Applied Energy 325 (2022) 119806

Contents lists available at ScienceDirect

Applied Energy

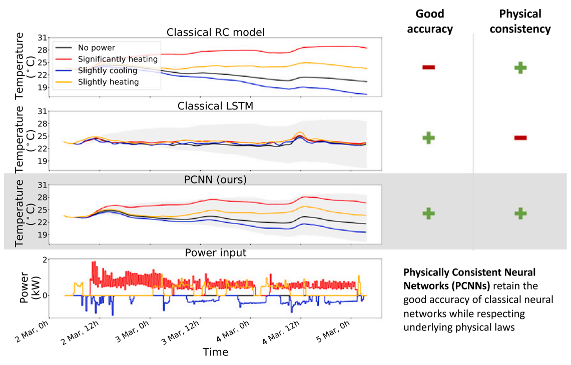
journal homepage: www.elsevier.com/locate/apenergy

Physically Consistent Neural Networks for building thermal modeling: Theory and analysis

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, Swiss Federal Institute of Technology Lausanne (EPFL), 1015 Lausanne, Switzerland

GRAPHICAL ABSTRACT



Physically Consistent Neural Networks (PCNNs) retain the good accuracy of classical neural networks while respecting underlying physical laws

ARTICLE INFO

Dataset link: https://gitlab.nccr-automation.ch/loris.dinatale/pcnn

Keywords:
Neural Networks
Physical consistency
Prior knowledge
Building models
Deep Learning

ABSTRACT

Due to their high energy intensity, buildings play a major role in the current worldwide energy transition. Building models are ubiquitous since they are needed at each stage of the life of buildings, i.e. for design, retrofitting, and control operations. Classical white-box models, based on physical equations, are bound to follow the laws of physics but the specific design of their underlying structure might hinder their expressiveness and hence their accuracy. On the other hand, black-box models are better suited to capture nonlinear building dynamics and thus can often achieve better accuracy, but they require a lot of data and might not follow the laws of physics, a problem that is particularly common for neural network (NN) models. To counter this known generalization issue, physics-informed NNs have recently been introduced, where researchers introduce prior knowledge in the structure of NNs to ground them in known underlying physical laws and avoid classical NN generalization issues.

In this work, we present a novel physics-informed NN architecture, dubbed Physically Consistent NN (PCNN), which only requires past operational data and no engineering overhead, including prior knowledge in a linear module running in parallel to a classical NN. We formally prove that such networks are physically consistent – by design and even on unseen data – with respect to different control inputs and temperatures outside and in neighboring zones. We demonstrate their performance on a case study, where the PCNN attains an accuracy up to 40% better than a classical physics-based resistance-capacitance model on 3-day long prediction horizons. Furthermore, despite their constrained structure, PCNNs attain similar performance to classical NNs on the validation data, overfitting the training data less and retaining high expressiveness to tackle the generalization issue.

* 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.2022.119806
Received 6 December 2021; Received in revised form 11 July 2022; Accepted 6 August 2022
Available online 19 August 2022

Nomenclature

PCNN variables

D	Unforced dynamics
E	Energy accumulator
P	Power
Q^{sun}	Solar gains
T	Temperature of the modeled zone
T^{neigh}	Temperature of the neighboring zone
T^{out}	Outside temperature
T^{w}	Water temperature of the heating system
a	Heating effect scaling parameter
b	Heat losses to the outside scaling parameter
c	Heat losses to the neighboring zone scaling parameter
d	Cooling effect scaling parameter
$\dot{m}$	Water mass flow rate in a radiator
u	Control inputs
x	Inputs to the black-box module

Gray-box model variables

ξ	Disturbance model
u	Controllable inputs
w^1, w^2	Uncontrollable inputs
z	State of the system

1. Introduction

Buildings consume 30% of global end-use energy, producing 28% of the world's Green House Gas (GHG) emissions related to energy according to the IEA [1], and those proportions rise to 40% of the total energy usage and 36% of the total GHG in the European Union (EU) [2]. Space heating and cooling have a major impact, with heating alone being responsible for 64% of household energy consumption in the EU [3]. To follow the Paris Agreement pledges to limit global warming to well below 2 °C [4], there is thus a need to decrease the energy intensity of the building sector.

1.1. The importance of building models

There are three technology-driven ways to attain this decarbonization objective: through better designs, retrofits, or improved operations of buildings. In all cases, models play a central role, either to find the best design [5], the most effective refurbishment [6], or to learn intelligent controllers to replace poorly performing rule-based controllers [7].

Modeling buildings is a challenging task in general since inside temperatures, air quality, or visual comfort, among others, all depend on highly stochastic exogenous factors mainly driven by the weather and the behavior of the occupants [8–11]. Additionally, the introduction of solar panels, heat pumps, battery storage, electric vehicles, and other new technologies makes it harder to model building operations as a whole and calls for scalable and flexible methods [12].

Most of the existing models are focused on commercial buildings [8] and study single-step predictors [13,14], which work adequately for energy consumption predictions in design or retrofitting applications. On the other hand, in this work, we design control-oriented thermal models for a residential case study that could, for example, be used in to learn Reinforcement Learning (RL) control policies. This calls for short-term multi-step temperature predictions, to be able to minimize energy consumption while maintaining the comfort of the occupants. Indeed, while RL can generally be applied in a model-free fashion, the data-inefficiency of RL algorithms [15] and the slow dynamics of buildings often require agents to be trained over thousands of days of data [16,17], which is not feasible in practice.

While classically engineered physics-based models still dominate the field, researchers recently started to leverage the growing amount of available data to design data-driven building models. Such models generally perform better, are more flexible, and rely on less technical knowledge, but require a lot of past data to be trained on and lack generalization guarantees outside of the training data [13]. This is particularly true for models based on Neural Networks (NNs), which can be very data-inefficient and fail when new inputs they were not trained on are fed to them [18], which is known as their *generalization issue*. In particular, there are no guarantees that a classical NN follows the underlying physics, e.g. the laws of thermodynamics in the case of thermal modeling. This is however critical for control-oriented applications , e.g. to ensure the RL agents trained on these models capture the impact of heating and cooling correctly.

1.2. The generalization issue of neural networks

Originally spotted by Szegedy et al. [19], the generalization issue of NNs led to the field of adversarial examples, where researchers aim to find input perturbations that fool NNs, showing how brittle their predictions can be [20,21], even when only little noise is applied to the input.

To circumvent this generalization issue, researchers often rely on better sets of data that cover the entire spectrum of inputs and allow NNs to react to any situation. This requires vast amounts of resources and is only possible in fields where a significant amount of data is available, such as for tasks related to natural language processing [22] or images [23]. Additionally, to ensure some level of generalization, practitioners typically separate the data into training and validation sets, the former being used to train the network and the latter to assess its performance on unseen data to avoid *overfitting* the training data [24]. However, classical NNs cannot be robust to input modifications that do not exist in the entire data set.

In the case of building thermal models, even if several years of data are available, one will always face an input coverage problem. Indeed, buildings are usually inhabited and operated in a typical fashion to maintain a comfortable temperature — heating when it gets cold in winter and cooling when it gets hot in summer. Most data sets are hence inherently incomplete and we cannot hope to learn robust NNs that grasp the effect of heating in summer, for example. When predicting the evolution of the temperature over long horizons of several days, classical NNs might therefore fail to capture the underlying physics, i.e. the impact of heating and cooling on the temperature . This is illustrated in Fig. 1, where one can compare the temperature predictions of a classical physics-based resistance-capacitance (RC) model, a classical Long Short-Term Memory network (LSTM), and a Physically Consistent NN (PCNN) proposed in this work under different heating and cooling power inputs. Interestingly, the LSTM achieves a superior accuracy than both other models on the training data, overfitting it, but clearly fails to capture the impact of heating and cooling.

1.3. Introducing physics-based prior knowledge

In general, classical NNs suffer from *underspecification*, as reported in a large-scale study from Google [25]. As a countermeasure, we should find ways to include prior knowledge, typically about the underlying laws of physics, into NNs to facilitate their training and improve their performance. This trend already began several years ago with the emergence of physics-guided machine learning [26] and the creation of specific network structures that represent known physical systems [18, 27,28]. In such NNs, physics can, for example, be introduced directly in the structure of the network or through custom loss functions, among

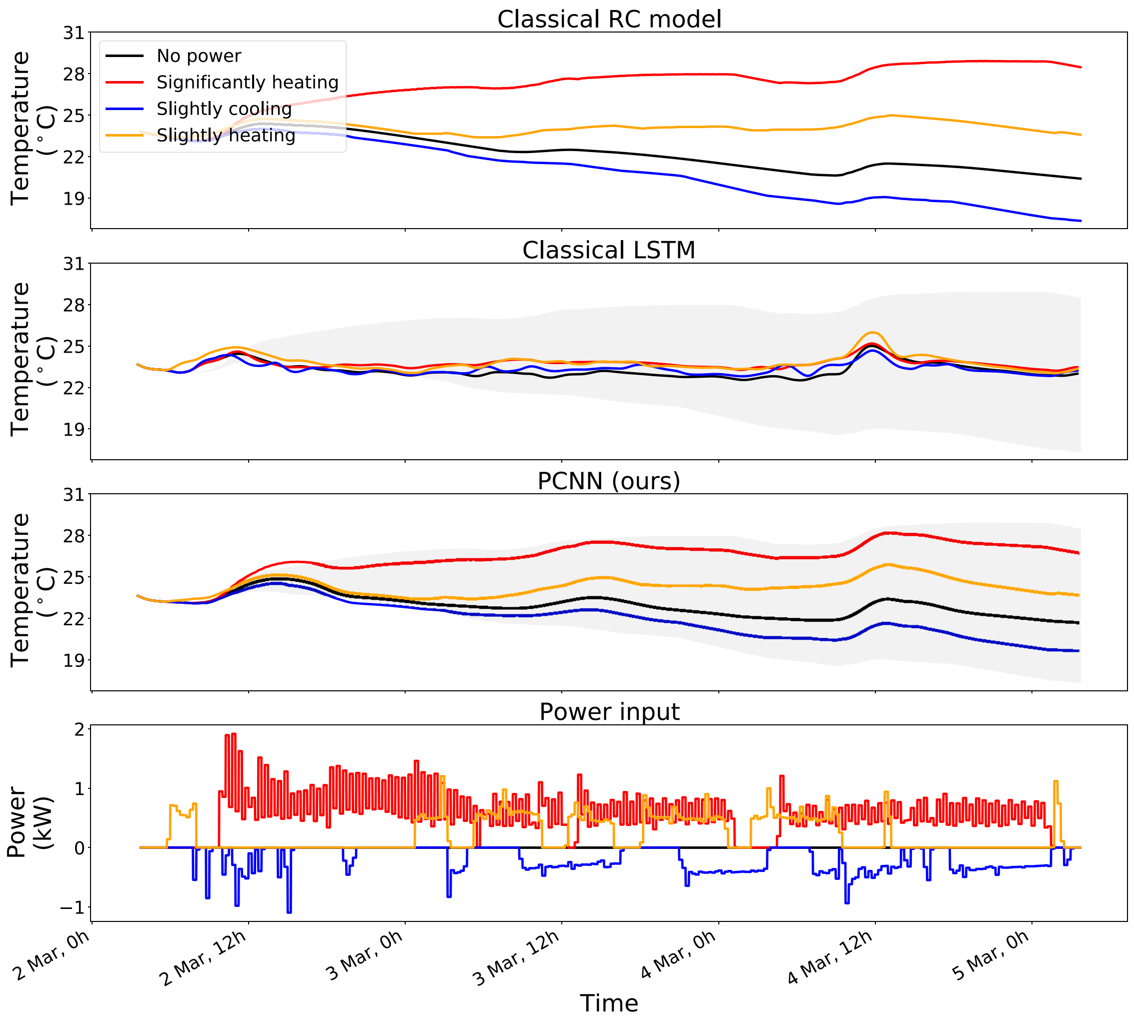


Fig. 1. Temperature predictions of the RC model and the proposed PCNN detailed and analyzed in Section 5 compared to a classical LSTM under different control inputs. The gray-shaded areas represent the span of the RC model predictions to provide a visual comparison with both black-box methods. While the LSTM presents a lower training error than the PCNN (see Section 5), indicating a good fit to the data, it does not capture the impact of the different heating/cooling powers applied to the system, e.g. predicting higher temperatures when cooling is on than when heating is. The specific structure of PCNNs introduced in Section 3, on the other hand, allows them to retain physical consistency, similarly to classical physics-based models, while improving the prediction accuracy (see Section 5.2)..

others [29]. In this paper, we refer to these models as Physics-informed Neural Networks (PiNNs).

To the best of the authors' knowledge, Drgoňa et al. [30] were the first to use PiNNs as control-oriented building models, but they did not provide theoretical guarantees of their models following the underlying physics, except for the hard-encoded dissipativity. Furthermore, the performance of their models, which remarkably work in the multi-zone setting, was not benchmarked against classical methods. Concurrently to our work, Gokhale et al. developed another PiNN structure for control-oriented building modeling, but they modified the loss function of their NNs and not their architecture [31], contrary to PCNNs. Finally, while not relying on PiNNs, we want to mention here the recent work of Bünning et al. on physics-inspired linear regression for buildings [32], which is philosophically related to the general efforts to introduce physical priors in otherwise black-box models.

1.4. Contribution

To tackle the aforementioned generalization issues of classical NNs, we introduce a novel PiNN architecture, dubbed PCNN, which includes existing knowledge on the physics of the system at its core, with an application to building zone temperature modeling. The introduction of prior knowledge essentially works as an inductive bias, such that PCNNs do not need to learn everything from data, but only what we cannot easily characterize a priori.

While PCNNs model unforced temperature dynamics[1] with classical NNs, they treat parts of the inputs separately: the power input to the zone and the heat losses to the environment and neighboring zones are processed in parallel by a linear module inspired by classical physics-based RC models. This module ensures the positive correlation between power inputs and zone temperatures while forcing heat losses to be proportional to the corresponding temperature gradients to provide physically consistent predictions. This solves parts of the generalization issue of NN building models and makes PCNNs well-suited for control applications. The key however is that, unlike in classical physics-based models, no engineering effort is required to design and identify the parameters of PCNNs: we only need access to past data, and PCNNs are then trained in an end-to-end fashion to learn all the parameters simultaneously. Furthermore, we show that PCNNs achieve better accuracy than a baseline RC model on a case study. Moreover, they attain a precision on par with classical LSTMs on the validation data, despite performing worse on the training data. This shows that PCNNs do not

[1] Throughout this work, *unforced dynamics* represent the temperature evolution in the zone when no heating or cooling is applied and heat losses are neglected.

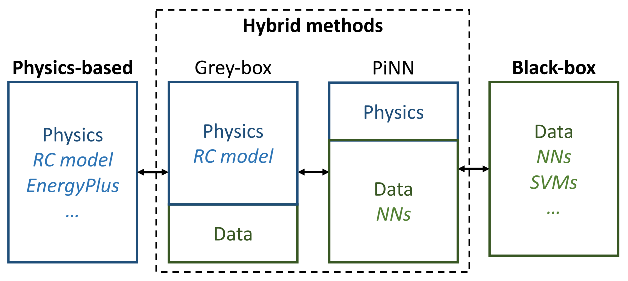


Fig. 2. Structural differences between the different methods..

lose much expressiveness due to their constrained architecture and have less tendency to overfit the training data. The main contributions of this work can be summarized as follows:

- PCNNs, novel PiNNs, are introduced and applied to zone temperature modeling.
- The physical consistency of PCNNs with respect to control inputs and exogenous temperatures[2] is formally proven.
- PCNNs perform comparably to LSTMs on the validation data and overfit the training data less.
- PCNNs attain better performance than classical RC models on a case study while avoiding any engineering overhead.

The rest of the paper is structured as follows. We start with a brief overview of the building modeling and PiNN literature in Section 2. We then describe what it means to be *physically consistent with respect to a given input*, present the PCNN architecture and formally prove its physical consistency in Section 3. The case study is described in Section 4, and we then compare the performance of this method against a classical RC model and LSTMs and provide a graphical interpretation of its physical consistency in Section 5. Finally, we discuss the potential and limitations of PCNNs in Section 6 and Section 7 concludes the paper.

2. Background

This section presents an overview of the existing literature on building models, which can be broadly classified into three categories: physics-based, black-box, and hybrid methods, as pictured in Fig. 2. While the latter can generally be further broken down into gray-box approaches and PiNNs, only gray-box modeling was previously applied to buildings, with the exception of [30,31]. For this reason, we propose a more general overview of PiNNs in Section 2.3.2.

Due to the vast literature on building modeling, we only provide a short summary of the strengths and weaknesses of the different techniques, and more details can be found in dedicated reviews, such as [8,13,33–38].

2.1. Physics-based building models

Also known as white-box or first principle models, physics-based models rely on Ordinary Differential Equations (ODEs), such as convection, radiation, or conduction equations, to describe building thermal dynamics. These methods were dominating the field early on when the lack of available data hindered the development of data-driven models [33].

Since they are grounded in first principles, a natural advantage of these approaches is the interpretability of the solutions [33]. Additionally, this gives them interesting generalization capabilities outside of the training data [36]. On the other hand, however, due to the complexity of detailed thermal models, assumptions and simplifications have to be made, such as in the choice of the ODEs, which can limit the accuracy of physics-based models [9]. Moreover, the more precision desired, the more knowledge and time is required to design the model and find the corresponding parameters, typically concerning the building envelope and the HVAC system, which might introduce uncertainty [11,39].

To simplify the design of physics-based models, various detailed simulation tools were developed, such as EnergyPlus, Modelica, TRNSYS, or IDA ICE [40–42]. While these models can attain good accuracy and respect the underlying physical laws, they are notoriously hard to calibrate [43–45], entail considerable development and implementation costs to find and detail all the required parameters [46], and suffer from a high computational burden at run-time [47].

2.2. Black-box building models

As opposed to physics-based methods, black-box (or data-driven) models do not rely on first principles but derive patterns from historical operational data. The most widely used methods rely on Multiple Linear Regression (MLR), Support Vector Regression (SVR), NNs, and ensembles, apart from the classical Autoregressive Integrated Moving Average (ARIMA) models, as reviewed by Bourdeau et al. [37].

Black-box models are generally easier to use than physics-based ones since no expert knowledge is required at the design stage, but often lack generalization guarantees outside of the data they are trained on [13,33]. Furthermore, they need historical data as input, sometimes in large amounts, to achieve satisfactory accuracy [13], and the data additionally has to be *exciting enough*, i.e. to cover the different operating conditions of the building, something not trivial, as discussed in Section 1. The subsequent data imbalance issue can, for example, be tackled through the creation of sub-models, like in Zhang et al. [48]. Moreover, black-box models are sensitive to the choice of features – or feature extraction methods – used as model inputs [9]. On the other hand, an advantage of data-driven methods is their flexibility, as they can be scaled to large systems in a more straightforward manner than physics-based methods [49]. Additionally, they are generally easier to transfer from one building to another since similar model architectures can be used and all the parameters are learned from data.

Very recently, as a consequence of the growing amount of available data, Deep Learning (DL) has started to be applied to building modeling [37]. For example, recurrent NNs (RNNs) were shown to provide better accuracy than feedforward NNs for the prediction of energy consumption [50]. In another study, a specific gated Convolutional NN (CNN) was shown to outperform RNNs and the classical Seasonal ARIMAX model on day-ahead multistep hourly predictions of the electricity consumption [14]. Due to the nonconvexity of classical NN-based models, which makes them hard to use in optimization procedures, researchers also used specific control-oriented models, such as Input Convex NN (ICNN), to model building dynamics [51].

2.3. Hybrid methods

Hybrid methods combine physics-based knowledge with existing data to have the best of both worlds. Note that some researchers use the term “hybrid methods” to refer to the fact that they first build a physics-based model and then fit a black-box model to it to then accelerate the inference procedure at run-time, such as [47,52], which is out of the scope of this overview and hence not covered here.

[2] We refer to the temperature outside and in neighboring zones together as *exogenous temperatures* in this work.

2.3.1. Gray-box building models

In gray-box modeling, one generally starts from simplified physics-based equations and uses data-driven methods to identify the model parameters [10,12,46] and/or learn an unknown disturbance model on top of it [53]. The simplified base model requires less expert knowledge and time to be designed than pure physics-based models but still allows one to retain the interpretability of physics-based models. Furthermore, this basis includes physical knowledge in the model, so that less information has to be learned from data compared to pure black-box models, which in turn implies that less historical data is required to fit such models [34].

Typical gray-box models start with linear state-space models and identify their parameters from data, even if some nonlinearities are not well captured by this approach [49]. Due to the difficulty of finding good parameters in general, low complexity RC models usually perform better, with models with one or two capacitances usually being selected [46,54,55]. Higher-order models furthermore entail more complexity and hinder the generalization capability of gray-box models, which also advocates in favor of low-complexity frameworks [56]. As a partial solution, a feature assessment framework to test the flexibility, scalability, and interoperability of gray-box models and select the right model characteristics was proposed by Shamsi et al. [12]. In essence, gray-box approaches hence allow for a trade-off between the accuracy and the complexity of building models [56].

Due to the effectiveness of low-order RC models, we hence rely on linear first-order RC modeling techniques inspired from Bünning et al. [57] and simplified versions of Maasoumy et al. [58–60] to construct the PCNNs proposed in this work.

2.3.2. Physics-informed neural networks

While early DL applications used classical feedforward NNs, researchers soon realized how transferring prior knowledge to NNs could be beneficial. Among the success stories, one can find the CNN and RNN families, specially designed to capture spatial invariance [61] and temporal dependencies [62] in the data, respectively.

In recent years, a new field emerged in the Machine Learning community to tackle the generalization issue of neural networks and create new NN architectures bound to follow given physical laws, such as Hamiltonian NNs [28] or Lagrangian NNs [27], later generalized by Djeumou et al. [18]. In parallel, PiNN architectures flourished, pioneered by the physics-guided NNs of Karpatne et al. [26,63] and the more general physics-informed Deep Learning (DL) framework originally proposed by Raissi et al. [64–66]. Since then, various methods to include prior knowledge in NNs have been proposed, several of which can be found in [29], where the authors tried to classify them.

Methodologically, the PCNNs proposed in this work are close to the physics-interpretable shallow NNs, where the inputs are also processed by two modules in parallel, one to retain physical exactness when possible and one to capture nonlinearities through a shallow NN [67]. Also related in spirit to the PCNN architecture, Hu et al. introduced a specific learning pipeline, where the output of the forward NN is fed back through a physics-inspired NN structure to reconstruct the input and hence ensure the forward process retains physical consistency [68].

Finally, two recent works applied PiNNs to create control-oriented building models [30,31]. Drgoňa et al. replaced the state, input, disturbance, and output matrices of classical linear models with four NNs and leveraged known physical rules to enforce constraints on them [30]. They additionally used the Perron–Frobenius theorem to enforce the stability and dissipativity of the system by bounding the eigenvalues of all the NNs. On the other hand, Gokhale et al. relied on a more classical PiNN approach with the introduction of a new physics-inspired loss term to guide the learning towards physically meaningful solutions without modifying the NN architecture [31]. However, neither of these works provide physical consistency guarantees, unlike the PCNN architecture presented in this work.

3. Methods

This section firstly defines a notion of physical consistency and then details the novel PCNN structure proposed in this work, where the effect of the control inputs and the heat losses to the environment and neighboring zones are separated from the unforced temperature dynamics. Finally, we formally prove the physical consistency of PCNNs with respect to control inputs and exogenous temperatures.

3.1. Respecting the underlying physical laws

Throughout this work, we define a model as being *physically consistent with respect to a given input* when any change in this input leads to a change of the output that follows the underlying physical laws. In our case, for example, we need models that are physically consistent with respect to control inputs to ensure that turning the heating on leads to higher zone temperatures than when heating is off, and vice versa for cooling. Mathematically, we can express this requirement as follows for a zone with power input $P \in \mathbb{R}$ at time step j and temperature prediction $T \in \mathbb{R}$ at time step k:

$$\frac{\partial T_k}{\partial P_j} > 0 \qquad \forall 0 \le j < k, \tag{1}$$

where we consider $P < 0$ in the cooling case by convention throughout this paper. We can similarly define physical consistency with respect to the outside temperature $T^{out} \in \mathbb{R}$, the temperature in a neighboring zone $T^{neigh} \in \mathbb{R}$, and the solar gains $Q^{sun} \in \mathbb{R}$ as follows:

$$\frac{\partial T_k}{\partial T_j^{out}} > 0 \qquad \forall 0 \le j < k, \tag{2}$$

$$\frac{\partial T_k}{\partial T_j^{neigh}} > 0 \qquad \forall 0 \le j < k, \tag{3}$$

$$\frac{\partial T_k}{\partial Q_j^{sun}} > 0 \qquad \forall 0 \le j < k, \tag{4}$$

since higher exogenous temperatures or solar gains all lead to increased zone temperatures.

3.2. Physically consistent neural networks

The proposed PCNN architecture is sketched in Fig. 3 for one time step k, and we apply it recursively over the prediction horizon. The temperature of the zone T is computed as the sum of two latent variables evolving through time: the *unforced dynamics* $D \in \mathbb{R}$, and the *energy accumulator* $E \in \mathbb{R}$, which includes prior knowledge about thermal dynamics. Mathematically, we thus have:

$$T_{k+1} = D_{k+1} + E_{k+1} \tag{5}$$

Remark 1 (*Extension to several neighboring zones*). While we describe and analyze the case with a single neighboring zone throughout this paper, it is straightforward to extend PCNNs to model a zone connected to several other zones by adding further energy loss terms with their corresponding scaling constants.

3.2.1. Linear physics-inspired module

The energy accumulator E is firstly positively influenced by the power input to the zone $g(u) \in \mathbb{R}$, which depends on the control input $u \in \mathbb{R}^m$, e.g. the opening pattern of radiator valves. The latter is scaled by a constant a in the heating and d in the cooling case to represent its effect on the air mass in the room. Note that the power input is negative in the cooling case by convention, so that cooling lowers the energy accumulated in E, as expected.

Secondly, from the laws of thermodynamics, we know that the modeled zone loses energy through heat transfers to the environment

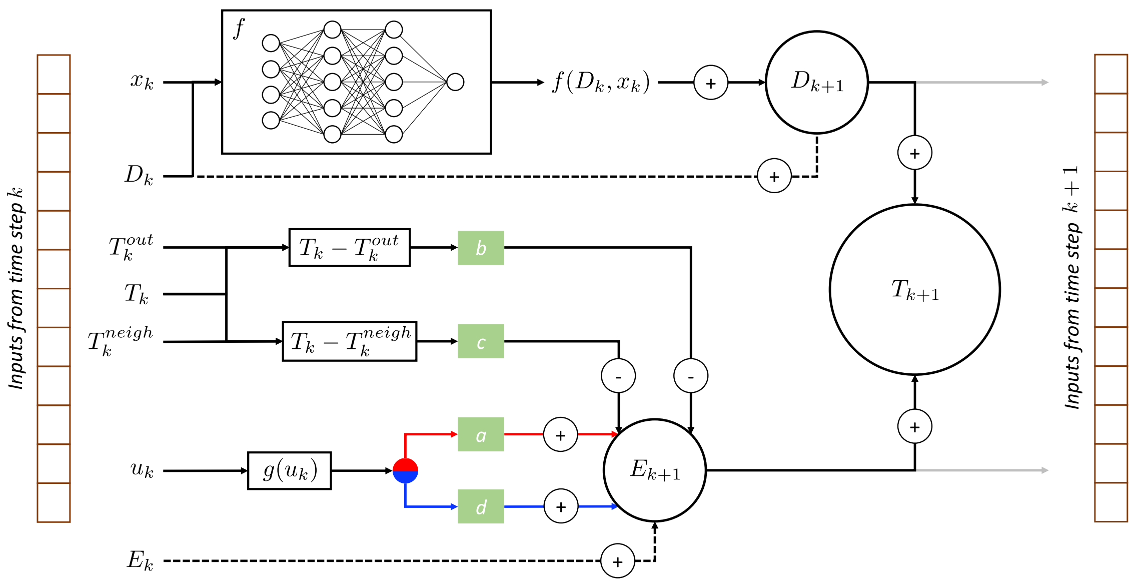


Fig. 3. The proposed PCNN architecture used recursively at each time step. The control inputs u, transformed into power inputs by the function g, and the losses to the environment $b(T - T^{out})$ and neighboring zone $c(T - T^{neigh})$ all influence an energy accumulator E, which accumulates or dissipates energy at each time step. Here, the separation between red and blue lines signals a different treatment of the power inputs in the heating and cooling case, respectively, since they are scaled by different constants a and d. The accumulated energy is then added to the unforced dynamics D, modeled by a residual NN that takes all the features apart from u, T^{out}, and T^{neigh} – gathered in x – as input, to get the final zone temperature prediction T..

and the neighboring zone. We hence subtract these effects, which are proportional to the corresponding temperature gradients with the outside temperature T^{out}, respectively the temperature in the neighboring zone T^{neigh}, scaled by parameters b, respectively c, learned from data. Mathematically, in the heating case, we can hence write the evolution of the physics-inspired module as follows:

$$E_{k+1} = E_k + ag(u_k) - b(T_k - T_k^{out}) - c(T_k - T_k^{neigh}), \tag{6}$$

with $E_0 = 0$. In the cooling case, one simply needs to exchange the parameter a with d. As can readily be seen, Eq. (6) is heavily inspired by classical first-order RC building models, found e.g. in Bünning et al. [57] and detailed in Appendix A.1. The main difference with the generic RC model in Eq. (22) is that the proposed PCNN architecture allows us to treat nonlinear solar and additional unknown heat gains using neural networks or other nonlinear functions in D instead of relying on engineered linear solutions. Furthermore, the physics-inspired parameters a, b, c, and d are learned from data simultaneously to the parameters of the black-box module described below (see Section 3.2.3).

Remark 2 (*Design of g*). In some cases, we can directly control the heating or cooling power input to the zone, i.e. $g(u) = u$. When this is not possible, e.g. when u controls the opening of the valves in radiators, we need to process the controllable inputs into power inputs through some function g. This function might be engineered, for example as $g(u) = u * \dot{m} * (T^w - T)$ in the case of a radiator, with $\dot{m}$ the mass flow and T^w the temperature of the water in the pipes, or it could be learned from data, e.g. using NNs. This learned function should be strictly monotonically increasing with $g(0) = 0$, i.e. no energy is consumed when there is no control input, $g(u) < 0$ when cooling is on, and $g(u) > 0$ when heating is applied. Importantly, since everything is trained together in an end-to-end fashion (see Section 3.2.3), g can seamlessly be learned in parallel to the other parameters.

Remark 3 (*Coupling between D and E*). Note that since $T_k = D_k + E_k$, the nonlinear black-box module D influences the evolution of the energy accumulator E in Eq. (6), which is one of the main differences with classical gray-box techniques, where the physics-based and black-box modules are usually completely separated. This furthermore requires learning the parameters a, b, c, and d simultaneously to the NNs in D, as presented in Section 3.2.3. Further details on the differences between classical gray-box approaches and PCNNs are discussed in Section 6.1.

3.2.2. *Black-box module*

Running in parallel of the linear module, the nonlinear black-box module processes all inputs not treated in E, such as solar gains and time information, gathered in $x \in \mathbb{R}^n$, to capture the unforced temperature dynamics, i.e. when no heating or cooling is applied and heat losses are neglected. This can typically be modeled using residual NNs, which leads to the following expression:

$$D_{k+1} = D_k + f(D_k, x_k) \tag{7}$$
$$D_0 = T(t_0),$$

where $T(t_0)$ is the measured temperature at the beginning of the prediction horizon and f is a potentially highly nonlinear function, typically based on recurrent neural networks. Remarkably, the unforced dynamics D are independent of power inputs and heat losses by design, which will allow us to prove the physical consistency of PCNNs with respect to these inputs in Section 3.3.

Remark 4 (*Design of f*). While f is composed of an encoder-LSTM-decoder structure in our case (see Section 4.2), any NN architecture – and even functions that do not contain NNs – can be used without affecting the physical consistency of the predictions. Nonetheless, due to the sequential nature of temperature dynamics and the expressiveness of NNs, we suspect RNNs to be a good choice in general.

3.2.3. *Training procedure*

Importantly, PCNNs do not require any engineering or knowledge about the building structure or parameters beyond connectivity information, i.e. which zones are adjacent to the modeled one. Given a data set of measurements $\mathcal{D} = \{\tilde{x}(t), T(t)\}_{t=1}^N$, where $\tilde{x} = \{x, u, T^{out}, T^{neigh}\}$ and N is the number of data points, we can directly optimize all the parameters of both the physics-inspired and black-box modules together.

To that end, we first construct a set of S time series $\mathcal{D}_S = \{\tilde{x}^{(s)}(t), T^{(s)}(t)\}_{s=1}^S$, each consisting of consecutive data points from

$\mathcal{D}$. Note that these sequences might overlap in practice to increase the data efficiency of the proposed method. We then minimize the Mean Squared Error (MSE) between the PCNN predictions $T_{k+1}(\tilde{x}^{(s)}_{0:k})$, recursively computed from past inputs $\tilde{x}^{(s)}_{0:k}$ up to time k, and the true measurements $T^{(s)}(k+1)$ for each time series s over the predefined prediction horizon H:

$$\frac{1}{S}\sum_{s=1}^{S}\left[\frac{1}{H}\sum_{k=0}^{H-1}\left(T_{k+1}(\tilde{x}^{(s)}_{0:k})-T^{(s)}(k+1)\right)^2\right], \tag{8}$$

In this work, we parametrize f using RNNs and rely on the standard automatic Backpropagation Through Time (BPTT) algorithm [69] and the PyTorch library [70] to solve this optimization problem.

3.3. PCNNs follow physical laws by design

Plugging Eqs. (6)–(7) in (5), we get:

$$T_{k+1} = T_k + f(D_k, x_k) + ag(u_k) \\ -b(T_k - T^{out}_k) - c(T_k - T^{neigh}_k) \tag{9}$$

Applying Eq. (9) recursively, as detailed in Appendix B.2, one can express the temperature prediction of the PCNNs at any future time step i as follows:

$$T_{k+i} = (1-b-c)^i T_k \\ + \sum_{j=1}^{i}(1-b-c)^{(j-1)}[f(D_{k+i-j}, x_{k+i-j}) \\ + ag(u_{k+i-j}) + bT^{out}_{k+i-j} + cT^{neigh}_{k+i-j}] \tag{10}$$

Here, it is important to note that $D_{m+1} = D_m + f(D_m, x_m)$ is independent of the variables u, T^{out}, and T^{neigh} at any step m, it solely depends on the other inputs in x, so we do not need to explicitly write the recursion out.

In the case when we can directly control the power input to the zone, i.e. $g(u) = u$, we can now formally assess the physical consistency of PCNNs with respect to control inputs and exogenous temperatures since we get the following partial derivatives:

$$\frac{\partial T_{k+i}}{\partial u_{k+i-j}} = (1-b-c)^{(j-1)}a \qquad \forall j = 1,\dots,i. \tag{11}$$

$$\frac{\partial T_{k+i}}{\partial T^{out}_{k+i-j}} = (1-b-c)^{(j-1)}b \qquad \forall j = 1,\dots,i. \tag{12}$$

$$\frac{\partial T_{k+i}}{\partial T^{neigh}_{k+i-j}} = (1-b-c)^{(j-1)}c \qquad \forall j = 1,\dots,i. \tag{13}$$

Remarkably, these derivatives take the same form in a classical RC model, as shown in , Appendix B.1. PCNNs hence satisfy the physical consistency criteria of Eqs. (1)–(3) as long as the conditions below hold:

$$a, b, c > 0 \\ 1-b-c > 0 \tag{14}$$

This is the case for real systems since a, b, and c are small positive physical constants, i.e. inverses of resistances and capacitances. Moreover, it gives us simple verification criteria to ensure that the learned PCNN stays physically consistent as these conditions could easily be enforced during the training of the models, even though it was not needed in our experiments[3].

Note that even when we do not have access to the power input directly and have to process the control inputs through an engineered or learned function g, we still get:

$$\frac{\partial T_{k+i}}{\partial g(u_{k+i-j})} = (1-b-c)^{(j-1)}a \qquad \forall j = 1,\dots,i, \tag{15}$$

[3] The values learned by PCNNs in practice have orders of magnitude $10^{-1}-10^{-2}$ for a and d and $10^{-3}-10^{-4}$ for b and c.

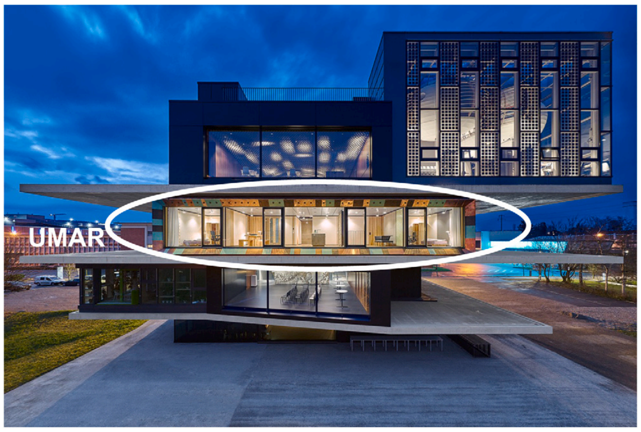


Fig. 4. NEST building, Duebendorf, and the UMAR unit circled in white © Zooey Braun, Stuttgart..

which remains positive under the same conditions. This ensures that any change in the power input, as computed by a function g, still yields the expected physically consistent outcome on the zone temperature. As long as g satisfies the conditions in Remark 2, we furthermore observe that:

$$\frac{\partial T_{k+i}}{\partial u_{k+i-j}} = \frac{\partial T_{k+i}}{\partial g(u_{k+i-j})}\frac{\partial g(u_{k+i-j})}{\partial u_{k+i-j}} \\ = (1-b-c)^{(j-1)}a\frac{\partial g(u_{k+i-j})}{\partial u_{k+i-j}}, \tag{16}$$

which remains positive, hence satisfying Eq. (1), as long as the conditions in Eq. (14) hold since g is defined as a monotonically increasing function.

Remark 5 (*Condition on d*)**.** Replacing a by d throughout Eqs. (9)–(16) yields similar conditions for the cooling case. In particular, we require to have $d > 0$ to retain physical consistency, additionally to the conditions in Eq. (14).

4. Case study

In this work, we take advantage of NEST, a vertically integrated district located in Duebendorf, Switzerland, and pictured in Fig. 4 [71]. NEST is composed of several residential and office units, and we focus our attention on the "Urban Mining and Recycling" (UMAR) unit, where more than three years of data is available to assess the quality of our models.

4.1. UMAR

UMAR is an apartment composed of two bedrooms, with a living room in between them, and two small bathrooms. We model the temperature of one of the bedrooms throughout this work. All the rooms are equipped with radiant heating/cooling panels in the ceiling and controlled by opening and closing valves to let hot or cold water flow through them depending on the season. Since individual room power consumption measurements are not available, we approximate them by disaggregating the total consumption of UMAR using the design mass flows and the amount of time the valves in each room are open. Apart from the temperature and power consumption of the rooms, we also use data about the solar irradiation and the ambient temperature on-site. Details on the data preprocessing can be found in Appendix C.

Since both bathrooms are much smaller and have significantly less heating/cooling power than the bedrooms and the living room, we assume that the heat transfers between the former and the latter are negligible compared to the other heat transfers. In other words, we do not consider the bathrooms as distinct zones and only include the living room as neighboring zone of the modeled bedroom.

4.2. Implementation details

In our case, the inputs x are comprised of the raw solar irradiation and time information, i.e. the day of the week, the month of the year, and the current time of the day, and we assume direct control over the power input, i.e. $g(u) = u$. All the models are trained to predict the temperature of the zone over a horizon of three days with time steps of 15 minu, and the data was split in a training and validation set. For our experiments, we chose to design f with an encoder-LSTM-decoder structure, where both the encoder and decoder have two layers of 128 units and the LSTM is composed of two layers of size 512. The code of the PCNNs can be found on GitLab[4] and further implementation details in Appendix D.

One critical implementation point is the initialization of the parameters a, b, c, and d of the PCNN in Fig. 3. Indeed, as they are inspired by the known physics of buildings, they must correspond to meaningful values. Furthermore, due to the recurrent use of these parameters to modify the state of the energy accumulator along the prediction horizon, wrong values would have a large impact on the quality of the model and the PCNN might get stuck in a local minimum. In practice, we saw that using rules of thumb to initialize those parameters to plausible values using our prior knowledge and then letting the PCNN modify them during the back-propagation procedure led to good results, as presented in Section 5.2. We thus use our intuition about how UMAR behaves to define initial values such that:

- For a and d: The temperature in the zone rises/drops by 1 °C in 2 h when the maximal heating/cooling power is applied.
- For b and c: The temperature drops by 1.5 °C in 6 h when the exogenous temperature is 25 °C lower.

These rules of thumb can be derived from historical data, for example looking at how much time it generally takes for the temperature to rise by 1 °C when the zone is heated at full power for a, respectively to drop by 1.5 °C when it is 25 °C colder outside and heating is off for b. Similar investigations will give plausible initial values for c and d. Since these parameters turn out to be very small in practice, we learn their inverse for numerical stability (see Appendix D).

5. Results

In this section, we analyze the performance of the PCNN that obtained the best error on the validation set of the case study and compare it to a physics-based RC model baseline. Since more complex structures often do not improve the accuracy of RC models, as discussed in Section 2.3.1, we chose a 2R1C model with two heat sources, the heating/cooling power and the solar gains, (see Appendix A). This simple architecture furthermore presents the advantage to have a very similar form to the physics-based module E in PCNNs (see Appendix B), which allows us to assess the impact of the NNs in D on the model performance. Note that the RC model has a sampling time of 1 min, we thus keep the power input fixed over intervals of 15 min when we compare its predictions with the ones of the PCNN.

5.1. Improving the generalization issue of NNs

While we only discuss one PCNN in depth throughout this section, a broader analysis can be found in Appendix E, where we trained PCNNs with different random seeds and on the other bedroom in UMAR for comparison. We obtained consistent results, showing the robustness, respectively the flexibility of the approach.

We additionally performed an ablation study where we removed the physics-inspired prior E and used only the classical encoder-LSTMs-decoder framework in f to predict the temperature evolution, concatenating all the available features as inputs (hence losing any physical

[4] https://gitlab.nccr-automation.ch/loris.dinatale/pcnn.

Table 1
Comparison training and validation loss for three classical LSTMs and PCNNs, scaled by 10^3 (full table in Appendix E).

	Seed	Training loss	Validation loss
LSTMs	0	0.57	2.28
	1	0.57	1.92
	2	1.14	2.30
	Mean	**0.76**	**2.17**
PCNNs	0	1.83	1.93
	1	1.85	1.65
	2	2.06	1.75
	Mean	**1.91**	**1.78**

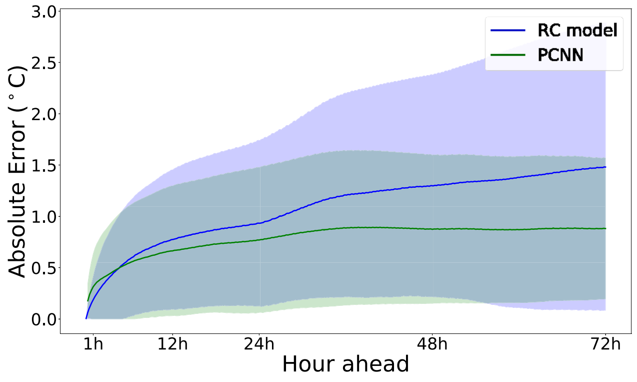


Fig. 5. Mean and standard deviation of the error at each time step of the prediction horizon for both the RC model in blue and the PCNN in green, where the statistics were computed from almost 2000 predictions from the validation set..

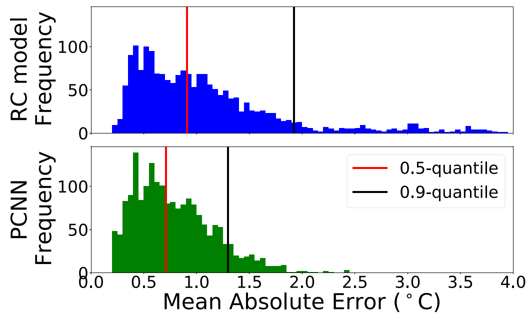
consistency guarantee). Interestingly, as presented in Table 1, while PCNNs could not attain the performance of classical LSTMs on the training data due to their constrained structure to follow the underlying physical laws, they obtained lower errors on the validation set. This confirms that PCNNs solve part of the generalization issue of classical NNs, having a smaller tendency to overfit the training data but retaining enough expressiveness to perform well on new data.

In the rest of this section, however, we only investigate in detail the accuracy of the best PCNN compared to the RC model since LSTMs were found to be physically inconsistent in our experiments, as pictured in Fig. 1, a critical issue for control-oriented thermal models.

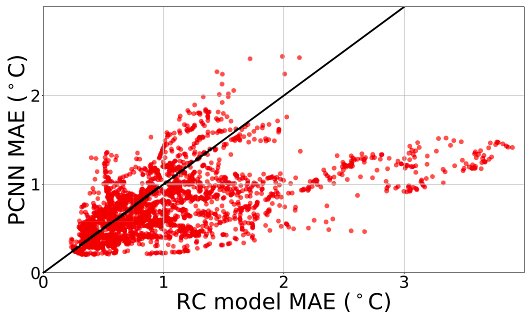
5.2. Performance analysis

Since predicting the evolution of the temperature for several time steps entails a recursive use of the architecture in Fig. 3, we leverage the ability of LSTMs to handle long sequences of data to minimize the error over the entire horizon. On the other hand, RC models are usually fitted over a single step, leading to error propagation, as pictured in Fig. 5, where we plotted the average Absolute Error (AE) and one standard deviation for both models over almost 2000 possibly overlapping 3-day long sequences of data from the validation set of the PCNN, i.e. unseen data. Note that while the RC model used as baseline in this work is not optimal, it was nonetheless tuned to obtain good accuracy, with an average error below 1 °C after 24 h.

One can observe the PCNN providing better predictions than the RC model in general, which is supported by the average AE reported at key points along the horizon in Table 2. In particular, the PCNN is able to keep a good accuracy even on long horizons, with an error more than 40% lower than the RC model after three days. On the other hand, it presents slightly higher errors at the beginning of the horizon because of the warm start that is implemented (Appendix D): since



(a) Distribution of the MAE of both models over the test sequences, with the 50% and 90% quantiles marked in red, respectively black.



(b) Scatter plot of the MAEs of both models on each test sequence, with the black diagonal line representing equal performance.

Fig. 6. Comparison of the MAE of both the PCNN and RC model over almost 2000 predictions of three days, taken from the unseen validation data of the PCNN.

Table 2
Comparison of the MAE of the two models over the prediction horizon.

Hours ahead	RC model	PCNN (ours)
1 h	**0.19** °C	0.31 °C
6 h	0.58 °C	**0.55** °C
12 h	0.78 °C	**0.66** °C
24 h	0.93 °C	**0.77** °C
48 h	1.30 °C	**0.88** °C
72 h	1.48 °C	**0.88** °C

they firstly predict past data – the last 3 h – PCNNs might indeed start the actual prediction horizon at a temperature different from the true one. Nonetheless, since we observed that the warm start benefited the overall performance of PCNNs during our experiments, we kept it in the final implementations.

To investigate the Mean Absolute Errors (MAEs) obtained by both models on each sequence of data, we also provide the corresponding histograms and scatter plot in Fig. 6. In general, one can see the PCNN dominating the RC model: there are only a few sequences where its error is significantly larger than the one of the RC model, represented by points over the black diagonal line in the right plot. On the other hand, towards the lower and right side of this figure, we find data sequences where the PCNN presents a significantly better accuracy than the RC model. This is confirmed by looking at the error distributions of both models in the left plot, with the errors of the PCNN (green) clustered below 1 °C and almost always below 2 °C while the errors of the RC models in blue are much more spread out. This indicates that the PCNN is robust with respect to different inputs, even on unseen data.

Altogether, we can hence conclude that the PCNN is less prone to extreme errors and keeps the majority of errors lower than the given RC baseline, proving its robustness and effectiveness. Remarkably, all the results were obtained on over three years of data, hence under various weather conditions and during all the seasons, which also hints that exogenous variables do not impact the quality of the model much.

5.3. *Empirical analysis of the physical consistency*

With the physical consistency of the models formally proven in Section 3.3, we can now visualize its impact empirically. Note that the main point of this analysis is to show that the PCNN retains physical consistency even on *unseen* data, i.e. data from the validation set and with various engineered power inputs that do not exist in the data, avoiding the classical generalization issue of NNs detailed in Section 1. To that end, we take an input sequence from the validation set and compare the temperature predictions of both the RC model and the PCNN when:

- the original and true power inputs are applied (blue),
- only the first half of the power inputs are used (red),
- only the second half of the input is applied (orange),
- no power is used (black), hereafter named *uncontrolled*,

where we separate the power inputs in half with respect to their magnitudes, i.e. so that both the red and orange control sequence apply roughly the same total power. One such experiment is summarized in Fig. 7 for a heating case, where we also added the ground truth in dashed blue as reference for both models.

Firstly, comparing the blue predictions with the dashed ground truth, we see both models performing well, exemplifying the results discussed in Section 5.2. In particular, the proposed PCNN is able to grasp the general trends to match the ground truth despite the large amount of heating power applied and the temperature rising to more than 30 °C, something unusual in a real setting and hence not well covered by the training data.

Furthermore, looking at the three other predictions, for which we do not have a ground truth anymore, both models again show similar behaviors. This is the visual consequence of the physical consistency proven in Section 3.3, with the red predictions deviating from the blue ones at the same point in time for both models: as soon as we stop heating the room, we get lower temperatures. Similarly, the orange predictions deviate from the uncontrolled dynamics at the same points in time for both models. Finally, looking at the uncontrolled predictions, one can observe smoother patterns for the PCNNs due to the unforced base dynamics being captured by LSTMs instead of the more aggressive linear regression at the core of the RC model.

To get a better visualization of the behavior of both models with respect to the different control inputs, we can subtract the uncontrolled predictions from the other curves. The result is pictured in Fig. 8 and allows us to assess the impact of the three different control sequences on the final predictions. As expected, both models still exhibit similar behaviors, with predictions diverging from the baseline as soon as heating is turned on. On the other hand, when heating is off, the gap with the baseline gets slowly closed because of the higher inside temperature leading to higher energy losses to the environment and the neighboring zone. Note that the impact of the neighboring room is hard to distinguish in that plot since it is an order of magnitude smaller than the losses to the outside.

6. Discussion

In this section, we briefly discuss the main differences between PCNNs and classical gray-box models and then mention potential applications of PCNNs, leveraging their physical consistency, and some hurdles that still need clarification.

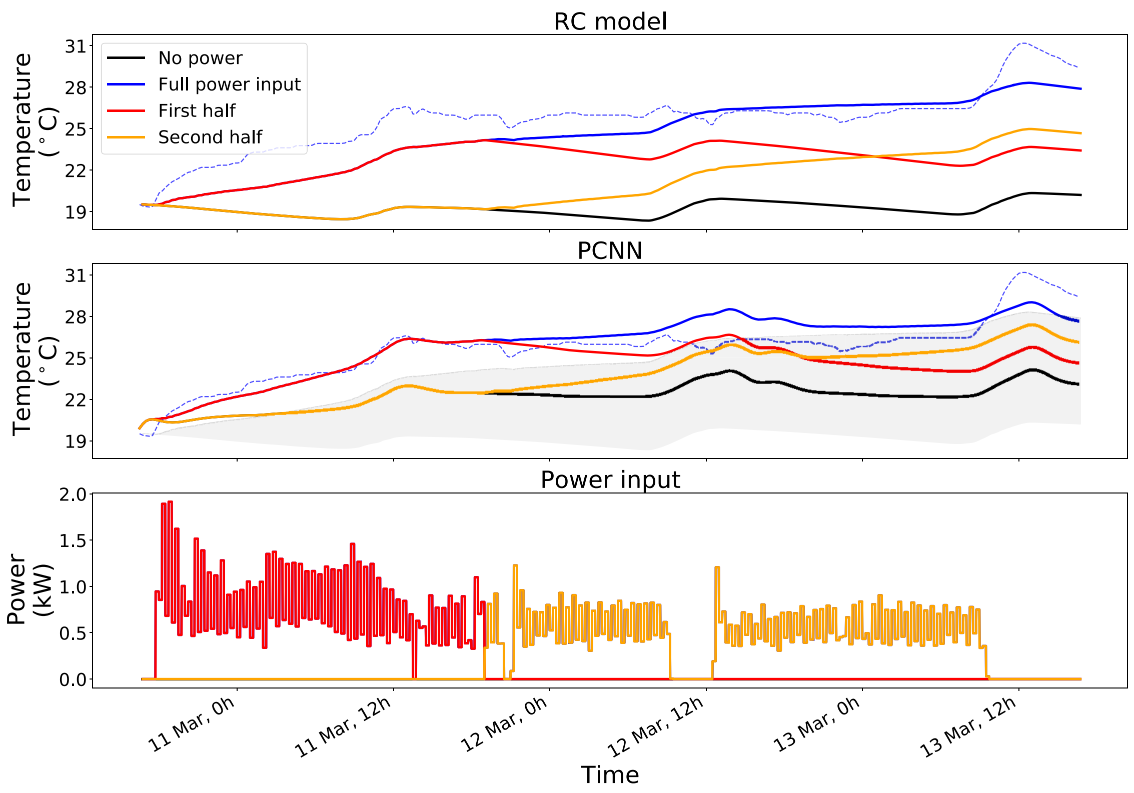


Fig. 7. Comparison between the RC model (top) and the PCNN (middle) given the bottom heating control sequence, over three days. In blue, one can assess the precision of both models compared to the ground truth (dashed), where the full control sequence was used. Then, red and orange show the result when only the first half of the control input, respectively the second one, is used. Finally, the black uncontrolled dynamics reflect the case when no power is used, and we shaded the span of the RC model predictions in the middle plot as reference..

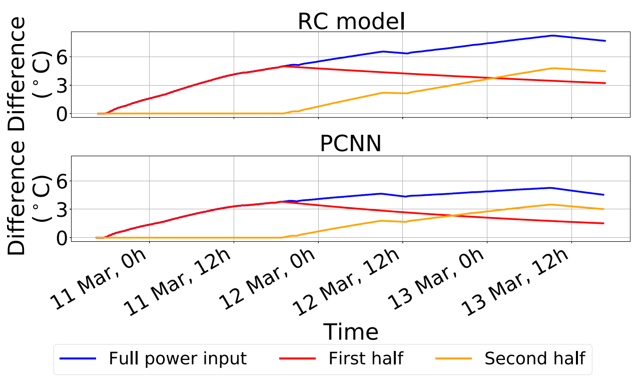


Fig. 8. Difference between each control input and the black baseline (no energy) in Fig. 7, for the physics-based model (top) and the proposed black-box structure (middle)..

6.1. Contrasting PCNNs with gray-box models

Since PCNNs are heavily inspired from classical RC models, we can derive them as a specific form of gray-box models, as detailed in Appendix F, written as follows:

$$\begin{aligned} \xi_{k+1} &= \xi_k + m(\xi_k, w_k^2) \\ z_{k+1} &= Az_k + B_u g(u_k) + B_{w^1} w_k^1 \\ &\quad + B_d \xi_k + \xi_{k+1} \end{aligned} \tag{17}$$

where z represents the state of the system, u the controllable inputs, w^1 uncontrollable ones, and ξ is a disturbance model computed as a function m of the rest of the uncontrollable inputs w^2. The main structural difference between Eq. (17) and classical gray-box formulations is the impact of the disturbance ξ, which appears both with and without a lag of one in the state update function. Traditional approaches generally first forget about the unknown disturbance ξ to identify A, B_u, and B_{w^1}, and then fit a disturbance model to the residuals, e.g. using Gaussian Processes [72].

Despite the similarity with classical gray-box models, PCNNs are fundamentally different both in terms of philosophy and training procedure. Firstly, the linear evolution of the state z captures the main dynamics of gray-box models, including the impact of control inputs, and the nonlinear disturbance ξ corrects them to match the data. On the other hand, in PCNNs, the main (unforced) dynamics D are processed by nonlinear NNs, while the linear energy accumulator E adjusts the predictions according to the controllable inputs and known disturbances, i.e. heat losses.

Secondly, contrary to classical techniques modeling the disturbance ξ as a separate process, all the parameters of PCNNs are trained simultaneously over the entire prediction horizon – PCNNs are multi-step-ahead models – and in an end-to-end fashion to capture dependencies between D and E, leveraging automatic BPTT.

6.2. Potential of PCNNs

As discussed, the good accuracy and physical consistency of PCNNs make them natural candidates for control-oriented zone temperature models. They could however also be used or integrated into Digital Twins (DTs), a fast-growing field that suffers from two problems that PCNNs could solve. Firstly, if historical data is available, they could reduce the effort and knowledge required to calibrate DTs. Secondly, even when the calibration is successful, DTs often remain slow at run-time due to the level of detail included, something that can be improved by training PCNNs to imitate their behavior and subsequently accelerate the inference procedure.

PCNNs could also be used in retrofitting operations, albeit restricted to the renewal of energy systems. Indeed, since PCNNs are robust to changes in the control inputs, one could assess the possibilities arising from different power systems, e.g. the impact of adding or subtracting some heating capacity. Combined with an intelligent controller, one could for example anticipate the potential energy savings of the new system and balance them with the installation costs to compute the return on investment of the operation.

Overall, we thus see good potential for PCNNs in the field of building modeling and beyond. Indeed, while we present a specific case study on zone temperature models in this work, it is noteworthy that the structure of PCNNs, with a physics-informed module in parallel to a black-box one, is very general and flexible. If more information about the physics of the system is known or required, the physics-informed module can be expanded to incorporate it. For example, if PCNNs are used to analyze the impact of solar gains, then E should include a detailed model of this process on top of the current formulation. Thanks to the modularity of PCNNs, the rest of the pipeline of Fig. 3 would not be modified, apart from possible changes in which inputs are contained in x. Typically, if a detailed model of the solar gains is included in E, then the solar irradiation does not need to be processed in D anymore. In general, any system with similar underlying physical processes, i.e. systems that can *accumulate* and *dissipate* energy, might be modeled with PCNNs, adjusting the physics-informed module.

6.3. Limits of our application

Firstly, it is important to note that the proposed structure does not fully solve the generalization issue of NNs: PCNNs are only physically consistent with respect to control inputs and exogenous temperatures, i.e. they satisfy the conditions in Eqs. (1)–(3). Should other inputs in x vary, we cannot guarantee the robustness of the model anymore. In particular, the current version of PCNNs does not meet the condition in Eq. (4). This is left for future work for two main reasons: one usually does not have direct access to solar gains through the windows of the zone to model, but only to the horizontal irradiation measurement, which has a nonlinear effect on the zone temperature. Furthermore, Eq. (4) is not strictly needed in control-oriented models, the main target of this work. Indeed, modifying the heating or cooling power inside a zone only impacts its temperature, and hence heat transfers indirectly, but the solar gains always have the same impact under any control input.

Secondly, it would be of interest to investigate the tuning of the various parameters, both to optimize the black-box f and, in particular, to understand how to initialize and learn meaningful values for a, b, c, and d. In this work, we initialized them using prior knowledge and rules of thumb, it remains unclear how and why they get to their final values, which can differ depending on the random seed used during training. Furthermore, we observed that the quality of the solution can vary significantly if they are initialized to unrealistic values, i.e. PCNNs do not always recover physically consistent parameters from data. Consequently, it might be useful to add constraints to ensure they retain meaningful values throughout training, i.e. they continuously meet the conditions in Eq. (14). In practice, however, we did not have that issue, with the parameters only slowly changing around their physics-informed initial value and thus staying physically consistent throughout our experiments.

Lastly, one should keep in mind that the RC baseline used in this paper, albeit tuned to obtain satisfying performance, still remains a low complexity model. While our PCNN was able to get better accuracy than this RC model, it might be possible to find better physics-based models. Nonetheless, PCNNs seem very competitive and avoid any need for engineering, which makes them attractive in general.

7. Conclusion

In this work, we presented a novel neural network architecture, dubbed PCNN, with an application to building zone temperature modeling. By treating some input variables separately from the main NN in a physics-informed module, PCNNs include prior knowledge in their structure. They are hence able to capture parts of the underlying physics while leveraging the accuracy of NNs to attain significant performance improvement over classical physics-based models without any engineering overhead.

A key advantage of PCNNs over existing NN-based thermal models is that we could formally prove that their temperature predictions remain physically consistent with respect to any control inputs and exogenous temperatures and over the entire prediction horizon. Furthermore, grounding PCNNs in the underlying physics allowed us to mitigate the usual generalization issue of classical NN frameworks.

These results were confirmed by our experiments on a bedroom temperature modeling case study, in which PCNNs generally obtained better results than classical LSTMs over the validation data, even when they were performing worse during the training phase, strongly indicating that they do not suffer from generalization issues as much as classical frameworks. Additionally, PCNNs clearly outperformed the RC model baseline while following the underlying physical laws, reducing the error by more than 40% at the end of the 3-day long prediction horizon.

Since PCNNs are solely based on data and do not require any engineering, they are very flexible and easy to use. This makes them interesting for different applications in the field of building modeling and beyond. To complete the analysis of their potential, it would be of interest to assess the sensitivity of PCNNs with respect to the key parameters of the physics-informed module, which should have links with physical quantities, and the amount of data required to attain satisfactory performance. In future work, we plan to focus our research on extending the current architecture to the multi-zone setting and use PCNNs in various control schemes to learn intelligent temperature controllers.

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 code and data used in this article are available open source and can be found on https://gitlab.nccr-automation.ch/loris.dinatale/pcnn.

Acknowledgments

This research was supported by the Swiss National Science Foundation under NCCR Automation, Switzerland, grant agreement 51NF40_180545.

Appendix A. RC building model

A.1. General RC models

In general, we can describe the thermal dynamics of a room with the following ordinary differential equation (ODE):

$$C\frac{dT}{dt} = \frac{dQ^{heat}}{dt} + \frac{dQ^{irr}}{dt} + \frac{dQ^{occ}}{dt} + \sum \frac{dQ^{cond}}{dt} + \sum \frac{dQ^{conv}}{dt}, \quad (18)$$

where T is the temperature, C the heat capacitance of the air mass, Q respectively represents heat flows from the heating/cooling system (negative values represent cooling energy), the solar irradiation, the occupants, heat conduction, and heat convection, respectively, where both sums are taken over the number of surfaces adjacent to the measured volume of air.

In this work, we consider conductive and convective transfer together in two heat transfers: one to represent transfer to the neighboring zone (assuming there is only one) and the other to gather losses to the environment, both being proportional to the corresponding temperature gradient. Additionally, we process the horizontal solar irradiation data to reflect the solar gains through the windows as follows:

$$Q^{irr}(t) = \frac{\sin(\theta - \theta_0)\cos(\phi)}{\sin(\phi)} I(t), \quad (19)$$

where I is the irradiation measured by the sensor, ϕ the altitude and θ the azimuth of the sun, and θ_0 accounts for the orientation of the window (as counter clock-wise rotation from a north–south-aligned surface facing east). Altogether, we can then rewrite the thermal dynamics as:

$$\frac{dT}{dt} = \frac{1}{C}\frac{dQ^{heat}}{dt} + \frac{\epsilon}{C}\frac{dQ^{irr}}{dt} + \frac{\eta}{C}\frac{dQ^{rest}}{dt} - \frac{1}{CR_{out}}\frac{d(T - T^{out})}{dt} - \frac{1}{CR_{neigh}}\frac{d(T - T^{neigh})}{dt} \quad (20)$$

with ϵ representing the lumped permissivity of the windows and exterior walls, R_{out} and R_{neigh} the thermal resistance of the walls adjacent to the outside, respectively the neighboring zone, and T^{out} and T^{neigh} the temperature outside, respectively in the neighboring zone. We then group all the other heat gains in Q^{rest}, scaled by a parameter η and discretize this ODE with the Euler forward method and the time step $\Delta_t = 1$ min, yielding:

$$T_{k+1} = T_k + \Delta_t * \Big[\frac{1}{C}Q_k^{heat} + \frac{\epsilon}{C}Q_k^{irr} + \frac{\eta}{C}Q_k^{rest} - \frac{1}{CR_{out}}(T_k - T_k^{out}) - \frac{1}{CR_{neigh}}(T_k - T_k^{neigh})\Big] \quad (21)$$

Grouping the constants together and defining new parameters a, b, c, e_1 and e_2, we can reformulate it as follows:

$$T_{k+1} = T_k + aQ_k^{heat} - b(T_k - T_k^{out}) - c(T_k - T_k^{neigh}) + e_1Q_k^{irr} + e_2Q_k^{rest} \quad (22)$$

A.2. Baseline RC model

In this work, to create a simple RC model to use as a comparison baseline, we assume no knowledge of the occupants and other heat gains and discard the corresponding term $e_2Q_k^{rest}$. Rewriting Eq. (22), we get:

$$T_{k+1} - T_k = \begin{bmatrix} Q_k^{heat} \\ -(T_k - T_k^{out}) \\ -(T_k - T_k^{neigh}) \\ Q^{irr} \end{bmatrix}^T \begin{bmatrix} a \\ b \\ c \\ e_1 \end{bmatrix}$$

$$\Delta T_{k+1} = y_k^T p, \quad (23)$$

where ΔT represents the temperature difference, y groups the factors influencing it, and p the unknown parameters. Doing this for every time step, we can create matrices of data, grouping all the temperature differences in matrix X and the external factors in Y:

$$\begin{bmatrix} \Delta T_1 \\ \vdots \\ \Delta T_N \end{bmatrix} = \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix} p$$
$$X = Yp \quad (24)$$

Finally, we can use Least Squares to identify the parameters:

$$Y^TX = Y^TYp$$
$$p = (Y^TY)^{-1}Y^TX \quad (25)$$

Appendix B. Mathematical derivations

B.1. Physics-based predictions

We can rewrite the predictions of the RC model from Eq. (22) as follows:

$$T_{k+1} = (1-b-c)T_k + aQ_k^{heat} + bT_k^{out} + cT_k^{neigh} + eQ_k^{irr}, \quad (26)$$

Applying this transformation recursively yields the following two-steps-ahead temperature predictions:

$$\begin{aligned} T_{k+2} &= (1-b-c)T_{k+1} + aQ_{k+1}^{heat} + bT_{k+1}^{out} + cT_{k+1}^{neigh} + eQ_{k+1}^{irr} \\ &= (1-b-c)[(1-b-c)T_k + aQ_k^{heat} + bT_k^{out} + cT_k^{neigh} + eQ_k^{irr}] \\ &\quad + aQ_{k+1}^{heat} + bT_{k+1}^{out} + cT_{k+1}^{neigh} + eQ_{k+1}^{irr} \\ &= (1-b-c)^2T_k + (1-b-c)[aQ_k^{heat} + bT_k^{out} + cT_k^{neigh} + eQ_k^{irr}] \\ &\quad + aQ_{k+1}^{heat} + bT_{k+1}^{out} + cT_{k+1}^{neigh} + eQ_{k+1}^{irr} \end{aligned} \quad (27)$$

This leads to the general formula below for the temperature prediction i time steps ahead:

$$T_{k+i} = (1-b-c)^iT_k + \sum_{j=1}^{i}(1-b-c)^{(j-1)}[aQ_{k+i-j}^{heat} + bT_{k+i-j}^{out} + cT_{k+i-j}^{neigh} + eQ_{k+i-j}^{irr}] \quad (28)$$

Remarkably, this model is known to follow the laws of physics by design, i.e. it satisfies Eqs. (1)–(4) as long as all the parameters a, b, c, and e are small and positive. This is true for real systems since they represent small positive physical constants, i.e. inverses of resistances and capacitances.

B.2. Black-box predictions

PCNN predictions from Eq. (9) can be rewritten as follows:

$$T_{k+1} = (1-b-c)T_k + f(D_k, x_k) + ag(u_k) + bT_k^{out} + cT_k^{neigh} \quad (29)$$

When this formula is applied recursively, what the model does in practice, we get:

$$\begin{aligned}T_{k+2} &= (1-b-c)T_{k+1} + f(D_{k+1}, x_{k+1}) \\ &\quad + ag(u_{k+1}) + bT^{out}_{k+1} + cT^{neigh}_{k+1} \\ &= (1-b-c)[(1-b-c)T_k + f(D_k, x_k) \\ &\qquad + ag(u_k) + bT^{out}_k + cT^{neigh}_k] \\ &\quad + f(D_{k+1}, x_{k+1}) + ag(u_{k+1}) \\ &\quad + bT^{out}_{k+1} + cT^{neigh}_{k+1}\end{aligned} \tag{30}$$

Note that $D_{k+1} = D_k + f(D_k, x_k)$ is independent from all the other variables u, T^{out}, and T^{neigh}. We can thus keep the recursion as is, only noting that at each time step D goes through the neural network function f so that we end up with a nested application of f to the inputs x. However and crucially, they do not get impacted by changes of control input, and we can write temperature predictions from the model as follows:

$$\begin{aligned}T_{k+2} &= (1-b-c)^2 T_k + (1-b-c)[f(D_k, x_k) \\ &\qquad + ag(u_k) + bT^{out}_k + cT^{neigh}_k] \\ &\quad + f(D_{k+1}, x_{k+1}) + ag(u_{k+1}) \\ &\quad + bT^{out}_{k+1} + cT^{neigh}_{k+1}\end{aligned} \tag{31}$$

Similarly to the physics-based case, this leads to the following general formula for any future predictions:

$$\begin{aligned}T_{k+i} &= (1-b-c)^i T_k \\ &\quad + \sum_{j=1}^{i} (1-b-c)^{(j-1)}[f(D_{k+i-j}, x_{k+i-j}) \\ &\qquad + ag(u_{k+i-j}) + bT^{out}_{k+i-j} + cT^{neigh}_{k+i-j}]\end{aligned} \tag{32}$$

Appendix C. Data preprocessing

C.1. NEST data

Data from all the sensors in NEST is sampled and stored at a frequency of one minute. Concerning the solar irradiation data, we delete constant streaks of more than 20 h than indicate a fault of the sensor – where *deleting* refers to setting the values to *NaN* – and clip the measurement at 0 since it cannot be negative. For the outside temperature, we delete constant streaks of more than 30 min. Both measurements are then smoothed with a Gaussian filter with $\sigma = 2$. For power inputs, we delete constant streaks of more than 1 day and smooth the measurements with a Gaussian filter with $\sigma = 1$. Finally, the temperature measurements in both the room of interest and the neighboring one are smoothed with $\sigma = 5$.

Before using the created data, we linearly interpolate all the missing values when less than 30 min of information is missing. When we use it to train and test PCNNs, the data is subsampled to 15 min intervals through averaging.

C.2. Individual room energy consumption

As mentioned in Section 4, UMAR has a unique power meter and we need to disaggregate this global measurement P^{tot} into individual consumption for each room. To that end, we use the design mass flow $\dot{m}^i$ of room i, something known from technical construction sheets. At each time step t, we then approximate the power consumed by each room, P^i, as follows:

$$P^i_t = \frac{u^i_t \dot{m}^i_t}{\sum_k u^k_t \dot{m}^k_t} P^{tot}_t, \tag{33}$$

where u^i is the amount of time the valves are opened and we sum over all the $k = 5$ rooms in UMAR. In words, we approximate the individual energy consumption of each to be proportional to the amount of water flowing through its ceiling panels.

Appendix D. Implementation details

The month and time of day variables are represented by sine and cosine functions to introduce periodicity, so that the last month has a value close to the first month of the year for example. Mathematically, two variables are created:

$$t^{sin}_m = \sin\left(\frac{m}{12}2\pi\right), \quad t^{cos}_m = \cos\left(\frac{m}{12}2\pi\right), \tag{34}$$

where the months m are labeled linearly and in order from 1 to 12. The same processing is done for the time step in during the day, replacing the factor 12 in Eq. (34) by 96, the number of 15 min interval in one day.

We let the initial hidden and cell state of the LSTM be learned during training and additionally give the model a warm start of 3 h, i.e. the PCNN first predicts the last 12 time steps in the past, where we feed the true temperatures back to the network to initialize all the internal states, before predicting the temperature over the given horizon. We train the PCNN to minimize the Mean Square Error (MSE) of the predictions over a horizon of 3 days with 15-min time steps, and use the Adam optimizer with a decreasing learning rate of $\frac{0.001}{\sqrt{h}}$ at epoch h. We create sequences of data using sliding windows of minimum 12 h – and maximum 3 days – with a stride of 1 h. We then separate both the heating and cooling season in training and validation data with an 80%–20% split to ensure a fair partition of heating and cooling cases in the training and validation sets. Finally, the data is normalized between 0.1 and 0.9.

Since a, b, c, and d are very small values that could be unstable during training – hence leading to physically inconsistent parameters –, we rewrite:

$$s = s_0\tilde{s}, \quad \forall s \in \{a, b, c, d\}$$

where s_0 is the initial value of the parameter. We initialize $\tilde{s} = 1$ and let the backpropagation algorithm modify this much more stable value instead.

Appendix E. Additional results

E.1. Robustness of PCNNs

To further analyze the robustness of the PCNN discussed in Section 5, we trained five other networks with the same structure but different random seeds. As pictured in Fig. 9, all six models provide similar accuracy over the validation set, except at the beginning of the horizon. Two out of the six PCNNs trained indeed showed oscillatory behavior on the first prediction steps, leading to higher errors.

E.2. Learned parameters

To complete the analysis of the PCNN presented in Section 5, we also display the final values of the parameters a, b, c, and d in Table 3. Overall, we see that the parameters do not change much, and the same conclusion was drawn for the other PCNNs trained during our experiments. Out of the six PCNNs plotted in Fig. 9, only two modified the values substantially, even though by a maximum of 10%–15%, and they correspond to the two models showing the worst performance overall.

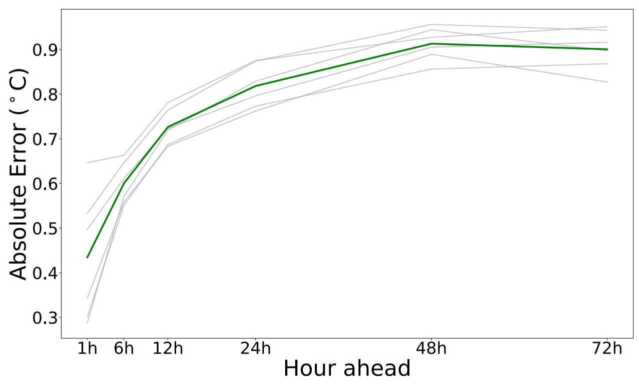


Fig. 9. MAE of six PCNNs with different random seeds at six chosen prediction steps in gray and the average in green, where the statistics were computed from almost 2000 predictions from the validation set..

Table 3
Comparison between the initial and learned values of the PCNN parameters, in degrees Celsius. For a and d, it represents how many degrees are gained in 4 h when heating/cooling at full power, while for b and c it represents how many degrees are lost through heat transfer in 6 h when the exogenous temperature is 25 °C lower.

Parameter	Starting value	Learned value
a	2	2.01
b	1.5	1.50
c	1.5	1.51
d	2	1.97

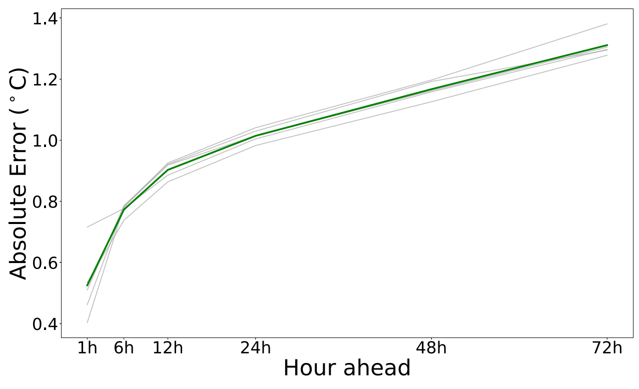


Fig. 10. MAE on the other bedroom in UMAR at key time steps of the prediction horizon for the PCNN with five different random seeds, where the statistics were computed from almost 2000 predictions from the validation set..

E.3. Flexibility of PCNNs

Additionally, to test the flexibility of our approach, we trained five PCNNs on the other bedroom in UMAR, again with five different random seeds. As can be observed in Fig. 10, the models again arrive at a similar accuracy to what was obtained for the first bedroom (see Fig. 9), except towards the end of the prediction horizon, where the error is 20%–40% higher. Nonetheless, the performance of PCNNs is comparable for both rooms, which is particularly interesting since no engineering was required to transfer the model between them: we used the same architecture for both bedrooms, simply changing the training and validation data sets. The training and validation errors displayed in Table 4 confirm these conclusions.

Table 4
Training and validation losses of five PCNNs on the other bedroom in UMAR, scaled by 10^3.

Seed	Training loss	Validation loss
0	1.82	2.42
1	1.66	2.44
2	1.58	2.52
3	1.66	2.54
4	1.66	2.39
Mean	**1.68**	**2.46**

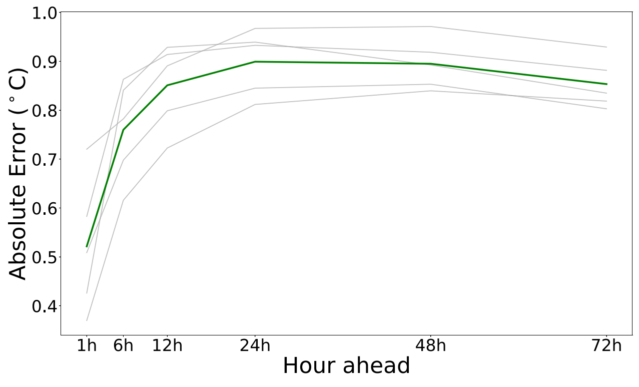


Fig. 11. MAE at key time steps of the prediction horizon for the classical encoder-LSTMs-decoder network with five different random seeds in gray and the average in green, where the statistics were computed from almost 2000 predictions from the validation set.

Table 5
Comparison training and validation loss for five classical LSTMs and six PCNNs, scaled by 10^3.

	Seed	Training loss	Validation loss
LSTMs	0	0.57	2.28
	1	0.57	1.92
	2	1.14	2.30
	3	0.97	2.22
	4	1.00	1.77
	Mean	**0.85**	**2.10**
PCNNs	0	1.83	1.93
	1	1.85	1.65
	2	2.06	1.75
	3	2.28	1.73
	4	1.90	1.97
	5	2.38	1.66
	Mean	**2.05**	**1.78**

E.4. Comparison to classical LSTMs

Finally, to analyze the impact of the prior knowledge inclusion in E, we performed a small ablation experiment by training a classical black-box LSTM network, i.e. only training D with all the inputs in x, including the power and exogenous temperatures, again with five different random seeds. For comparison purposes, the training and validation losses can be found in Table 5 and the error propagation over the validation set in Fig. 11, where one can remark similar to worse performance compared to the proposed PCNNs in Fig. 9. This is a strong indication that PCNNs do not lose much expressiveness, even if we constrain the structure to follow some given laws. On the contrary, the linear physics-informed module inside PCNNs seems to give them useful information, as they are able to beat the performance of classical unconstrained LSTMs.

Appendix F. Deriving PCNNs from classical gray-box models

Classical gray-box models generally start from a linear RC model of the following form [73]:

$$\dot{z}(t) = Az(t) + Bq(t) \tag{35}$$

where z is the state of the system[5] and q captures various heat fluxes like heating/cooling inputs, heat losses to the environment and neighboring zones, or heat gains from solar irradiation. After using Euler's discretization, we obtain the following discrete-time linear model:

$$z_{k+1} = Az_k + Bq_k \tag{36}$$

Using system identification, one can then identify the parameters A and B of this model, yielding a gray-box model.

Traditionally, researchers separate the various heat fluxes in q between *controllable* and *uncontrollable* variables v and w, respectively. The former generally captures power inputs in building models, but it could be extended to include blind controls for example. On the other hand, w captures the effect of the sun, the occupants, and other disturbances that cannot be controlled. In the past, both types of heat fluxes have usually been separated linearly, leading to models of the form [73]:

$$z_{k+1} = Az_k + B_v v_k + B_w w_k \tag{37}$$

However, while some exogenous factors in w do indeed present a linear impact on the zone temperatures, others are much harder to capture, such as the solar gains or the effect of the occupants. One way to capture more complex phenomena is to introduce bilinear terms coupling v and w in Eq. (37), as in Sturzenegger et al. [73] for example. The natural disadvantage arising from such additional coupling terms in control application is that the subsequent optimization of v gets more complicated.

On the other hand, PCNNs effectively separate the uncontrollable variables into two sets w^1 and w^2 using prior knowledge on the law of physics in buildings. The former groups variables known to have a linear impact on the zone temperature, the temperature outside and in the neighboring zones. w^2 then gathers the other inputs with nonlinear effects, which computed in a separate disturbance process ξ based on an unknown residual function m, which yields a process similar to:

$$\begin{aligned} \xi_{k+1} &= \xi_k + m(\xi_k, w_k^2) \\ z_{k+1} &= Az_k + B_u g(u_k) + B_{w^1} w_k^1 + \xi_{k+1} \end{aligned} \tag{38}$$

Note that we introduced a more general form of the controllable inputs $v = g(u)$ since the power inputs v are for example not directly controllable in general. We hence represent them by a function $g(u)$ where u is the true control variable, typically the opening of the valves in the case of radiators. Remarkably, the model presented in Eq. (38) still retains a linear structure with respect to power inputs $v = g(u)$, which is very well-suited for control applications.

However, PCNNs have yet a slightly different structure: the disturbance model ξ is influencing the state of the system both with and without a lag of one time step, as can be observed in Eq. (5) since E_{k+1} depends on D_k. Altogether, we can thus rewrite the equations of the PCNN as follows:

$$\begin{aligned} \xi_{k+1} &= \xi_k + m(\xi_k, w_k^2) \\ z_{k+1} &= Az_k + B_u g(u_k) + B_{w^1} w_k^1 \\ &\quad + B_d \xi_k + \xi_{k+1} \end{aligned} \tag{39}$$

[5] We adopt the unconventional notation z for the state to avoid confusion with the NN inputs x in PCNNs.

One can verify that Eqs. (9) and (39) are equivalent, with:

$$\begin{aligned} \xi &= D & z &= T \\ w^1 &= \begin{bmatrix} T^{out} & T^{neigh} \end{bmatrix}^T & w^2 &= x \\ A &= 1 - b - c & B_u &= a \\ B_{w^1} &= \begin{bmatrix} -b & -c \end{bmatrix} & B_d &= -b - c \end{aligned}$$

References

[1] International Energy Agency (IEA). Tracking buildings 2020. 2020, https://www.iea.org/reports/tracking-buildings-2020 [Accessed: 28 May 2021].

[2] European Commission (EC). Factsheet: The energy performance of buildings directive. 2019, https://ec.europa.eu/energy/sites/ener/files/documents/buildings_performance_factsheet.pdf [Accessed: 06 Jan 2021].

[3] Eurostat, statistics explained. Energy consumption in households. 2020, https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Energy_consumption_in_households [Accessed: 28 May 2021].

[4] United Nations Framework Convention on Climate Change (UNFCCC). Paris agreement to the united nations framework convention on climate change. 2015, https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement [Accessed: 05 May 2021].

[5] Westermann P, Evins R. Surrogate modelling for sustainable building design–a review. Energy Build 2019;198:170–86. http://dx.doi.org/10.1016/j.enbuild.2019.05.057.

[6] Rabani M, Madessa HB, Mohseni O, Nord N. Minimizing delivered energy and life cycle cost using graphical script: An office building retrofitting case. Appl Energy 2020;268:114929. http://dx.doi.org/10.1016/j.apenergy.2020.114929.

[7] 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 2021;118127. http://dx.doi.org/10.1016/j.apenergy.2021.118127.

[8] Boodi A, Beddiar K, Benamour M, Amirat Y, Benbouzid M. Intelligent systems for building energy and occupant comfort optimization: A state of the art review and recommendations. Energies 2018;11(10):2604. http://dx.doi.org/10.3390/en11102604.

[9] Fan C, Xiao F, Zhao Y. A short-term building cooling load prediction method using deep learning algorithms. Appl Energy 2017;195:222–33. http://dx.doi.org/10.1016/j.apenergy.2017.03.064.

[10] Shamsi MH, Ali U, Mangina E, O'Donnell J. A framework for uncertainty quantification in building heat demand simulations using reduced-order grey-box energy models. Appl Energy 2020;275:115141. http://dx.doi.org/10.1016/j.apenergy.2020.115141.

[11] Tian W, Heo Y, De Wilde P, Li Z, Yan D, Park CS, et al. A review of uncertainty analysis in building energy assessment. Renew Sustain Energy Rev 2018;93:285–301. http://dx.doi.org/10.1016/j.rser.2018.05.029.

[12] 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.

[13] Afroz Z, Shafiullah G, Urmee T, Higgins G. Modeling techniques used in building HVAC control systems: A review. Renew Sustain Energy Rev 2018;83:64–84. http://dx.doi.org/10.1016/j.rser.2017.10.044.

[14] Cai M, Pipattanasomporn M, Rahman S. Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques. Appl Energy 2019;236:1078–88. http://dx.doi.org/10.1016/j.apenergy.2018.12.042.

[15] Wang Z, Hong T. Reinforcement learning for building controls: The opportunities and challenges. Appl Energy 2020;269:115036. http://dx.doi.org/10.1016/j.apenergy.2020.115036.

[16] Wan Z, Li H, He H. Residential energy management with deep reinforcement learning. In: 2018 international joint conference on neural networks. IEEE; 2018, p. 1–7. http://dx.doi.org/10.1109/IJCNN.2018.8489210.

[17] Yu L, Xie W, Xie D, Zou Y, Zhang D, Sun Z, et al. Deep reinforcement learning for smart home energy management. IEEE Internet Things J 2019. http://dx.doi.org/10.1109/JIOT.2019.2957289.

[18] Djeumou F, Neary C, Goubault E, Putot S, Topcu U. Neural networks with physics-informed architectures and constraints for dynamical systems modeling. 2021, arXiv preprint arXiv:2109.06407.

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

[20] Wiyatno RR, Xu A, Dia O, de Berker A. Adversarial examples in modern machine learning: A review. 2019, arXiv preprint arXiv:1911.05268.

[21] Moosavi-Dezfooli S-M, Fawzi A, Frossard P. Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, p. 2574–82.

[22] Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, et al. Language models are few-shot learners. 2020, arXiv preprint arXiv:2005.14165.

[23] Xie Q, Luong M-T, Hovy E, Le QV. Self-training with noisy student improves imagenet classification. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 10687–98.

[24] Xu Y, Goodacre R. On splitting training and validation set: a comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. J Anal Test 2018;2(3):249–62. http://dx.doi.org/10.1007/s41664--018--0068--2.

[25] D'Amour A, Heller K, Moldovan D, Adlam B, Alipanahi B, Beutel A, et al. Underspecification presents challenges for credibility in modern machine learning. 2020, arXiv preprint arXiv:2011.03395.

[26] 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.

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

[28] Greydanus S, Dzamba M, Yosinski J. Hamiltonian neural networks. Adv Neural Inf Process Syst 2019;32:15379–89.

[29] von Rueden L, Mayer S, Beckh K, Georgiev B, Giesselbach S, Heese R, et al. Informed machine learning - a taxonomy and survey of integrating prior knowledge into learning systems. IEEE Trans Knowl Data Eng 2021;1. http://dx.doi.org/10.1109/TKDE.2021.3079836.

[30] 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.

[31] Gokhale G, Claessens B, Develder C. Physics informed neural networks for control oriented thermal modeling of buildings. Appl Energy 2022;314:118852. http://dx.doi.org/10.1016/j.apenergy.2022.118852.

[32] Bünning F, Huber B, Schalbetter A, Aboudonia A, de Badyn MH, Heer P, et al. Physics-informed linear regression is a competitive approach compared to machine learning methods in building MPC. 2021, arXiv preprint arXiv:2110.15911.

[33] Homod RZ. Review on the HVAC system modeling types and the shortcomings of their application. J Energy 2013;2013. http://dx.doi.org/10.1155/2013/768632.

[34] 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. http://dx.doi.org/10.1016/j.rser.2013.03.004.

[35] Li X, Wen J. Review of building energy modeling for control and operation. Renew Sustain Energy Rev 2014;37:517–37. http://dx.doi.org/10.1016/j.rser.2014.05.056.

[36] 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. http://dx.doi.org/10.1016/j.rser.2017.02.085.

[37] 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.

[38] Ali U, Shamsi MH, Hoare C, Mangina E, O'Donnell J. Review of urban building energy modeling (UBEM) approaches, methods and tools using qualitative and quantitative analysis. Energy Build 2021;246:111073. http://dx.doi.org/10.1016/j.enbuild.2021.111073.

[39] Wei T, Ren S, Zhu Q. Deep reinforcement learning for joint datacenter and HVAC load control in distributed mixed-use buildings. IEEE Trans Sustain Comput 2019. http://dx.doi.org/10.1109/TSUSC.2019.2910533.

[40] Crawley DB, Lawrie LK, Winkelmann FC, Buhl WF, Huang YJ, Pedersen CO, 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.

[41] Wetter M, Haugstetter C. Modelica versus TRNSYS–a comparison between an equation-based and a procedural modeling language for building energy simulation. Proc SimBuild 2006;2(1).

[42] 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.

[43] Ding X, Du W, Cerpa A. OCTOPUS: Deep reinforcement learning for holistic smart building control. In: Proceedings of the 6th ACM international conference on systems for energy-efficient buildings, cities, and transportation. 2019, p. 326–35. http://dx.doi.org/10.1145/3360322.3360857.

[44] 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. http://dx.doi.org/10.1016/j.enbuild.2019.07.029.

[45] Chakrabarty A, Maddalena E, Qiao H, Laughman C. Scalable Bayesian optimization for model calibration: Case study on coupled building and HVAC dynamics. Energy Build 2021;111460. http://dx.doi.org/10.1016/j.enbuild.2021.111460.

[46] 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. http://dx.doi.org/10.1016/j.enbuild.2016.02.021.

[47] 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. http://dx.doi.org/10.1016/j.energy.2016.10.126.

[48] Zhang C, Li J, Zhao Y, Li T, Chen Q, Zhang X, et al. Problem of data imbalance in building energy load prediction: Concept, influence, and solution. Appl Energy 2021;297:117139. http://dx.doi.org/10.1016/j.apenergy.2021.117139.

[49] 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. IEEE; 2016, p. 1–6. http://dx.doi.org/10.1109/EEEIC.2016.7555567.

[50] Rahman A, Smith AD. Predicting heating demand and sizing a stratified thermal storage tank using deep learning algorithms. Appl Energy 2018;228:108–21. http://dx.doi.org/10.1016/j.apenergy.2018.06.064.

[51] Bünning F, Schalbetter A, Aboudonia A, de Badyn MH, Heer P, Lygeros J. Input convex neural networks for building MPC. In: Proceedings of the 3rd conference on learning for dynamics and control. Proceedings of machine learning research, vol.144, PMLR; 2021, p. 251–62.

[52] Li X, Yao R. Modelling heating and cooling energy demand for building stock using a hybrid approach. Energy Build 2021;235:110740. http://dx.doi.org/10.1016/j.enbuild.2021.110740.

[53] Gray FM, Schmidt M. A hybrid approach to thermal building modelling using a combination of Gaussian processes and grey-box models. Energy Build 2018;165:56–63. http://dx.doi.org/10.1016/j.enbuild.2018.01.039.

[54] Fux SF, Ashouri A, Benz MJ, Guzzella L. EKF based self-adaptive thermal model for a passive house. Energy Build 2014;68:811–7. http://dx.doi.org/10.1016/j.enbuild.2012.06.016.

[55] 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.

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

[57] Bünning F, Huber B, Schalbetter A, Aboudonia A, de Badyn MH, Heer P, et al. Physics-informed linear regression is competitive with two machine learning methods in residential building MPC. Appl Energy 2022;310:118491. http://dx.doi.org/10.1016/j.apenergy.2021.118491.

[58] Maasoumy M, Pinto A, Sangiovanni-Vincentelli A. Model-based hierarchical optimal control design for HVAC systems. In: Dynamic systems and control conference, Vol. 54754. 2011, p. 271–8. http://dx.doi.org/10.1115/DSCC2011-6078.

[59] 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. http://dx.doi.org/10.1016/j.enbuild.2014.03.057.

[60] 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. http://dx.doi.org/10.1109/ACC.2014.6858875.

[61] Kayhan OS, Gemert JCv. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 14274–85.

[62] Yu Y, Si X, Hu C, Zhang J. A review of recurrent neural networks: LSTM cells and network architectures. Neural Comput 2019;31(7):1235–70. http://dx.doi.org/10.1162/neco_a_01199.

[63] Karpatne A, Atluri G, Faghmous JH, Steinbach M, Banerjee A, Ganguly A, et al. Theory-guided data science: A new paradigm for scientific discovery from data. IEEE Trans Knowl Data Eng 2017;29(10):2318–31. http://dx.doi.org/10.1109/TKDE.2017.2720168.

[64] Raissi M, Perdikaris P, Karniadakis GE. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. 2017, arXiv preprint arXiv:1711.10561.

[65] Raissi M, Perdikaris P, Karniadakis GE. Physics informed deep learning (part II): Data-driven solutions of nonlinear partial differential equations. 2017, arXiv preprint arXiv:1711.10561.

[66] Yang Y, Perdikaris P. Physics-informed deep generative models. 2018, arXiv preprint arXiv:1812.03511.

[67] Yuan J, Weng Y. Physics interpretable shallow-deep neural networks for physical system identification with unobservability. 2021.

[68] Hu X, Hu H, Verma S, Zhang Z-L. Physics-guided deep neural networks for power flow analysis. IEEE Trans Power Syst 2020;36(3):2082–92. http://dx.doi.org/10.1109/TPWRS.2020.3029557.

[69] Werbos PJ. Backpropagation through time: what it does and how to do it. Proc IEEE 1990;78(10):1550–60.

[70] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, dÁlché Buc F, Fox E, Garnett R, editors. Advances in neural information processing systems, Vol. 32. Curran Associates, Inc.; 2019, p. 8024–35.

[71] Empa. NEST. 2021, https://www.empa.ch/web/nest/overview [Accessed: 04 Oct 2021].

[72] Hewing L, Kabzan J, Zeilinger MN. Cautious model predictive control using gaussian process regression. IEEE Trans Control Syst Technol 2019;28(6):2736–43. http://dx.doi.org/10.1109/TCST.2019.2949757.

[73] Sturzenegger D, Gyalistras D, Morari M, Smith RS. Model predictive climate control of a swiss office building: Implementation, results, and cost–benefit analysis. IEEE Trans Control Syst Technol 2015;24(1):1–12. http://dx.doi.org/10.1109/TCST.2015.2415411.