as proposed in (38). However, since each gradient computation requires one forward and one backward pass of the data, the computational complexity grows linearly with the number of predictions to analyze. Consequently, we only compute the gradients of the last predictions with respect to all the control inputs and ambient temperatures observed along the horizon to steer PiNNs, alleviating the associated computational burden.

5. Results

This section provides a comprehensive analysis of the models presented in Table 1. All the results discussed hereafter were obtained by comparing the multi-step prediction performance of the different models on more than 750 three-day-long time series from the validation set. Each model is recursively applied to predict the temperature in all the zones for 288 steps, i.e., three days,[2] assuming knowledge of all the inputs, and compared to the true measured temperatures. The *error* of a model is then defined as its average performance over the three zones.

Section 5.1 starts with a discussion on the performance of the different models In Section 5.2, we then provide a visual and qualitative discussion of the NN-based model predictions[3] before numerically investigating their physical consistency more in-depth in Section 5.3. Finally, Section 5.4 concludes with a brief overview of the computational complexity associated with all the models. Altogether, this will allow us to understand the trade-offs between the physical consistency, accuracy, and computational complexity of the various data-driven building modeling methods examined in this work.

Note that the NN-based models were run with several random seeds, and, unless stated otherwise, the results discussed throughout this Section were obtained using the best-performing seed in each case. Remarkably, however, this does not impact our conclusions significantly since all the architectures proved to be robust to the choice of random seed, with standard deviations in the range of 0.01–0.04 and 0–0.2% for the MAE and MAPE, respectively, as detailed in Appendix E.

5.1. Performance analysis

The best performance of all the analyzed methods in terms of MAE and MAPE is reported in Table 1. As can be observed, all the proposed PCNN architectures attain state-of-the-art accuracy, both in terms of MAE and MAPE. They are followed by physically inconsistent black-box methods, especially the ones relying on very expressive NNs. As expected, the least expressive class of methods, gray-box models, performs the worst. Combining these results with the physical consistency of each method, we can conclude that the proposed PCNN architectures take the best out of both gray- and black-box methods, attaining state-of-the-art performance while respecting the underlying physical laws by construction without trade-off, making them ideal thermal building models.

While X-PCNNs achieve the best performance in Table 1, we suspect these results to be influenced by the analyzed case study. Indeed, the temperature dynamics in UMAR are strongly impacted by solar gains, which reduces the importance of energy exchanges between the zones. This might explain why it is possible to fit the overall building dynamics well even when independently training one model for each zone, as for X-PCNNs, and why the *post hoc* correction (10) only has a little impact on the final model performance. We suspect this required correction might have a stronger influence on multi-zone buildings where temperature dynamics are less impacted by weather conditions and more governed by energy exchanges between the zones, which might, in turn, decrease the quality of X-PCNNs.

Remarkably, enforcing the physical consistency of LSTMs, as in PCNNs, seems to improve their accuracy in this case study despite the introduced constraints. This confirms the ongoing trend in ML research to include prior knowledge in NN architectures. Even if it might intuitively seem that introducing structural constraints should hinder the expressiveness of LSTMs, these results suggest that it can on the contrary be helpful. Moreover, one can draw similar conclusions with the two residual models investigated in this work, with *Res-cons* clearly outperforming its physically inconsistent counterpart despite both models relying on the same linear basis. While these results are not reported here, ensuring the black-box modules to be independent of D^z, as mentioned in Remark 4, also increased the performance of X-PCNNs. Altogether, these results point towards performance benefits of grounding NN architectures in the underlying physics, ensuring that they learn meaningful solutions.

As shown in Table 1, the residual models (*Res* and *Res-cons*), which are conceptually close to PCNNs,[4] are unable to attain similar performance to PCNNs. This hints towards the benefits of learning all the parameters together in an end-to-end fashion instead of first identifying the linear part and then fitting the residual errors.

PCNNs are on average 30–35% and 17–22% more accurate than the other physically consistent methods, namely the *Linear* and *Res-cons* model, respectively. To visualize the error propagation of these methods over three days, their MAE is plotted in Fig. 4. This shows that the proposed PCNNs not only perform better on average, but along the entire prediction horizon, except during the first few hours, where the linear and residual model attain similar performance. The main reason behind this behavior is the warm start of PCNNs, which often gives erroneous first predictions, but they quickly make up for it and show much stronger performance in the long run. At the end of the horizon,

[2] Given a warm start of 3 h for the models based on LSTMs.

[3] The LSTM, PiNN, and PCNN architectures. Despite also being composed of an NN, residual models are not considered as *NN-based* models in this work since their main dynamics are still captured by the underlying linear model, and not the NN.

[4] Especially *Res-cons*, where the only difference in architecture comes from the solar irradiation processing.

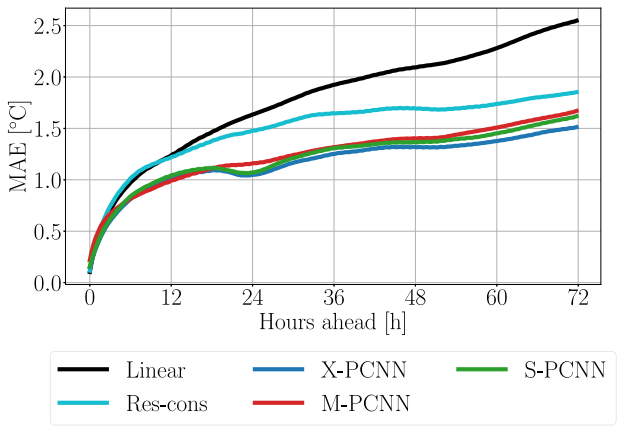


Fig. 4. MAE of all the physically consistent methods over the prediction horizon averaged over the three zones and the time series of three days in the validation data set.

PCNNs indeed show an error 34–41% and 10–18% lower than *Linear* and *Res-cons*, respectively, with the best performance again achieved by the X-PCNN.

The results of our case study suggest that the black-box modules of PCNNs are able to process the raw solar irradiation data and infer its impact on the zone temperatures. Indeed, they outperform gray-box models, which have access to engineered solar gains (Appendix C). Interestingly, since the impact of the sun is implicitly computed by their black-box modules, PCNNs could easily be applied to any building, even when shading comes into play, making the engineered preprocessing of solar data required for gray-box models much more complex. Furthermore, while the only nonlinear gains considered in this paper come from solar irradiation, the flexibility of the black-box module would also allow it to learn other gains, such as the ones stemming from occupants.

5.2. The necessity of physical consistency

Now that Section 5.1 established that PCNNs attain state-of-the-art performance in terms of accuracy, even outperforming pure black-box methods, let us visualize the behavior of one S-PCNN, one PiNN, and one LSTM for a given random seed in Fig. 5. To that end, the thermal power is turned off in Zone 1 and 2 and we examine the impact of heating (red), cooling (blue), or providing no power input (black) in Zone 3. Note that the heating pattern corresponds to the true power inputs measured in Zone 3 in March 2021, which we mirror to create the cooling pattern.

As one can immediately realize, following the laws of thermodynamics, heating or cooling Zone 3 increases, respectively decreases its temperature in the S-PCNN model. This effect is then propagated to the adjacent Zone 2, and later to Zone 1, impacting their temperatures even though they are neither heated nor cooled, illustrating the effect of Corollary 1 ensuring physically consistent predictions. Note that while only the temperature predictions of one S-PCNN are pictured in Fig. 5, similar effects were observed for X-PCNNs and M-PCNNs. This is expected since all of them share the same properties, i.e., the same physics-inspired module, to ensure they follow the criteria (1)–(3).

On the other hand, all power inputs lead to almost indistinguishable temperature predictions for the PiNN and LSTM. Despite achieving a very good fit of the data (Section 5.1), these models are hence obviously flawed and can be misleading in practical applications. We can sometimes even observe lower temperature predictions when heating is turned on than when the zones are cooled, a clear sign of physical inconsistency (see Appendix F for zoomed in results). This clearly exemplifies the issue of shortcut learning in the case of thermal modeling, where NNs manage to fit the data well without understanding the underlying physics.

Furthermore, this illustrates how PiNNs only *steer* the learning towards interesting solutions without providing any guarantees concerning the actual behavior of the model. In fact, trained PiNNs always gave very similar predictions to LSTMs in our experiments, as can also be seen in Fig. 5, hinting that modifying the loss function $\mathcal{L}_{PiNN}$ of the model did not have much impact on the final solution found despite the large λ used. While tuning this hyperparameter might lead to better results, it is a notoriously cumbersome task and would still never guarantee the physical consistency of the final model [64]. Thus, it was not considered in this work.

Very importantly, these results point out a somewhat counter-intuitive and often overlooked characteristic of NNs: contrary to physics-based models, a good fit to the data does not necessarily imply that the quality of the model is good. In our case, the PiNNs and LSTMs were indeed able to fit the data well without considering the impact of heating and cooling, i.e., solely mapping external conditions to building temperatures. One hence has to be careful when NNs are used to model physical systems and make sure the trained models do not simply find shortcuts to fit the data well without respecting the underlying physical laws. This calls for physically grounded architectures, such as PCNNs, for applications where the physical consistency of the model is critical.

We suspect that LSTMs and PiNNs were able to fit the data very well without considering the impact of heating and cooling because of the specific data used in this case study. First, windows cover the entire East facade of UMAR, rendering the building especially sensitive to solar gains and external weather conditions. Second, while different controllers have been applied during the collection period of the data set, all of them had the same objective of maintaining the building temperature in a comfortable range and hence reacted similarly to external conditions. Coupling these facts, it seems indeed plausible to accurately predict building temperatures solely based on external conditions and without considering heating and cooling inputs. In other words, we suspect the very expressive LSTMs and PiNNs to have learned the *closed-loop* response of the system instead of the expected *open-loop* one, hence implicitly accounting for the influence of power inputs instead of explicitly modeling their effect. This might explain how they found non-physical shortcuts modeling the evolution of inside temperatures well. Interestingly, the identified linear model also failed to capture any significant impact of heating and cooling, showing that it is also possible to fit this data well without accounting for these inputs but still following the underlying laws of physics (Appendix F).

Interestingly, *Res* did capture a much more significant impact of heating and cooling, but remains completely oblivious to the underlying physics, with cooling often resulting in higher temperatures than heating. This illustrates the need to also consider physical consistency when designing residual models, such as in the proposed *Res-cons* architecture. In general, all these results hence suggest that physical consistency should always be considered when dealing with NNs for physical systems.

Note that the identification of the ARX-KF model assigned a negative scaling parameter for the power input to Zone 2, for example, meaning that heating this zone will lead to lower temperatures. We could observe similar issues with the parameters of the classical ARX model, indeed confirming that ARX models might not be physically consistent in practice, as claimed in Section 4.3.

5.3. Numerical analysis of physical consistency

This section investigates the gradients of the predictions of NN-based models numerically, to strengthen the theoretical and visual claims in Table 1 and Fig. 5. Since gradients can be retrieved automatically through the `torch.autograd` module [61], it allows us to numerically assess if the models respect criteria (1) and (2), a necessary

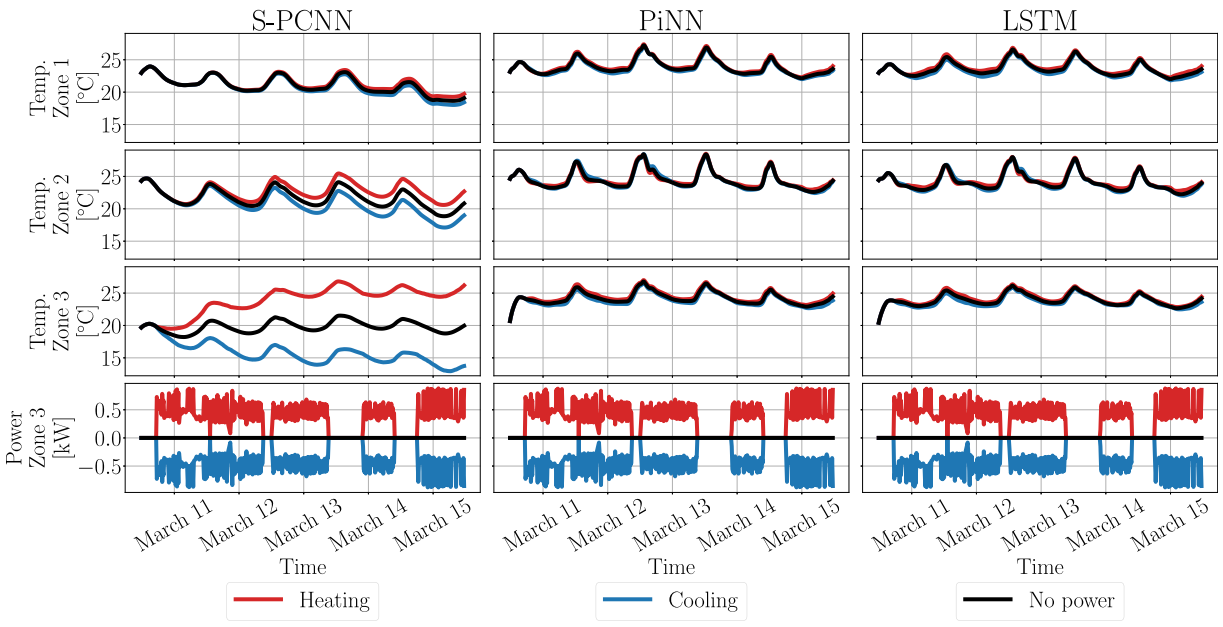


Fig. 5. Visualization of heat propagation for the proposed S-PCNN architecture on the left compared to a PiNN in the middle and an LSTM on the right. The bottom plots show the heating (red) and cooling (blue) patterns applied to Zone 3 while the power is turned off in Zone 1 and 2, compared to the situation when no power is applied (black). The other plots depict the corresponding temperature predictions of each model in each of the three zones.

condition to ensure physical consistency. Following Remark 10, we investigate the gradients of the temperature predictions at the end of the three-day-long horizon with respect to the power inputs and ambient temperatures observed at each time step. Note that this corresponds to the gradients used to steer the learning of PiNNs in (39), except for the X-PCNN, for which fewer gradients can be computed, as detailed in Appendix G. However, their magnitude does not have any physical meaning since NN-based models work with normalized data.

Overall, this gives us access to more than two million gradient values for each model, except the X-PCNN, with slightly over one million values, as computed in Appendix H. The resulting density histograms are shown in Fig. 6, where one can directly observe negative gradients only for the two black-box models not grounded in the underlying physics. In fact, penalizing negative gradients in $\mathcal{L}_{PiNN}$ decreased the magnitude of the PiNN gradients, steering them to zero, but did not change the proportion of negative ones. In other words, it did not improve the physical consistency of PiNNs since they still violate conditions (1) and (2) as often as classical LSTMs. Remarkably, the small magnitude of the PiNN and LSTM gradients corroborate what can be seen in Fig. 5, with very little impact of heating and cooling for these models. On the other hand, thanks to their physics-inspired module, the proposed PCNN architectures keep all the gradients that require positivity in $\mathbb{R}_+$ and with larger magnitudes, as desired and observed in Fig. 5 for the S-PCNN, providing a numerical argument supporting their physical consistency.

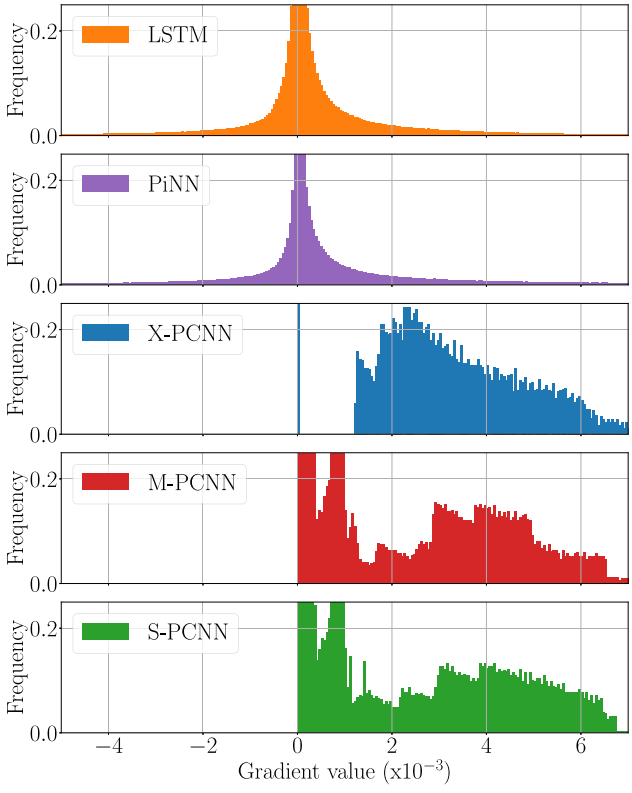


Fig. 6. Distribution of the gradients of the temperatures at the end of the prediction horizon with respect to power inputs and external temperatures observed along the horizon for the NN-based models.

5.4. Computational complexity

As final comparison metric between the models, Fig. 7 presents the time required by each model per training iteration. Importantly, these numbers are subject to implementation considerations and hence have to be taken with a grain of salt since we did not optimize the models. Nonetheless, all of them used the same backbone architecture, which allows relative comparisons, for example between the three proposed PCNNs, between the two residuals models, or between LSTMs and PiNNs. Note that the linear and ARX models are not considered here since their "training" procedure is very different: it does not require access to a GPU and does not rely on gradient descent

First, as expected, PiNNs take more time to run than classical LSTMs since each batch has to be forwarded and backwarded through the networks twice, once to compute the predictions used in $\mathcal{L}_{data}$

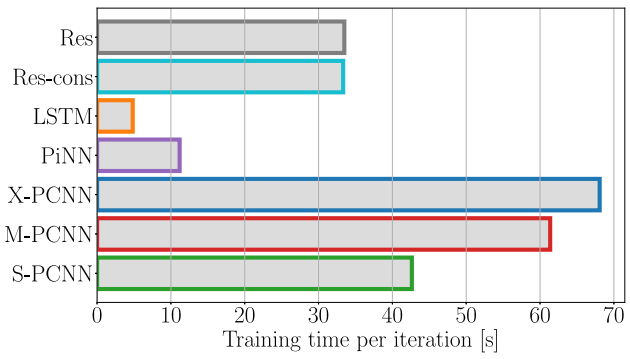


Fig. 7. Training time per iteration of the methods relying on a GPU.

and another time to calculate the gradients in $\mathcal{L}_{phys}$. Second, residual models need access to the predictions of the underlying linear model at each step to compute the residual errors before fitting them, which also entails a clear computational overhead compared to LSTMs. Finally, the PCNN architectures all require to compute both the black-box module output D_k and the physics-inspired module predictions E_k at each step k along the horizon, which also entails additional overhead on top of classical black-box models. Interestingly, this is comparable to what happens in residual models, explaining to some extent why the latter and S-PCNNs require similar amounts of resources.

Compared to S-PCNNs, M-PCNNs and X-PCNNs are significantly more computationally intensive. This intuitively follows from the shared black-box module of S-PCNNs reducing the number of parameters to fit compared to M-PCNNs. On the other hand, X-PCNNs require learning several models separately instead of everything together, which duplicates the computational overhead needed to create and move data to the GPU at each iteration and leads to an increased computational burden compared to M-PCNNs. Stemming from these remarks, we would expect these differences to grow if we were to apply PCNNs to larger buildings with more thermal zones.

Remark 11 (*Parallelizing X-PCNNs*)**.** The training times reported here correspond to the total time required to train each model for one iteration, i.e., the sum of training times of single-zone PCNNs in the case of X-PCNNs, to represent the total amount of computations needed. In practice, however, the different single-zone PCNNs can easily be trained in parallel since they are independent, which can significantly decrease the effective training time of X-PCNNs (dividing it approximately by three in our setting with three zones). This would make them the fastest multi-zone PCNNs to deploy but at the cost of additional computational complexity.

6. Conclusion

This work presented extensions of single-zone PCNNs to the multi-zone setting, thereby providing fully data-driven control-oriented multi-zone building thermal models. The main idea of PCNNs is to let a physics-inspired and a black-box module run in parallel, the former guaranteeing the compliance of the output with the underlying physical laws — the laws of thermodynamics in the case of building temperature modeling — and the latter capturing unknown nonlinear dynamics, typically relying on NNs.

The proposed multi-zone PCNNs respect the underlying physics by design and at all times despite requiring little engineering, contrary to classical physically consistent methods. On the other hand, they outperformed state-of-the-art black-box methods in terms of accuracy on a case study, hinting that the constrained structure introduced to ensure they follow some ground rules does not hinder their expressiveness. Our analyses showed little difference between S-, M-, and X-PCNNs in general, with S-PCNNs entailing the least computational complexity and X-PCNNs attaining the best accuracy on the analyzed case study. Remarkably, all of them showed significantly better performance than classical physically consistent data-driven methods, with accuracy improvements of 30–35% and 17–22% compared to a linear and a residual model, respectively. While these results were obtained on a specific building, the performance gap suggests that this trend would be observed for other applications. PCNNs should thus remain the best modeling choice in general, even if classical black-box methods might attain a better accuracy on different data sets.

Our investigations also illustrated a well-known pitfall of classical PiNNs and LSTMs, which can find shortcuts to fit the data well without respecting the underlying physical laws. This exemplifies the need to not solely consider the fit to the data as a measure of the quality of NNs but also ensure that their predictions make sense from a physical point of view. Our findings hence support the current trend to incorporate inductive biases, i.e., prior knowledge, in NNs to alleviate their infamous generalization issues, leading to more principled architectures like the proposed PCNNs.

In light of these results, PCNNs pave the way for NN-based methods that can simultaneously provide state-of-the-art performance *and* physical guarantees. Furthermore, while only solar irradiation measurements and time information were fed to the black-box module of PCNNs throughout this study, showcasing the ability of the proposed approach to handle highly nonlinear effects, other inputs could be integrated straightforwardly.

Thanks to their flexibility, we hence believe PCNNs to be an essential step towards the safe deployment of NNs in real-world applications, closing the sim2real gap of advanced control algorithms, and hope to spark an interest both in the building modeling community and beyond. It would indeed be interesting to investigate the capabilities of PCNNs to model different buildings, incorporate additional nonlinearities in their black-box modules or rules in the physics-inspired ones — for example leveraging Irreversible port-Hamiltonian dynamics [59] —, and tackle other complex physical systems.

CRediT authorship contribution statement

L. Di Natale: Conceptualization, Methodology, Software, Validation, Formal analysis, Data Curation, Visualization, Writing – original draft. **B. Svetozarevic:** Conceptualization, Methodology, Writing – review & editing, Supervision. **P. Heer:** Writing – review & editing, Resources, Funding acquisition. **C.N. Jones:** Conceptualization, Methodology, Writing – review & editing, Supervision.

Declaration of competing interest

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

Data availability

The data and code used are available on https://gitlab.nccr-automation.ch/loris.dinatale/multi-zone-pcnns.

Acknowledgments

This research was supported by the Swiss National Science Foundation under NCCR Automation, grant agreement 51NF40_180545, and in part by the Swiss Data Science Center, grant agreement C20-13.

Appendix A. Proofs of the main theoretical results

A.1. Proof of Proposition 1

The proof works by induction on i. Based on (20), we can immediately write, $\forall z, y \in \mathcal{B}$:

$$\frac{\partial \boldsymbol{T}^z_{k+j+1}}{\partial \boldsymbol{T}^y_{k+j}} = \begin{cases} 1 - b^z - \sum_{y \in \mathcal{N}(z)} c^{zy}, & \text{if } y = z, \\ c^{zy}, & \text{if } y \in \mathcal{N}(z), \\ 0, & \text{otherwise}, \end{cases} \tag{40}$$

where we used the definition of $\boldsymbol{\Delta T}$ in (12). By definition, if (22) and (23) hold, we hence get positive derivatives if $y = z$ or $y \in \mathcal{N}(z)$ and zeros for any other choice of y, satisfying (21) and completing the base case of the induction.

Let us now assume that:

$$\frac{\partial \boldsymbol{T}^x_{k+h}}{\partial \boldsymbol{T}^y_{k+j}} \geq 0, \ \forall y, x \in \mathcal{B}, \ \forall j < h < i, \tag{41}$$

with equality if and only if $y \notin \mathcal{N}^{(h-j)}(x)$, and show that the proposition holds for time step i. Since we know the temperature in zone z at time $k+i$ is potentially impacted by the temperature in the entire building at the previous step, we can decompose the partial derivative of interest as follows:

$$\frac{\partial \boldsymbol{T}^z_{k+i}}{\partial \boldsymbol{T}^y_{k+j}} = \sum_{x \in \mathcal{B}} \frac{\partial \boldsymbol{T}^z_{k+i}}{\partial \boldsymbol{T}^x_{k+i-1}} \frac{\partial \boldsymbol{T}^x_{k+i-1}}{\partial \boldsymbol{T}^y_{k+j}}, \tag{42}$$

for all $y, z \in \mathcal{B}$. Since (20) is time-invariant, we know that:

$$\frac{\partial \boldsymbol{T}^z_{k+i}}{\partial \boldsymbol{T}^x_{k+i-1}} = \frac{\partial \boldsymbol{T}^z_{k+j+1}}{\partial \boldsymbol{T}^x_{k+j}} \geq 0,$$

with equality if and only if $x \notin \mathcal{N}(z)$ by the base case of the induction (40) if (22) and (23) hold. Similarly, by the induction hypothesis (41), we know that:

$$\frac{\partial \boldsymbol{T}^x_{k+i-1}}{\partial \boldsymbol{T}^y_{k+j}} \geq 0, \ \forall y, x \in \mathcal{B},$$

with equality if and only if $y \notin \mathcal{N}^{(i-j-1)}(x)$. Putting the last two equations together, we see that:

$$\frac{\partial \boldsymbol{T}^z_{k+i}}{\partial \boldsymbol{T}^y_{k+j}} \geq 0,$$

with equality only if each term of the sum in Eq. (42) is zero. By the previous arguments, this means $y \notin \mathcal{N}^{(i-j-1)}(x)$ or $x \notin \mathcal{N}(z)$ for all zones x. This is equivalent to say that there is no path from y to z in $(i-j)$ steps, i.e., $y \notin \mathcal{N}^{(i-j)}(z)$, which concludes the inductive step.

A.2. Proof of Proposition 2

We start by noticing that $\forall y \in \mathcal{B}$, (20) implies:

$$\frac{\partial \boldsymbol{T}^y_{k+j+1}}{\partial \boldsymbol{u}^y_{k+j}} = \begin{cases} a^y_h, & \text{if } \boldsymbol{u}^y_{k+j} > 0, \\ a^y_c, & \text{if } \boldsymbol{u}^y_{k+j} < 0, \\ 0, & \text{otherwise}, \end{cases} \tag{43}$$

$$\frac{\partial \boldsymbol{T}^x_{k+j+1}}{\partial \boldsymbol{u}^y_{k+j}} = 0 \qquad \forall x \in \mathcal{B}, x \neq y, \tag{44}$$

$$\frac{\partial \boldsymbol{T}^y_{k+j+1}}{\partial \boldsymbol{T}^{out}_{k+j}} = b^y, \tag{45}$$

Note that this proves that (24) and (25) for the case $i = j+1$ if $a^y_h, a^y_c, b^y > 0$, $\forall y \in \mathcal{B}$. When $i > j+1$, Proposition 1 implies:

$$\frac{\partial \boldsymbol{T}^z_{k+i}}{\partial \boldsymbol{T}^y_{k+j+1}} \geq 0, \qquad \forall z, y \in \mathcal{B}, \ \forall 0 \leq j < i-1, \tag{46}$$

with equality if and only if $y \notin \mathcal{N}^{(i-j-1)}(z)$ if the conditions in (22) and (23) hold.

Relying on the fact that the temperatures at time $k+i$ are potentially influenced by the temperatures in the whole building at time $k+j+1$, we have:

$$\frac{\partial \boldsymbol{T}^z_{k+i}}{\partial \boldsymbol{u}^y_{k+j}} = \sum_{x \in \mathcal{B}} \frac{\partial \boldsymbol{T}^z_{k+i}}{\partial \boldsymbol{T}^x_{k+j+1}} \frac{\partial \boldsymbol{T}^x_{k+j+1}}{\partial \boldsymbol{u}^y_{k+j}}, \tag{47}$$

$$= \frac{\partial \boldsymbol{T}^z_{k+i}}{\partial \boldsymbol{T}^y_{k+j+1}} \frac{\partial \boldsymbol{T}^y_{k+j+1}}{\partial \boldsymbol{u}^y_{k+j}} \geq 0, \tag{48}$$

where the second equality follows from (44) and the inequality holds as long as (22) and (23) are respected and $a^y_h, a^y_c > 0$, $\forall y \in \mathcal{B}$. Furthermore, by Proposition 1, equality is only reached if $y \notin \mathcal{N}^{(i-j-1)}(z)$.

Similarly, we have:

$$\frac{\partial \boldsymbol{T}^z_{k+i}}{\partial \boldsymbol{T}^{out}_{k+j}} = \sum_{y \in \mathcal{B}} \frac{\partial \boldsymbol{T}^z_{k+i}}{\partial \boldsymbol{T}^y_{k+j+1}} \frac{\partial \boldsymbol{T}^y_{k+j+1}}{\partial \boldsymbol{T}^{out}_{k+j}}, \tag{49}$$

$$= \sum_{y \in \mathcal{B}} \frac{\partial \boldsymbol{T}^z_{k+i}}{\partial \boldsymbol{T}^y_{k+j+1}} b^y > 0, \tag{50}$$

where the strict inequality is respected as long as $b^y > 0$, $\forall y \in \mathcal{B}$. Indeed, since $z \in \mathcal{N}(z)$ by definition, Proposition 1 then implies that at least one of the terms in the sum is strictly positive, while the others are nonnegative.

Appendix B. Details on the data processing

Once the data had been subsampled and processed as in [39, App. C] and discarding the 23% of incomplete measurements, i.e. where at least the information from one sensor is missing, we were left with over 80,000 data points. Since the proposed PCNN architectures are based on NNs in our implementations (see Section 4.2), they are not able to handle missing values, which prompted us to create a data set of time series without missing values.

As we aimed to design models that are able to predict the temperature dynamics over three day-long horizons, we truncated each sequence to a maximum of three days, and separated the heating and cooling seasons. We allowed the time series to overlap each hour, i.e. each four steps, to increase the data efficiency of the approach. Finally, since we implemented a warm-start period of 3 h for all the models, we also made sure the last 3 h of data exist for each time series. To avoid very short time series, we also ensured they always span at least 12 h. Altogether, this allowed us to create more than 11,000 sequences of data without missing values, which were split in a training and a validation set with proportions 80%–20%, respectively denoted $\mathcal{D}_t$ and $\mathcal{D}_v$, and where $\mathcal{D}_t \cap \mathcal{D}_v = \emptyset$. For all NNs, the validation set is used to select the best set of weights along the training procedure.

Appendix C. Solar irradiation preprocessing

To compute the solar irradiation on the windows of a thermal zone z from the measured irradiation on a horizontal surface Q^{sun}, we rely on the altitude and azimuth angles, respectively ϕ and θ, of the sun. The former captures the elevation of the sun above the horizon while the latter represents its deviation from the north, in the clockwise direction.

First, using the altitude of the sun and basic trigonometry, one can easily show that the measured irradiation on a horizontal surface corresponds to $Q^{sun} = I \sin\phi$, where I is the global solar irradiation. Similarly, we know that the irradiation on a vertical surface following the sun, i.e., tracking its azimuth angle to stay perpendicular to the incoming rays, can be computed as $I^{vert} = I \cos\phi$. We can hence write the solar irradiation on a vertical surface following the sun as follows:

$$I^{vert} = Q^{sun} \frac{\cos\phi}{\sin\phi}. \tag{51}$$

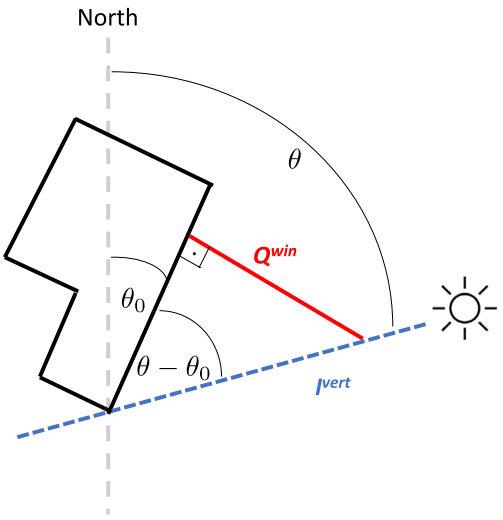


Fig. 8. Sketch of the azimuth angles used to compute the solar irradiation on the windows of a building from the irradiation on a fixed vertical surface.

Table 2
MAE and MAPE of the methods investigated in this work, average over the three thermal zone, the three-day long horizon, and more than 750 time series.

Model	MAE	MAPE
LSTM	1.33 ± 0.04	$5.7\% \pm 0.2\%$
PiNN	1.38 ± 0.01	$5.9\% \pm 0.1\%$
***X-PCNN* (Ours)**	$\mathbf{1.18 \pm 0.01}$	$\mathbf{5.0\% \pm 0.0\%}$
M-PCNN (Ours)	1.26 ± 0.01	$5.4\% \pm 0.0\%$
S-PCNN (Ours)	1.27 ± 0.04	$5.4\% \pm 0.2\%$

Since building facades and windows have a fixed orientation in practice and do not follow the sun azimuth, we again use basic trigonometry to compute the irradiation on a north–south aligned surface facing east as $I^{vert} \sin\theta$. Finally, if the facade is not exactly facing east, we also need to account for its own "azimuth" θ_0, i.e.,how much it is rotated clockwise starting from an east-facing position (Fig. 8), which leads to:

$$Q^{win} = I^{vert} \sin(\theta - \theta_0) \tag{52}$$

$$= Q^{sun} \frac{\cos\phi}{\sin\phi} \sin(\theta - \theta_0). \tag{53}$$

Once this has be done for each zone z, we can populate the required vector $\boldsymbol{Q}^{win}$ used by gray-box architectures in this work.

As one can readily observed, this processing only requires access to the elevation and azimuth angles of the sun, and to the orientation of the facade of interest. Furthermore, both solar angles solely depend on the geographical position of the building, i.e., its latitude and longitude, and the time at which the measurement was taken. The position and orientation of a building can easily be found on plans or Google Maps, and we used the `Astral` Python library (https://astral.readthedocs.io/en/latest/) to compute the solar angles corresponding to each time step in our data.

Note that, while this processing works very well for unobstructed facades when its orientation is known, it cannot be used when for example other buildings or trees exist in front of the windows and create shading patterns. In that case, one has to rely on architectures which are able to automatically process horizontal solar irradiation measurements depending on time information, such as the LSTMs used in the black-box module of PCNNs. Nonetheless, we can use it in this paper since UMAR is not obstructed, leading to a very efficient computation of the true solar irradiation patterns on the windows of each zone.

Appendix D. Linear model identification

As is classically done in linear system identification, we first used the least squares method to find the parameters a_h^z, a_c^z, b^z, c^{zy}, e^z best fitting the training data for each thermal zone z and neighboring zone $y \in \mathcal{N}(z)$, such as in [39, App. A.2]. However, ensuring none of these parameters is negative, which is necessary to respect the underlying physics, produced $c^{23} = 0$. This is clearly not physically meaningful, as it would mean there is no heat transfer from Zone 3 to Zone 2. Consequently, we also implemented a BO framework, relying on the `bayes_opt` Python library [65]. This allowed us to extensively search for the best physically consistent parameters for each zone over a five-step prediction horizon, constraining all the parameters to be positive, for 2300 iterations starting with 200 random initial points.

Appendix E. Impact of the random seed

The mean performance of the five NN-based model architectures, as well as the corresponding standard deviation, is presented in Table 2. While the LSTM and S-PCNN models were run on five seeds due to their slightly higher sensitivity, the other results were obtained over three seeds leading to very consistent performance. As in the original PCNN paper [39], this hints at the robustness of the proposed approach, which does not seem significantly impacted by the random seed, or at least similarly to classical NN models. On this case study, the proposed X-PCNN and M-PCNN seem more robust to the choice of random seed than the S-PCNN. However, as shown in Table 1, the latter sometimes outperforms M-PCNNs. Nonetheless, overall, X-PCNNs seem to have the upper hand, always attaining state-of-the-art performance even under different random seeds.

Appendix F. Visualization of predictions

To complement Fig. 5, the same experiment was carried on with the linear model, and the corresponding predictions can be found in Fig. 9 (left). Note that each subplot is using a custom scale to better visualize the impact of different power inputs. We additionally shaded physically inconsistent behaviors in each subplot in gray, i.e., whenever the predicted temperature when cooling is applied is higher than when heating is applied or no power input is used, or when the temperature when heating is applied is lower than when no power is used. This confirms that the identified linear model failed to fully capture the impact of heating and cooling but still behaves in a physically consistent manner, e.g., with heating leading to higher temperatures than cooling, similar to the behavior that can be observed for the S-PCNN in Fig. 5. On the other hand, both the PiNN and LSTM show inconsistent behaviors, especially in Zone 2 around the beginning of the prediction horizon.

Appendix G. X-PCNN gradients

In the case of X-PCNNs, at inference time, we use each single-zone PCNN to predict the next temperature in the corresponding zone. The new temperatures in the building are then updated in the data of all the single-zone PCNNs so they can predict the next step. This is required because the single-zone PCNNs cannot evolve independently over the prediction horizon since they depend on temperatures in neighboring zones at teach step. However, overwriting the data at each step breaks the automatic backpropagation of Python, and we cannot automatically compute the gradient of the temperature in zone z with respect to power inputs or temperatures in another zone z' without implementation overhead. We can only retrieve gradients with respect to each single-zone PCNN's inputs, i.e., the power u^z, and the ambient temperature. Note that, intuitively, these available gradients are expected to be larger in magnitude than the gradients with respect to power inputs in other zones since they have a direct impact on

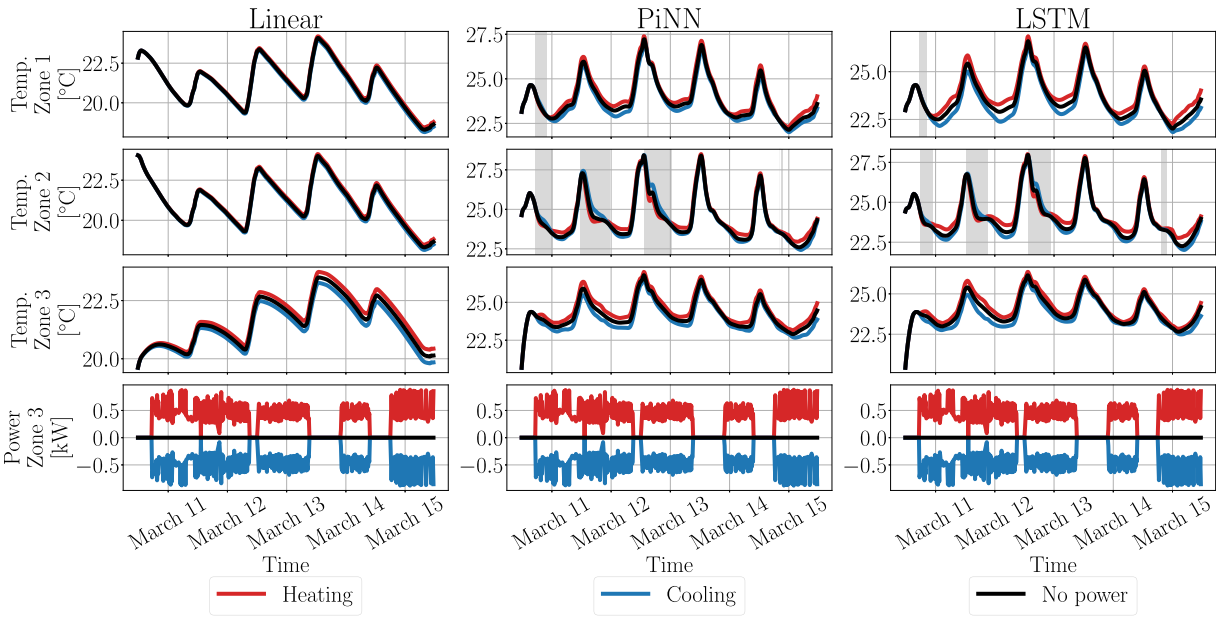


Fig. 9. Visualization of heat propagation for the linear on the left compared to a PiNN in the middle and an LSTM on the right. The bottom plots show the heating (red) and cooling (blue) patterns applied to Zone 3 while the power is turned off in Zone 1 and 2, compared to the situation when no power is applied (black). The other plots depict the corresponding temperature predictions of each model in each of the three zones. Gray-shaded area mark physical inconsistencies of the PiNN ans LSTM.

the zone of interest. This explains the absence of low gradient values ($<10^{-3}$) in Fig. 6 for X-PCNNs compared to M- and S-PCNNs. Even if we can only compute parts of the gradients automatically, we still show them in Fig. 6 for reference. Note that as we already know X-PCNNs are physically consistent since they satisfies the criteria of Corollary 1, these implementation considerations do not put the architecture in jeopardy.

Appendix H. Number of numerical gradient values

The numerical investigation of NN-based model gradients in Section 5.3 is carried out on the validation data set of more than 750 three-day long sequences (288 steps). Following Remark 10, for each of the three zones, we compute the gradients of its last temperature predictions with respect to power inputs in all the zones (3 values) and the ambient temperature (1 value) at each step, giving rise to more than $750\times3\times288\times(3+1) = 2{,}592{,}000$ values. In the case of X-PCNNs, we only have access to half of these values since we do not compute gradients with respect to power inputs in other zones (Appendix G), which still leaves us with more than 1 million values.

References

[1] International Energy Agency (IEA). Tracking buildings 2020. 2020, https://www.iea.org/reports/tracking-buildings-2020, Accessed: 2021.05.28.

[2] Commission E, for Energy D-G, Kranzl L, Fallahnejad M, Büchele R, Müller A, Hummel M, Fleiter T, Mandel T, Bagheri M, Deac G, Bernath C, Miosga J, Kiefer C, Fragoso J, Braungardt S, Bürger V, Spasova D, Viegand J, Naeraa R, Forthuber S. Renewable space heating under the revised Renewable Energy Directive : ENER/C1/2018-494 : final report. Publications Office; 2022.

[3] Westermann P, Evins R. Surrogate modelling for sustainable building design–a review. Energy Build 2019;198:170–86.

[4] Deb C, Schlueter A. Review of data-driven energy modelling techniques for building retrofit. Renew Sustain Energy Rev 2021;144:110990.

[5] Svetozarevic B, Baumann C, Muntwiler S, Di Natale L, Zeilinger MN, Heer P. Data-driven control of room temperature and bidirectional EV charging using deep reinforcement learning: Simulations and experiments. Appl Energy 2022;307:118127.

[6] Lei Y, Zhan S, Ono E, Peng Y, Zhang Z, Hasama T, Chong A. A practical deep reinforcement learning framework for multivariate occupant-centric control in buildings. Appl Energy 2022;324:119742.

[7] Höfer S, Bekris K, Handa A, Gamboa JC, Mozifian M, Golemo F, Atkeson C, Fox D, Goldberg K, Leonard J, et al. Sim2Real in robotics and automation: Applications and challenges. IEEE Trans Autom Sci Eng 2021;18(2):398–400.

[8] Kadian A, Truong J, Gokaslan A, Clegg A, Wijmans E, Lee S, Savva M, Chernova S, Batra D. Sim2real predictivity: Does evaluation in simulation predict real-world performance? IEEE Robot Autom Lett 2020;5(4):6670–7.

[9] Drgoňa J, Arroyo J, Figueroa IC, Blum D, Arendt K, Kim D, Ollé EP, Oravec J, Wetter M, Vrabie DL, et al. All you need to know about model predictive control for buildings. Annu Rev Control 2020;50:190–232.

[10] Di Natale L, Svetozarevic B, Heer P, Jones CN. Near-optimal deep reinforcement learning policies from data for zone temperature control. 2022, arXiv preprint arXiv:2203.05434.

[11] Yang T, Zhao L, Li W, Wu J, Zomaya AY. Towards healthy and cost-effective indoor environment management in smart homes: A deep reinforcement learning approach. Appl Energy 2021;300:117335.

[12] Dawood SM, Hatami A, Homod RZ. Trade-off decisions in a novel deep reinforcement learning for energy savings in HVAC systems. J Build Perform Simul 2022;15(6):809–31.

[13] Crawley DB, Lawrie LK, Winkelmann FC, Buhl WF, Huang YJ, Pedersen CO, Strand RK, Liesen RJ, Fisher DE, Witte MJ, et al. EnergyPlus: creating a new-generation building energy simulation program. Energy Build 2001;33(4):319–31. http://dx.doi.org/10.1016/S0378--7788(00)00114--6.

[14] Wetter M, Haugstetter C. Modelica versus TRNSYS–A comparison between an equation-based and a procedural modeling language for building energy simulation. In: Proceedings of SimBuild, Vol. 2. 2006.

[15] Mazzeo D, Matera N, Cornaro C, Oliveti G, Romagnoni P, De Santoli L. EnergyPlus, IDA ICE and TRNSYS predictive simulation accuracy for building thermal behaviour evaluation by using an experimental campaign in solar test boxes with and without a PCM module. Energy Build 2020;212:109812. http://dx.doi.org/10.1016/j.enbuild.2020.109812.

[16] Harb H, Boyanov N, Hernandez L, Streblow R, Müller D. Development and validation of grey-box models for forecasting the thermal response of occupied buildings. Energy Build 2016;117:199–207.

[17] Zhang Z, Chong A, Pan Y, Zhang C, Lam KP. Whole building energy model for HVAC optimal control: A practical framework based on deep reinforcement learning. Energy Build 2019;199:472–90.

[18] Ascione F, Bianco N, De Stasio C, Mauro GM, Vanoli GP. Artificial neural networks to predict energy performance and retrofit scenarios for any member of a building category: A novel approach. Energy 2017;118:999–1017.

[19] Bourdeau M, qiang Zhai X, Nefzaoui E, Guo X, Chatellier P. Modeling and forecasting building energy consumption: A review of data-driven techniques. Sustainable Cities Soc 2019;48:101533. http://dx.doi.org/10.1016/j.scs.2019.101533.

[20] Foucquier A, Robert S, Suard F, Stéphan L, Jay A. State of the art in building modelling and energy performances prediction: A review. Renew Sustain Energy Rev 2013;23:272–88.

[21] Maasoumy M, Razmara M, Shahbakhti M, Vincentelli AS. Handling model uncertainty in model predictive control for energy efficient buildings. Energy Build 2014;77:377–92.

[22] Maasoumy M, Razmara M, Shahbakhti M, Vincentelli AS. Selecting building predictive control based on model uncertainty. In: 2014 American control conference. IEEE; 2014, p. 404–11.

[23] Li Y, O'Neill Z, Zhang L, Chen J, Im P, DeGraw J. Grey-box modeling and application for building energy simulations-A critical review. Renew Sustain Energy Rev 2021;146:111174.

[24] Arroyo J, Spiessens F, Helsen L. Identification of multi-zone grey-box building models for use in model predictive control. J Build Perform Simul 2020;13(4):472–86.

[25] Lin Y, Middelkoop T, Barooah P. Issues in identification of control-oriented thermal models of zones in multi-zone buildings. In: 2012 IEEE 51st IEEE conference on decision and control. CDC, IEEE; 2012, p. 6932–7.

[26] Shamsi MH, Ali U, Mangina E, O'Donnell J. Feature assessment frameworks to evaluate reduced-order grey-box building energy models. Appl Energy 2021;298:117174. http://dx.doi.org/10.1016/j.apenergy.2021.117174.

[27] Shamsi MH, Ali U, O'Donnell J. A generalization approach for reduced order modelling of commercial buildings. J Build Perform Simul 2019;12(6):729–44. http://dx.doi.org/10.1080/19401493.2019.1641554.

[28] Berthou T, Stabat P, Salvazet R, Marchio D. Development and validation of a gray box model to predict thermal behavior of occupied office buildings. Energy Build 2014;74:91–100. http://dx.doi.org/10.1016/j.enbuild.2014.01.038.

[29] Sheng F, Jia L. Short-term load forecasting based on SARIMAX-LSTM. In: 2020 5th international conference on power and renewable energy. ICPRE, IEEE; 2020, p. 90–4.

[30] Li X, Wen J. Review of building energy modeling for control and operation. Renew Sustain Energy Rev 2014;37:517–37.

[31] Royer S, Thil S, Talbert T. Towards a generic procedure for modeling buildings and their thermal zones. In: 2016 IEEE 16th international conference on environment and electrical engineering. EEEIC, IEEE; 2016, p. 1–6. http://dx.doi.org/10.1109/EEEIC.2016.7555567.

[32] Tien PW, Wei S, Darkwa J, Wood C, Calautit JK. Machine learning and deep learning methods for enhancing building energy efficiency and indoor environmental quality–A review. Energy AI 2022;100198.

[33] Deb C, Zhang F, Yang J, Lee SE, Shah KW. A review on time series forecasting techniques for building energy consumption. Renew Sustain Energy Rev 2017;74:902–24.

[34] Abbasimehr H, Paki R. Improving time series forecasting using LSTM and attention models. J Ambient Intell Humaniz Comput 2022;13(1):673–91.

[35] Zou Z, Yu X, Ergan S. Towards optimal control of air handling units using deep reinforcement learning and recurrent neural network. Build Environ 2020;168:106535.

[36] Delcroix B, Ny JL, Bernier M, Azam M, Qu B, Venne J-S. Autoregressive neural networks with exogenous variables for indoor temperature prediction in buildings. In: Building simulation, Vol. 14. Springer; 2021, p. 165–78.

[37] Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R. Intriguing properties of neural networks. 2013, arXiv preprint arXiv:1312.6199.

[38] D'Amour A, Heller K, Moldovan D, Adlam B, Alipanahi B, Beutel A, Chen C, Deaton J, Eisenstein J, Hoffman MD, et al. Underspecification presents challenges for credibility in modern machine learning. 2020, arXiv preprint arXiv:2011.03395.

[39] Di Natale L, Svetozarevic B, Heer P, Jones CN. Physically consistent neural networks for building thermal modeling: theory and analysis. Appl Energy 2022;325:119806.

[40] Geirhos R, Jacobsen J-H, Michaelis C, Zemel R, Brendel W, Bethge M, Wichmann FA. Shortcut learning in deep neural networks. Nat Mach Intell 2020;2(11):665–73.

[41] Beery S, Van Horn G, Perona P. Recognition in terra incognita. In: Proceedings of the European conference on computer vision. ECCV, 2018, p. 456–73.

[42] Rosenfeld A, Zemel R, Tsotsos JK. The elephant in the room. 2018, arXiv preprint arXiv:1808.03305.

[43] Heuer H, Monz C, Smeulders AW. Generating captions without looking beyond objects. 2016, arXiv preprint arXiv:1610.03708.

[44] Zech JR, Badgeley MA, Liu M, Costa AB, Titano JJ, Oermann EK. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: a cross-sectional study. PLoS Med 2018;15(11):e1002683.

[45] Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L. Physics-informed machine learning. Nat Rev Phys 2021;3(6):422–40.

[46] Daw A, Karpatne A, Watkins W, Read J, Kumar V. Physics-guided neural networks (pgnn): An application in lake temperature modeling. 2017, arXiv preprint arXiv:1710.11431.

[47] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.

[48] Greydanus S, Dzamba M, Yosinski J. Hamiltonian neural networks. Adv Neural Inf Process Syst 2019;32.

[49] Lutter M, Ritter C, Peters J. Deep lagrangian networks: Using physics as model prior for deep learning. 2019, arXiv preprint arXiv:1907.04490.

[50] Cranmer M, Greydanus S, Hoyer S, Battaglia P, Spergel D, Ho S. Lagrangian neural networks. 2020, arXiv preprint arXiv:2003.04630.

[51] Hendriks J, Jidling C, Wills A, Schön T. Linearly constrained neural networks. 2020, arXiv preprint arXiv:2002.01600.

[52] LeCun Y, Kavukcuoglu K, Farabet C. Convolutional networks and applications in vision. In: Proceedings of 2010 IEEE international symposium on circuits and systems. IEEE; 2010, p. 253–6.

[53] Karpathy A, Johnson J, Fei-Fei L. Visualizing and understanding recurrent networks. 2015, arXiv preprint arXiv:1506.02078.

[54] Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. The graph neural network model. IEEE Trans Neural Netw 2008;20(1):61–80.

[55] Gokhale G, Claessens B, Develder C. Physics informed neural networks for control oriented thermal modeling of buildings. Appl Energy 2022;314:118852.

[56] Nagarathinam S, Chati YS, Venkat MP, Vasan A. PACMAN: physics-aware control MANager for HVAC. In: Proceedings of the 9th ACM international conference on systems for energy-efficient buildings, cities, and transportation. 2022, p. 11–20.

[57] Drgoňa J, Tuor AR, Chandan V, Vrabie DL. Physics-constrained deep learning of multi-zone building thermal dynamics. Energy Build 2021;243:110992. http://dx.doi.org/10.1016/j.enbuild.2021.110992.

[58] Chen Y, Yang Q, Chen Z, Yan C, Zeng S, Dai M. Physics-informed neural networks for building thermal modeling and demand response control. Build Environ 2023;110149.

[59] Zakwan M, Di Natale L, Svetozarevic B, Heer P, Jones CN, Trecate GF. Physically consistent neural ODEs for learning multi-physics systems. 2022, arXiv preprint arXiv:2211.06130.

[60] Empa. NEST. 2021, https://www.empa.ch/web/nest/overview. Accessed: 01.10.2022.

[61] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S. PyTorch: An imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché Buc F, Fox E, Garnett R, editors. Advances in neural information processing systems 32. Curran Associates, Inc.; 2019, p. 8024–35.

[62] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12:2825–30.

[63] Seabold S, Perktold J. statsmodels: Econometric and statistical modeling with python. In: 9th Python in science conference. 2010.

[64] Faroughi SA, Pawar N, Fernandes C, Das S, Kalantari NK, Mahjour SK. Physics-guided, physics-informed, and physics-encoded neural networks in scientific computing. 2022, arXiv preprint arXiv:2211.07377.

[65] Nogueira F. Bayesian Optimization: Open source constrained global optimization tool for Python. 2014.