The BRIGHT project is co-founded by the EU's Horizon 2020 innovation programme under grant agreement No 957816

Boosting DR through increased communIty-level consumer engaGement by combining Data-driven and blockcHain technology Tools with social science approaches and multi-value service design

Deliverable D4.5 Electrical and thermal communities DTs' models – first version

Authors: Matthias Strobbe (imec), Gargya Gokhale (imec), Evelyn Heylen (Centrica), Bert Claessens (Centrica), Giannis Kazdaridis (domX)

The project Boosting DR through increased communIty-level consumer engaGement by combining Data-driven and blockcHain technology Tools with social science approaches and multi-value service design (BRIGHT) has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957816. The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the Innovation and Networks Executive Agency (INEA) or the European Commission (EC). INEA or the EC are not responsible for any use that may be made of the information contained therein.

Imprint

Title:	**Electrical and thermal communities DTs' models – first version**
Contractual Date of Delivery to the EC:	31.01.2022
Actual Date of Delivery to the EC:	31.01.2022
Author(s):	Matthias Strobbe (imec), Gargya Gokhale (imec), Evelyn Heylen (Centrica), Bert Claessens (Centrica), Giannis Kazdaridis (domX)
Participant(s):	IMEC, CEN, DOMX
Project:	Boosting DR through increased communIty-level consumer engaGement by combining Data-driven and blockcHain technology Tools with social science approaches and multi-value service design (BRIGHT)
Work Package:	WP4 – Community and Customer Digital Twin Models
Task:	T4.5 – Digital Twins for electrical and thermal communities
Confidentiality:	Public
Version:	1.0

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

API	Application Programming Interface
ARIMA	Autoregressive Integrated Moving Average
BRIGHT	Boosting DR through increased communIty-level consumer engaGement by combining Data-driven and blockcHain technology Tools with social science approaches and multi-value service design
DB	Database
DR	Demand-Response
JWT	JSON Web Token
MAE	Mean Absolute Error
MDP	Markov Decision Process
ML	Machine Learning
MLP	Multilayer Perceptron
MPC	Model Predictive Control
RC	Resistor Capacitor
ReLU	Rectified Linear Units
RL	Reinforcement Learning
REST	Representational State Transfer
RMSE	Root Mean Squared Error
WP	Work Package

Table 1 List of Acronyms and Abbreviations

Executive Summary

The objective of T4.5 is to develop and evaluate digital twins models pertaining to the electrical and thermal flexibility available in residential energy communities. With these digital twins more optimal flexibility services can be developed in WP5.

This deliverable presents the first version of digital twin models for different thermal assets:

- A data-driven modeling approach using 2 variants of physics informed neural networks for the task of control-oriented thermal modeling of buildings.
- A hybrid digital twin model for domestic hot water tanks using also a combined approach of a physical model of the dynamics of the temperature of the tank and a data-driven model.
- A neural networks based approach to model and forecast the behavior and performance of thermal assets (heating system, building and occupants) for diverse household heating scenarios.

The used machine learning (ML) techniques for modelling these assets are described as well as initial training and validation results.

1. Introduction

To realize efficient demand response algorithms to valorize the flexibility of flexible loads, accurate models are needed of (groups of) these controllable loads. In this deliverable we present several digital twin models for flexible loads in residential energy communities, to be used in WP5 to deliver flexibility services on community and system level.

These digital twins have in common that they use data driven techniques for generating the models, often in combination with physics-based models (hybrid or grey-box approach). This results in generalizable and interpretable models that can be trained with relatively moderate sized datasets.

1.1. Purpose

This deliverable presents the first version of the developed digital twin models. It describes in detail the used machine learning (ML) techniques for modelling flexible thermal assets (space heating via district heating system, electric domestic hot water boiler, gas-based boiler for space heating and domestic hot water) and initial training and validation results. In the second version of this deliverable (to be released in M30) further optimized models will be presented, trained with larger datasets, a.o. from the BRIGHT pilots.

1.2. Relation to Other Activities

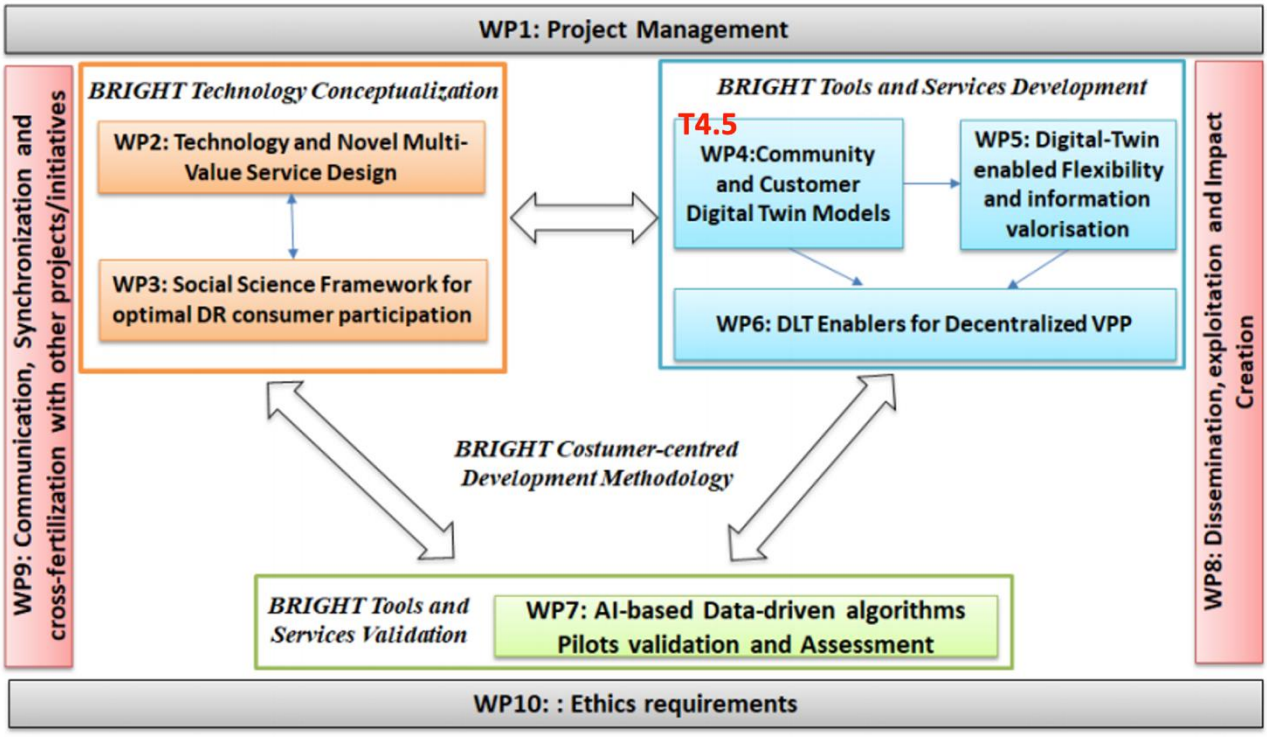


Figure 1. Relation of T4.5 to other BRIGHT activities

This deliverable is the output of Task 4.5 of work package 4 (Figure 1). It uses the outputs from WP2 and WP3 on requirements, use cases, and drivers and barriers for consumer engagement as input for the development of the digital twins. These models will be then be used in WP5 to realize

flexibility-based services on community and system level. The models in combination with these demand response services will be tested and validated in the BRIGHT pilots as part of WP7.

1.3. Structure of the Document

This deliverable is organized as follows:

- Section 2 presents currently developed digital twin models for thermal loads in energy communities. The used (hybrid) ML techniques to develop these models are discussed in detail.
- Section 3 presents the first results on the training and validation of these models and information on which datasets and pilot data will be used for the further optimization of the digital twins in the remainder of the project.
- Section 4 presents the conclusions.

2. Digital Twin Models for Energy Communities

2.1 Building Thermal Model

2.1.1 Introduction

Buildings account for 40% of the total primary energy consumption worldwide. Therefore improving their energy efficiency and shifting to the use of renewable energy sources is very important. To realize this energy efficiency improvements and provide the necessary flexibility to cope with the intermittency of solar and wind energy sources, smart control algorithms for the buildings' energy consumption will be crucial in the energy transition process.

Significant research has been carried out in the context of control algorithms for energy management in buildings, ranging from simple Rule-based Controllers to advanced controllers like Model Predictive Control (MPC) and Reinforcement Learning (RL) [1] . In MPC, a physical model of the system is used to anticipate the future behavior of the system and optimize its performance [2] . This enables MPC-based controllers to be sample efficient and produce interpretable control decisions. However, the accuracy of MPC is closely related to the fidelity of the model, which is often difficult to obtain for real-world scenarios [3] .

Contrary to this, data-driven controllers like RL, work directly with past interactions between the system, without the need for explicit physics knowledge. Although these RL-based controllers have shown promising results, they present a black-box solution that requires large amounts of training data. Additionally, in previous work such as [4] , an RL controller was trained using a physics model-based simulator to ensure that the training data obtained was sufficiently diverse and to avoid taking harmful exploration actions.

This makes obtaining accurate building models a crucial requirement for developing better control algorithms. A variety of modeling techniques have been studied previously and are broadly classified into physics models (white box, grey box) and data-driven models (black box) [5] . The physics models involve solving a system of partial differential equations based on the underlying physical laws, commonly achieved using numeric solvers such as EnergyPlus, Modelica, as presented in, e.g., [6] [7] . The use of such models however has been limited in the control domain, primarily due to the high computational cost associated with solving the underlying system of partial differential equations [6] . Alternatively, a lumped parameter model using resistive and capacitive networks is used for control-oriented modeling. With this framework, different thermal components in a building are modeled using a RC network and simplified to obtain a lower order model that is easier to solve. However, even with these approximations, the models obtained are highly specific and require significant modeling effort as demonstrated in [8] .

Data-driven models circumvent these modeling challenges by relying completely on obtained data. Previously, techniques such as ARIMA, Genetic Algorithms, Neural Networks, etc., have been studied and have shown good modeling capabilities [5] [9] . Yet, as discussed in [5] , these techniques have their own challenges in the form of huge training data requirement and lack of interpretability.

To get the best of both these worlds, in our research we incorporate self-learning, physics guided models with model-based reinforcement learning algorithms to develop interpretable control agents in a data-driven manner. We work with Physics Informed Neural Network architectures to learn physically relevant control-oriented models of real-world systems. This is achieved by explicitly providing information related to the underlying physics of the system to a deep neural network during the training procedure.

2.1.2 Related Work

Building Control and Modeling

Extensive research has been carried out previously in this domain, with work such as [1] [10] presenting exhaustive reviews of different control algorithms. MPC has emerged as an established control technique with works such as [11] [12] [13] presenting case studies for practical implementations in real-world buildings. These works show that MPC-based control strategies can lead to cost savings of about 20% compared to the rule-based control algorithms. An MPC strategy involves a physical model of the system and a set of constraints to formulate a receding horizon optimization problem that is solved at every time step to obtain optimum control actions [2] . Authors in [12] [13] utilize a grey-box RC model for the buildings. This leads to a bi-linear building model and results in a non-linear optimization problem that can still be solved with reasonable accuracy using a sequence of linear programs [13] . Solving this optimization problem is computationally expensive and can limit the practical applicability of MPC controllers. Besides this, MPC controllers are expensive to obtain as indicated in [13] , whose authors conducted a cost-benefit analysis of using MPC-based control strategies in real-world buildings. They concluded that, while MPCs can lead to a decrease in operating costs, the investment costs are much higher, primarily due to higher costs associated with the modeling of buildings, thus prohibiting widespread commercial application. Further, these building models are seldom scalable and need to be developed for individual buildings. E.g., in [8] , authors discuss the model identification process for a real-world building and present a procedure for estimating building parameters. This procedure leads to models with accurate multi-step temperature predictions (0.3°C prediction error). However, this identification process involves solving a quadratic program to obtain good initial estimates of the building parameters, followed by solving a multi-step prediction optimization problem to obtain the final model. This highlights the high computational requirements for obtaining a good building model and the lack of scalability. Additionally, the modeling approaches presented above approximate the non-linear thermal dynamics of the building using a first-order Euler discretization. This leads to approximate models which can be susceptible to errors and lead to biased control actions, as discussed in [6] [14] .

Data-driven control techniques circumvent aforementioned shortcomings of MPC by completely relying on collected data as presented in [15] . Owing to the recent success of works such as [16] , Reinforcement learning-based controllers are gaining importance in developing controllers for buildings [17] . RL-based controllers are self-learning controllers that use data collected from past interactions between the system and the controller to learn the dynamics of the system and achieve a predefined objective [18] . Works such as [4] [19] have studied RL-based controllers in the context of building control and show that such RL controllers can lead to 5 − 12% energy savings compared to rule-based controllers. Additionally, [20] compares the performance of MPC and RL controllers to show that RL controllers are able to outperform a linear MPC-based controller for two different test scenarios. Though these works indicate promising results for RL-based controllers, they also

highlight existing challenges in real-world deployment of RL. These include the large training data requirement, lack of interpretability, need for safe explorations, etc. [21] . E.g., in [19] , the authors use one year of data for training the RL controller using random explorations. Similarly, in [4] , the authors use 2 months of temperature data for obtaining a training data size equivalent to 3000 simulated trajectories. This data intensive nature and need for significant exploration represents a common challenge faced by RL-based control strategies. Hybrid control approaches have been studied to mitigate some of these problems by combining domain knowledge with these RL controllers [14] [22] . E.g., in [22] , the authors present a hybrid control strategy by merging model-free and model-based control strategies. They propose an aggregate-and-dispatch control framework for a cluster of water heaters in which an MPC controller calculates energy set-points for the cluster and the dispatch is carried out based on a fitted Q-iteration RL strategy.

Our approach is different: instead of using a model of the system directly, we focus on learning this model using the available data and then using it in a model-based RL approach. While different techniques for this control-oriented modeling problem have been studied previously, these techniques were focused on creating convex, linear (or bi-linear), time invariant models compatible with MPC formulation and available optimization solvers [23] [24] . In contrast, our objective is to learn a low dimensional, latent space dynamics model of the system to use in RL, where these latent representations can be used to learn optimum control policies as demonstrated in [25] [26] . Concretely, we use Physics Informed Neural Network architectures [27] .

Physics Informed Neural Network Architectures

As introduced in [27] , Physics informed neural networks represent a novel class of neural network architectures where prior knowledge about the system is encoded explicitly in the architecture. This work is similar to [28] , where inductive biases based on the underlying physics laws are coded directly into the network. Several works have built upon this idea and have shown promising results in obtaining approximate solutions for difficult physics problems such as two body mechanics [29] and heat transfer [30] . In [27] [30] , the encoded physics knowledge is strictly enforced on the predictions of the neural network and assumes the availability of complete physics. Differing slightly from this approach, in [29] , the authors enforce partially known physics and learn remaining physics parameters using the available data. These approaches show that trained models are better at extrapolating and require fewer training samples.

Consistent with these works, we use physics informed neural networks for modeling the thermal behavior of a building. However, instead of strictly enforcing prior knowledge, we guide the network to maximally adhere to the underlying physics. Differing from work presented in [31] , we encode the physics directly by using neural network outputs to calculate additional physics-based losses. Additionally, we propose to extract low-dimensional latent representations which correspond to the hidden states of the system and use prior physics knowledge to guide them towards a physically interpretable space. This ensures that the obtained latent representations are disentangled, as opposed to the unsupervised learning cases discussed in [32] [33] . Once trained, these physics informed neural network models can be used with model-based RL algorithms such as MuZero [25] and Dreamer [26] to obtain optimum control actions for building control.

2.1.3 Mathematical Model

In this section we will formulate the Physics-informed Neural Networks for modeling the thermal behavior of a household.

Problem Formulation

A Markov Decision Process (MDP) is a commonly used framework to model sequential decision-making problems [18] . An MDP consists of 4 main components: state space (X), action space (U), state transition function (f) and reward function (ρ). In a fully observable setting, the transition function $f: X \times U \times W \rightarrow X$ represents the true mapping at time step i, between the current state (x_i), the current action (u_i), an exogenous parameter (w_i) and the next state (x_{i+1}) of the system and is given as:

$$x_{i+1} = f(x_i, u_i, w_i) \qquad (1)$$

Here, w_i represents the stochasticity in the system and is assumed as an independent random variable. The transition function (f) represents the true dynamics of the system and to obtain a control-oriented model, it is necessary to approximate this transition function. For data-driven methods, this reduces the problem into a supervised learning problem with the objective of estimating the transition function using a labeled set of state transitions $F = \{(x_1, u_1, w_1, x_2), \dots, (x_N, u_N, w_N, x_{N+1})\}$.

E.g., a neural network with parameters θ can be trained to solve the following optimization problem:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} (x_{i+1} - f(x_i, u_i, w_i))^2 \qquad (2)$$

With this conventional approach, the neural network learns to estimate the transition function by fully relying on the training dataset, without explicitly learning the underlying physical relationships. It should be noted that Eq. (2) represents the scenario for a fully observable system, where complete state information is known and can be used to obtain predictions for the next states. However, a real-world system, such as a thermal model for a building, is generally partially observable where some state parameters cannot be measured or obtained directly. In such cases, using Eq. (2) directly is not useful as the observed states lack complete information. To mitigate this, [18] presents different approaches, one of which involves engineering new high-dimensional features based on the observed states. E.g., in case of a thermal model for a building, the observed state can include measurements of room temperature or actual power consumption, whereas a hidden state parameter can be the temperature of building thermal mass (e.g., walls, furniture, etc.) which is difficult to measure or estimate accurately. Accordingly, to compensate for this missing state parameter, a sequence of past room temperature measurements can be used instead of a single room temperature measurement, and this corresponds to an engineered feature for mitigating the partial observability of this system.

For such partially observable MDPs, the state space (X) consists of an observable component (X^{obs}) and a feature engineered component (X^f) such that $X = X^{obs} \times X^f$. With this high dimensional state representation, a neural network can be trained to estimate the next observable state (x_{k+1}^{obs})

given state, action, and other exogenous inputs, thus modifying the optimization problem in Eq. (2) as:

$$\hat{x}_{i+1}^{obs} = f_\theta(x_i, u_i, w_i),$$
$$\min_\theta \frac{1}{N}\sum_{i=1}^{N}(x_{i+1}^{obs} - f(x_i, u_i, w_i))^2 \qquad (3)$$

Aside from this data-driven approach, for problems where the physics of the system are known a priori, the system dynamics can be approximated using a system of ordinary or partial differential equations that relate the observable states (x_i^{obs}), actions (u_i), other exogenous factors (w_i), hidden state parameters (z_i) and system parameters (Ω). This is represented using a generic differential operator (D_Ω) as:

$$D_\Omega(x_i, u_i, z_i, z_{i+1}, w_i) = 0 \qquad (4)$$

In the following section, we present the physics informed neural network architectures, which combine Eq. (3)–(4) to obtain control-oriented models for systems where the underlying physics are known a priori.

Physics Informed Neural Network Architectures

Consistent with previous works such as [27] [29] , we explicitly encode the underlying physics of the system in a standard neural network architecture and then train it on the data collected. Assuming a partially observable setting, the network is trained to predict the next observable state (x_{i+1}^{obs}) and a latent representation (z_i) using a high dimensional state input (x_i), action (u_i) and exogenous information (w_i). This is done by setting up a constrained optimization problem based on Eq. (3) as:

$$\min_{\theta,\Omega} \frac{1}{N}\sum_{i=1}^{N}(x_{i+1}^{obs} - \hat{x}_{i+1}^{obs})^2 \qquad (5)$$

$$s.t. \quad D_\Omega(x_i, u_i, \hat{z}_i, \hat{z}_{i+1}, w_i) = 0,$$

$$\forall \left(x_i, u_i, w_i, x_{i+1}^{obs}\right) \in F$$

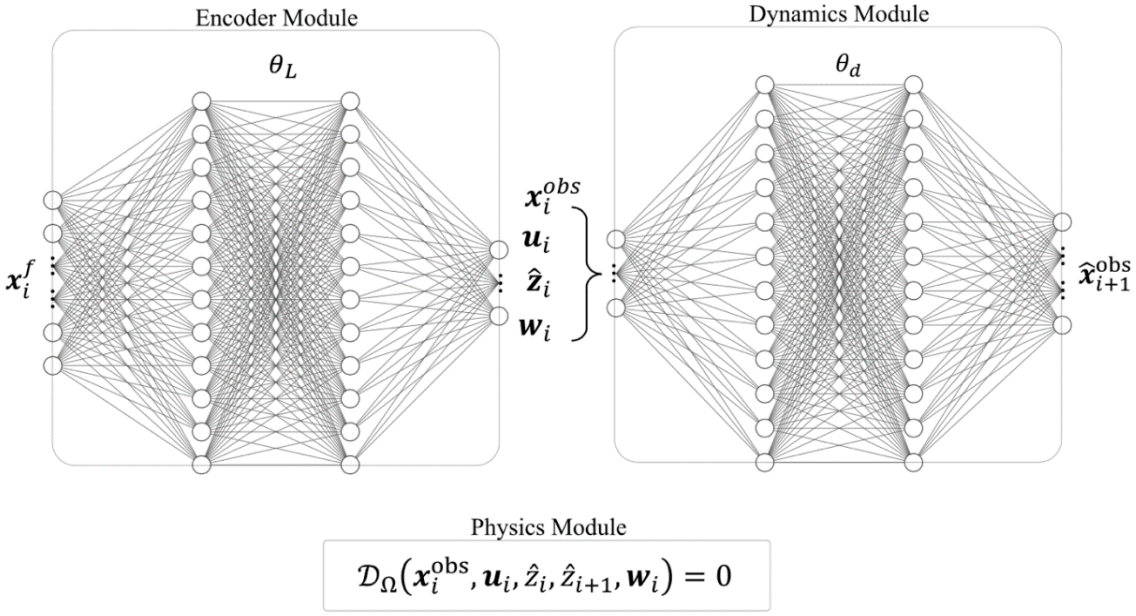
To solve this optimization problem, we define a loss function composed of two terms: L_{reg} represents the mean squared error loss for regression and L_{phys} represents the physics-based loss that makes the network adhere to the underlying physics. This is formulated as:

$$Loss = L_{reg} + \lambda L_{phys}$$

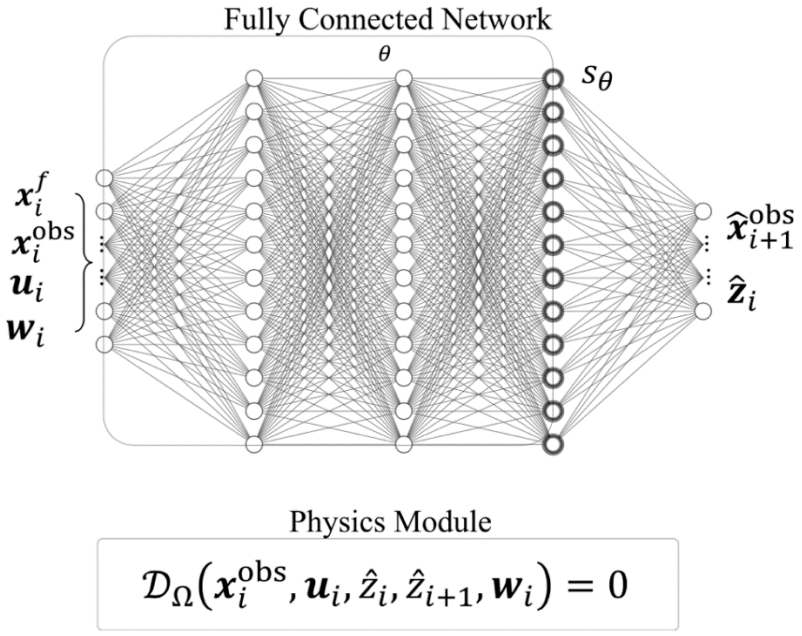
$$L_{reg} = \frac{1}{N}\sum_{i=1}^{N}(x_{i+1}^{obs} - \hat{x}_{i+1}^{obs})^2 \qquad (6)$$

$$L_{phys} = \frac{1}{N}\sum_{i=1}^{N}(D_\Omega(x_i, u_i, \hat{z}_i, \hat{z}_{i+1}, w_i))^2$$

The influence of the physics-based loss term is regulated using λ.



(a) PhysNet



(b) PhysReg MLP

Figure 2. Physics Informed Neural Network architectures.

Based on this formulation, we propose two variants of Physics Informed Neural Networks as shown in Figure 2. The proposed networks have different architectures but are both trained based on the methodology described in Eq. (5)–(6). The top subfigure presents an architecture comprising two modules, an Encoder and a Dynamics module. The encoder module is parameterized by θ_L and the dynamics module is parameterized by θ_d. The encoder module creates a bottleneck and encodes the high dimensional, feature engineered component of state inputs (x_i^f) into a low dimensional latent representation ($\hat{z}_i$). This latent representation along with observable state information (x_i^{obs}), action (u_i) and other exogenous information (w_i) are then used by the dynamics module of the network to predict the next observable state (x_i^{obs}) of the system. Thus, a forward pass of this network can be expressed as:

$$\hat{z}_i = g_{\theta_L}(x_i^f),$$

$$\hat{x}_{i+1}^{obs} = h_{\theta_d}(\hat{z}_i, x_i^{obs}, u_i, w_i) \qquad (7)$$

With this architecture, the prediction for next observable state depends on, among other parameters, the prediction of the latent representation ($\hat{z}_i$). This ensures that the encoded representation obtained from this network contains information regarding the dynamics of the system and can be leveraged in model based RL algorithms such as [25] [26] .

Differing slightly from this approach, the bottom subfigure presents a conventional fully connected neural network architecture where physics knowledge is incorporated based on Eq. (5)–(6). The inputs of this architecture comprise of the full state representation ($x_i = (x_i^f, x_i^{obs})$), action and exogenous information. With these inputs, the network predicts the next observable state and a latent representation simultaneously. Thus, there is explicit parameter sharing between these predictions and these shared parameters are represented by θ. The output layer uses an identity activation function and hence the outputs ($\hat{x}_{i+1}^{obs}$ and $\hat{z}_i$) are linear combinations of output of the last hidden layer of the network (s_θ). Thus, a forward pass of this network can be formulated as:

$$\hat{z}_i = g_1 s_\theta(x_i, u_i, w_i) + g_2,$$

$$\hat{x}_{i+1}^{obs} = h_1 s_\theta(x_i, u_i, w_i) + h_2 \qquad (8)$$

where g_1, g_2, h_1 and h_2 are matrices of appropriate dimensions. With this parameter sharing, the obtained latent representation does not contribute to obtaining the predictions for the next observable state. Consequently, this latent representation may not contain sufficient information about the dynamics of the system. However, due to the physics, the latent representation is physically relevant and represents the hidden parameters of the system. Hence, in this case, the physics module acts as a regularization term guiding the network to learn the dynamics of the system and some latent representations simultaneously. This can be leveraged by using this architecture in deep Q-networks [34] to improve the learning performance of the Q-network and extract additional insights in the form of physically relevant latent states.

2.2 Domestic Hot Water Model

This section presents a digital twin to obtain control policies for thermostatically controlled loads, such as hot water tanks used for domestic hot water usage. Centrica has data available of a portfolio of hot water tanks. These tanks should be controlled in such a way that the comfort of the end-users is guaranteed at minimal cost. A hybrid digital twin model is presented that is in line with physics informed neural networks. The hybrid digital twin model combines a physical model of the dynamics of the temperature of the tank with a data-driven model. Therefore, it is in line with the physics-informed neural network model presented earlier in this deliverable.

The physical, dynamic model of the temperature in the tank looks as follows:

$$M_s C_p \Delta T_k = q_k^{HP} - Q_k^{cons} - UA(T_k - T_k^A)$$
$$T_{k+1} = T_k + \Delta T_k$$

Where q_k^{HP} the heating power to heat the water in the tank, T_k^A the ambient temperature of the temp, T_k the temperature of the water in the tank, M_s the thermal mass of the water in the tank, C_p the specific heat constant of the water in the tank, A the outer surface area of the tank, U the thermal transmittance of the outer surface area of the tank and Q_k^{cons} the thermal power consumed by domestic hot water usage. This physical equation represents a linear transition function. The last two terms in the function are unknown or hard to measure directly. Nevertheless, measurement data are available for the water temperature of the tank T_k and the heating power q_k^{HP}. Based on these data, we can derive the model for the tank temperature. The decision variable in this model is the variable q_k^{HP}, which determines the power of the heater.

We construct a hybrid model that combines the linear physical transition function and a black box neural network model $f_\lambda\left(D_k,\ H_k, dT_k^{(1)}, dT_k^{(2)}, T_k\right)$ to model the domestic hot water usage and the tank losses:

$$T_{k+1} = \alpha . T_k + C . q_k^{HP} - L_k$$
$$L_k = Q_k^{cons} - UA(T_k - T_k^A) = f_\lambda\left(D_k,\ H_k, dT_k^{(1)}, dT_k^{(2)}, T_k\right)$$

The black box neural network model modelling the thermal load uses the day of the week (D_k), the hour of the day (H_k), the temperature differences at the previous time instants $dT_k^{(1)}$ and $dT_k^{(2)}$ and the tank temperature at the given time instant (T_k) as inputs. λ represent the parameters of the multilayer perceptron model.

2.3 Thermal Loads & Assets Dynamic Simulator

The Dynamic Simulator tool will enable the automated performance simulation of thermal assets for household heating scenarios, by generating the digital twin representation of their core components (heating system, building, occupants). The tool's purpose is to generate synthetic time-series that reflect the real-world behaviour of the considered set of thermal assets under different scenarios and/or constraints. The outcomes of such analysis are particularly useful when comparing the performance of the considered set of assets between different configurations, i.e., when trying to quantify the potential improvement when replacing for instance the heating system with another, or when trying to evaluate the performance of a considered configuration under different thermal loads or other circumstances and specialized use cases, e.g., night use and day use.

The tool runs on demand, taking as input historical data collected from:

- heating system data (boiler water temperature, etc.);
- energy consumption data as captured from the attached domX heating controller;
- user heating preferences as captured by the domX smartphone application and the room thermostat;
- indoor and outdoor environmental data (temperature, humidity, etc.) as captured by various connected sensors and web services.

Currently a forecasting pipeline has been developed in order to support the modelling of thermal assets, like gas consumption, using statistical and machine learning models to simulate linear behaviours and deep neural networks to learn non-linear time-dependent behaviour out of energy (e.g., boiler gas consumption) and non-energy data (e.g., boiler operating temperature).

After the forecasted time-series are generated, they are stored in a time-series database and are accessible for integration with other components through a web API. The output can also be integrated with a custom dashboard for visualizing the forecasted values as well as the various monitored parameters (e.g., indoor temperature, boiler consumption) across time.

Specifically, a custom web-based graphical user interface has been developed by combining the Vue.js[1] and Grafana[2] frameworks for visualizing the outputs of the thermal load simulation tool. Vue.js is an open-source front-end JavaScript framework for building user interfaces and single-page applications. Grafana is an open source analytics and interactive visualization web application which can be connected to data source and display data analytics using charts, graphs, and alerts. Real-time data collected from the heating controllers are integrated using the MQTT message broker and stored in the Influx time-series DB. Pandas, Statsmodels and TensorFlow are used to handle the data and train the forecasting models. The simulation results are stored in the Influx time-series DB and MySQL DB, exposed over a REST API while at the same time employing JWT tokens for guaranteeing secure data exchange with other services (Figure 3).

[1] https://vuejs.org/
[2] https://grafana.com/

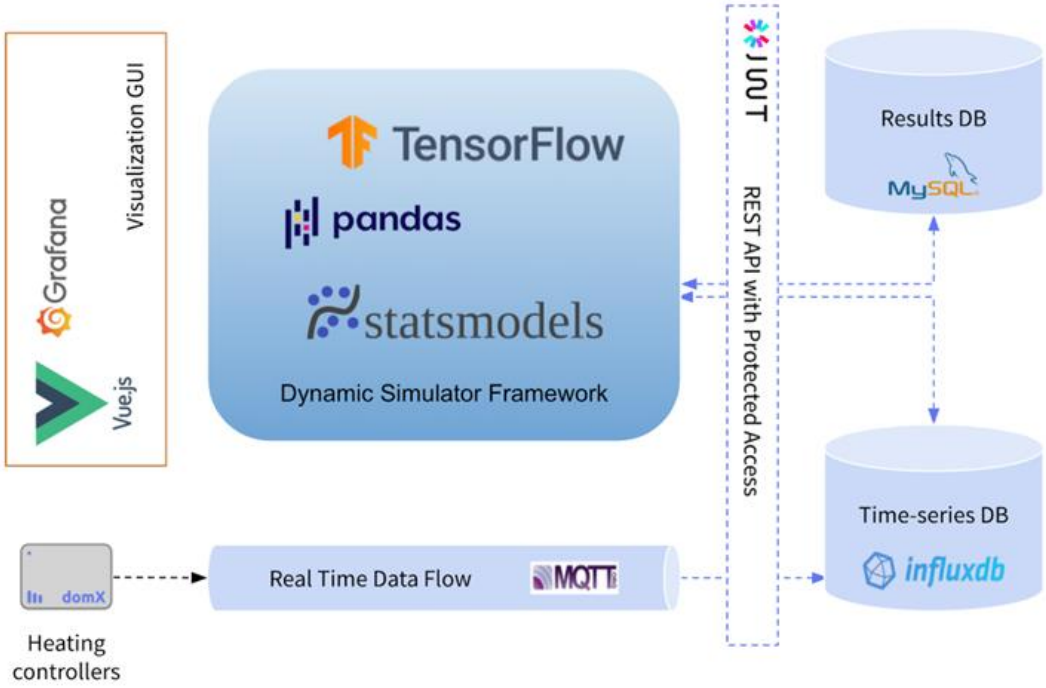


Figure 3. Thermal Loads & Assets Dynamic Simulator high-level architecture.

2.3.1. Neural networks for consumption forecasting

Initially we have experimented with the development of efficient pipelines for training and evaluating time-series forecasting models which in this case are tuned to the prediction of energy consumption levels in future time steps. This is an important processing step and will be the core functionality which will subsequently allow the creation of digital twin models for every desired configuration of thermal assets in future versions. Time-series forecasting makes use of the best fitting model essential to predicting the future values based on complex processing of current and previous observations. In general, time-series data are usually composed from time dependent components which are related to:

- Trends: describe increasing or decreasing behaviours of the time-series and are frequently presented in linear modes;
- Seasonality: highlight repeating patterns of cycles of behaviour over time;
- Irregularity/Noise: regard the non-systematic aspect of time series deviating from the common model values;
- Cyclicity: to identify the repetitive changes in the time series and define their placement in the cycle.

Statistical analysis and mathematical modelling can be used to analyse and discover whether the aforementioned components exist in order to produce models for the classification and forecasting on time series problems and can be especially effective in cases where some prior knowledge for the system is available, usually in the form of physical or mathematical properties and constraints and can be manually incorporated to the modelling pipeline. In addition, these classical methods can perform well on a wide range of problems especially when certain assumptions for the data can be made beforehand in order to suitably prepare the data and configure the method, e.g., in the case of known linear relationships in the data.

While the heating systems we are aiming to model are physical systems and therefore abide to the laws of physics and to physical constrains, they are of proprietary manufacturing origin and most of the system components and operations are unknown. Even though there are some high-level elements known for all heating systems and could be incorporated to mathematical or physical models, to create grey box models the analysis of data observations would still be required to figure out to model the unknown physical properties of the system. Instead, our approach is purely data-driven and takes advantage of the elevated capacity of deep neural networks to learn the dynamics of the system through training with an abundance of data observations.

Specifically, we experimented with a single-layer linear neural network, a 3-layer fully-connected (dense) neural network and a 1D convolutional neural network to carry out the forecasting tasks. A typical neural network (Figure 4) is composed of layers of neurons which are stacked one after another and are connected with weights. Each neuron performs initially a weighted average of its inputs, and then filters the output using an activation function which can be a simple linear or non-linear function. The produced output is further forwarded as input to the neurons of the next layer. The connections between neurons carry learnable weights which define which inputs will be mostly activated when passed through neurons. In this way, after the network has learnt optimal weights for all connections, it can function as a very complex network of simple elements. In order to learn optimal weights, the output of the final layer is fed to a loss function which defines the optimization goal of the learning process. The actual learning is done in an iterative manner. In the beginning of a cycle inputs are first fed through the network (forward pass) and the loss is calculated. Then with an algorithm called backpropagation, the networks' error is used to tune its weights to follow a correction course with a process called stochastic gradient descend. The difference between dense networks and convolutional networks is that in the first category, the neurons are all connected with each other (there exist a weight for every pair of neurons), while convolutional networks are based on a shared-weight architecture, where the weights for each layer are stored in convolution kernels or filters that slide along input features and provide translation equivariant responses known as feature maps.

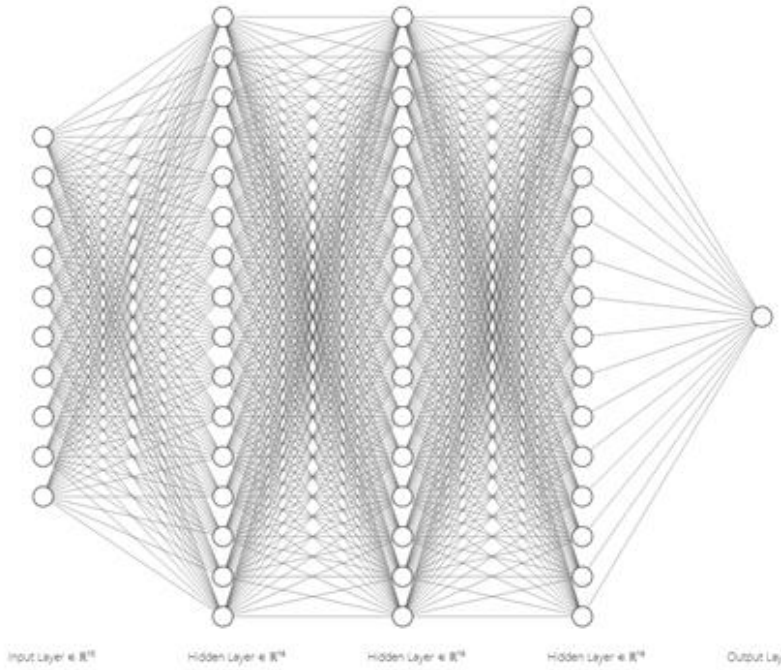


Figure 4. A typical 3-layer dense neural network. The first layer is the input layer, then follow 3 hidden layers and the final layer is producing a single output value. Layers are fully-connected which means that every neuron takes input from all neurons of the previous layer.

In order to build a neural network, certain design choices have to be considered which are related to the number of layers, the depth of each layer (number of neurons), the activation function, the loss function and the parameters of the learning algorithm. The learning rate is an important parameter of the algorithm for example, which defines the aggressiveness at which the correction of weights will be performed. Using a small learning rate will take more time to converge to an optimal solution, but with a large learning rate the solution may start deviating and never converge to a minimum. Training can also be achieved in multiple ways: online training requires a forward and a backward pass for every single observation while minibatch training can calculate the cumulative loss of a batch of observations and perform the backward pass once to correct the weights.

In all experiments we performed minibatch training, by passing 10000 random samples from the training set at once after each iteration. After a certain number of iterations, the network has seen every data observation in the training set at least once and is considered to have finished an epoch of training. We let the training run for 500 epochs as a maximum limit with early stopping activated, which is a technique that monitors the loss value and stops the training process if the loss didn't improve by at least 0.0001 for 10 straight epochs. We also globally set the learning rate to be 0.01 which empirically is known to perform well in a variety of learning tasks. For the single-layer network we setup a single layer with a single neuron which calculates a linear operation of the inputs. A neural network with a linear activation function is simply a linear regression model. For the 3-layer dense network we used 3 layers with a depth of 32 in each one and Rectified Linear Units (ReLU) as activation function in the neurons. ReLU takes the following form:

$$f(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases}$$

ReLU has a non-linear behaviour which allows the modelling of more complex associations between data, and is also more efficient in training. For the loss function we choose in all experiments to minimize the Mean Squared Error between the network output and the target values.

3. Training and Validation of Digital Twin Models

3.1 Building Thermal Model

In this section, we apply the general methodology introduced in Section 2.1 for the case of thermal modeling of a building. We will detail the used thermal building model, the type of experiments and the used configurations of physics informed neural network architectures. We compare the prediction accuracy of both architectures against a similar conventional neural network and assess whether the proposed architectures can be used for control applications.

3.1.1 Thermal Model of a Building

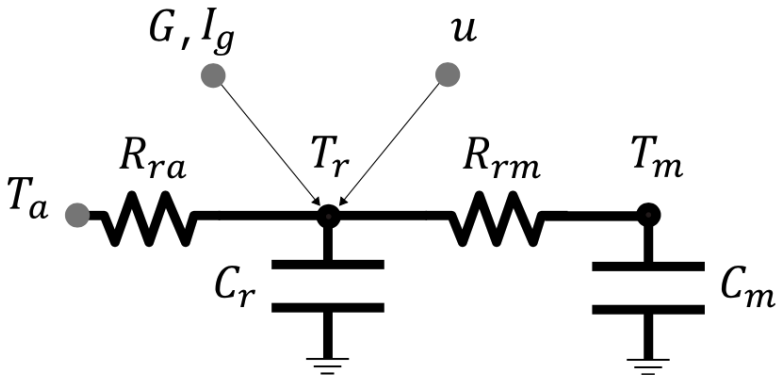


Figure 5. The resistance-capacitance network thermal model of a building.

A simplified scenario is considered with a single room (or zone) that is heated using a heat source. The inside room temperature and power consumed by the heating source are monitored over fixed time intervals (Δ_t) with a resolution of 30 minutes. The objective of the model is to predict the room temperature and the power consumed for subsequent time steps. To model this scenario, we adopt a grey box modeling approach using the 2R2C network model [35] , illustrated in Figure 5 and with the following state-space formulation:

$$\begin{bmatrix} \dot{T}_r \\ \dot{T}_m \end{bmatrix} = \begin{bmatrix} -\left(\frac{1}{C_r R_{ra}} + \frac{1}{C_r R_{rm}}\right) & \frac{1}{C_r R_{rm}} \\ \frac{1}{C_m R_{rm}} & -\frac{1}{C_m R_{rm}} \end{bmatrix} \cdot \begin{bmatrix} T_r \\ T_m \end{bmatrix} + \begin{bmatrix} b \\ 0 \end{bmatrix} \cdot u + \begin{bmatrix} \frac{\alpha}{C_r} & \frac{\beta}{C_r} & \frac{1}{C_r R_{ra}} \\ \frac{1-\alpha}{C_m} & \frac{1-\beta}{C_m} & 0 \end{bmatrix} \cdot \begin{bmatrix} G \\ I_g \\ T_a \end{bmatrix} \quad (9)$$

Here, T_r, T_m and T_a are the room temperature, temperature of building's thermal mass and outside temperature respectively, G, I_g represent solar irradiance and internal heat gains, and R_i, C_j correspond to heat transfer parameters of the building. The room temperature T_r is an observable state of the system that can be measured. Contrarily, T_m is a hidden state of the system which cannot be measured directly and, in most cases, is extremely difficult to estimate. This modeling approach hence leads to a partially observable model of the building. Additionally, a low-level back-up controller is assumed which ensures that the room temperature remains within a predefined set of limits based on the comfort of the user. The action of this back-up controller affects the actual power consumption (u_i^{phys}) which is modeled as:

$$u_i^{phys} = \begin{cases} 0 & : T_{r,i} > T_r^{max} \\ u_i & : T_r^{min} \leq T_{r,i} \leq T_r^{max} \\ u^{max} & : T_{r,i} < T_r^{min} \end{cases} \qquad (10)$$

This backup controller ensures the comfort of the user and its actions leads to a difference between the power demanded (u_i) and the actual power consumed (u_i^{phys}). To solve Eq. (9)–(10), an accurate estimate of hidden state (T_m) is required along with accurate measurements related to exogenous quantities like G and I_g. Since in practice precise estimates are difficult to obtain, we will eventually get only an approximate solution. Further, the building parameters like conductivity of different walls change over time due to deterioration and lead to model bias. Hence modeling a household directly using Eq. (9)–(10) is a difficult and expensive process and can lead to biased and sub-optimal control policies.

3.1.2 Physics Informed Neural Network Configurations

Symbol	Physical Meaning
$\boldsymbol{x_i^f}$	$\{(T_{r,i-k}, \dots, T_{r,i-1}), (u_{i-k-1}^{phys}, \dots, u_{i-2}^{phys})\}$
$\boldsymbol{x_i^{obs}}$	$(T_{r,i}, u_{i-1}^{phys})$
$\boldsymbol{w_i}$	$(t_i, T_{a,i})$
$\boldsymbol{z_i}$	$T_{m,i}$
$\boldsymbol{u_i}$	Controller Power

Table 2. State and Action Definitions for time step i

Both architectures shown in Figure 2 were used and prior physics information in the form of Eq. (9) was given to these networks. The state-space model defined using Eq. (9) leads to a partially observable system. To mitigate this, input in the form of a sequence of past k observable states and actions (x_i^f) along with observable state, actions, and other exogenous information in the form of time of day (t) and outside air temperature is used. With these inputs, the networks predict the room temperature, power consumption, and estimate the temperature of building thermal mass (T_m). The resulting state and action definitions are summarized in Table 2.

The parameter k, referred to as 'depth', controls the amount of information given to the neural network. It is important to note that the observed states for time step i consist of the room temperature at this time step ($T_{r,i}$) and the actual power that was consumed during the last time step (u_{i-1}^{phys}). Prior physics knowledge is provided to these architectures by directly plugging in Eq. (9) in the form of

$$\begin{bmatrix} \dot{T}_r \\ \dot{T}_m \end{bmatrix} = \begin{bmatrix} -a_{11} & a_{12} \\ a_{21} & -a_{22} \end{bmatrix} \cdot \begin{bmatrix} T_r \\ T_m \end{bmatrix} + \begin{bmatrix} b \\ 0 \end{bmatrix} \cdot u + \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{23} & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ T_a \end{bmatrix} \qquad (11)$$

The parameters a_i, b, c_j are building specific parameters and initialized based on the building EPC values and further tuned during the training phase. This ensures that in the absence of accurate values of these parameters, the model can be initialized with approximate values. Moreover, these

values can be tuned over time, thus taking into account any natural variations. With this information setting, the different components of the loss function defined in Eq. (6) can be formulated as:

$$L_{reg} = \frac{1}{N}\sum_{i=1}^{N}(T_{r,i} - \hat{T}_{r,i})^2 + \frac{1}{N}\sum_{i=1}^{N}(u_i^{phys} - \hat{u}_i^{phys})^2$$

$$L_{phys} = \frac{1}{N}\sum_{i=1}^{N}(T_{m,i}^{M} - \hat{T}_{m,i})^2 \qquad (12)$$

The hidden state ($T_{m,i}^{M}$) represents the physics module output and is computed by first estimating $\dot{T}_{r,i}$ and then using Eq. (11) to obtain $T_{m,i}^{M}$ as shown in Eq. (13).

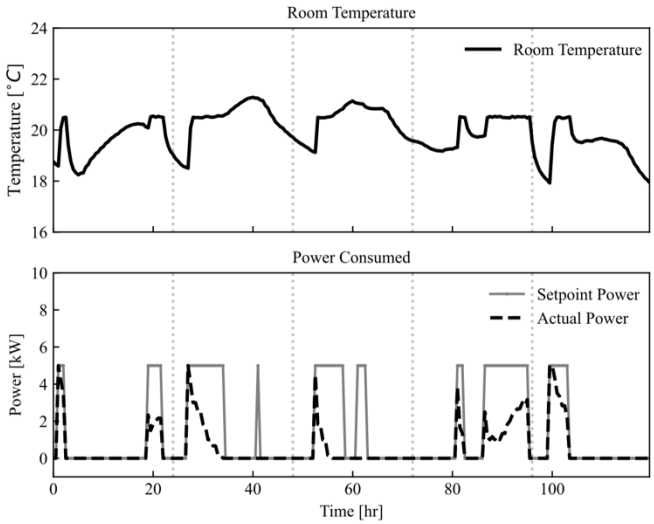
$$\dot{T}_{r,i} = \frac{T_{r,i+1} - T_{r,i}}{\Delta t}$$

$$T_{m,i}^{M} = \frac{1}{a_{12}}\left(\dot{T}_{r,i} + a_{11}\hat{T}_{r,i} - b\hat{u}_i^{phys} - c_{13}T_{a,i}\right) \qquad (13)$$

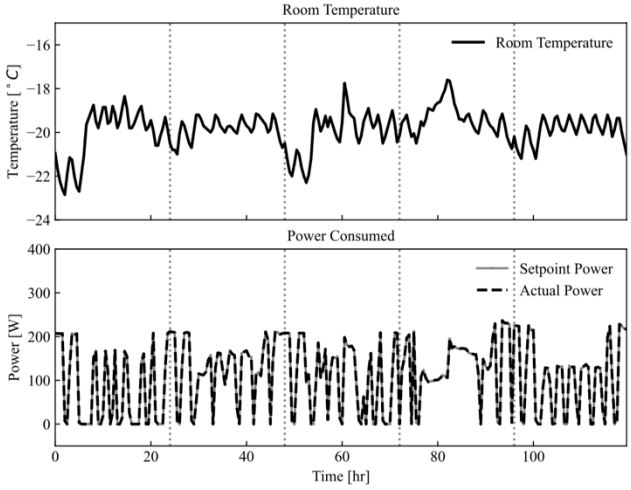
Here, $\hat{T}_{r,i}$ is obtained as an output by using input sample $i-1$ and $\hat{u}_i^{phys}$. The target value for the hidden state is explicitly dependent on the predictions of the room temperature and the power consumed. The loss functions defined in Eq. (12) guides the outputs of the networks towards physically relevant values.

3.1.3 Training Data

We considered two different scenarios for obtaining training data for these architectures: (1) a *simulated* single household environment, and (2) *real-world cold storage* data. The simulated scenario has been specifically designed to assess the capacity of the proposed physics informed neural network architectures to estimate the hidden state T_m of a building. After establishing this, later experiments focused on the real-world data scenario and assessed the performance of our proposed architectures for different configurations. Both scenarios involved observations related to room temperature, actual power consumption, control actions and outside air temperature. The frequency of these measurements was set to 1 measurement per 30 minutes, as this is enough for a heating/cooling system which has a rather slow reaction time. A training dataset equivalent to 120 days of such measurements was generated/collected. Similarly, a test dataset was generated equivalent to 5 days that were not a part of the training set. Each day corresponds to 48 input samples and the test sets for both scenarios used are shown in Figure 6.



(a) Simulated data for a household



(b) Real-world cold storage data

Figure 6. Test datasets

Simulated Data

In this scenario, Eq. (9)–(10) were solved, as discussed in [38]. A discretization step of 1 minute was assumed and a control action was taken every 30 minutes. For every minute, a first order approximation of Eq. (9) was solved to obtain the room temperature, hidden state, and actual power consumption. To further simplify the scenario, we ignored the effects of solar irradiance and internal heat gain. The simulation was initialized by setting $T_r = T_m = 17°C$. Further, control action u was chosen randomly and did not follow any active control logic. Figure 6 (top) shows a subset of data generated in this scenario.

Real-World Data

This scenario involved data obtained from a cold storage. This scenario is more complex than the simulated data generated as it involved actions taken by an active control strategy and also included solar irradiance, internal heat gains, etc. The influence of these exogenous factors was recorded only indirectly, via the room temperature measurements, due to absence of sensors to directly

measure them. Figure 6 (bottom) shows a subset of data corresponding to this scenario. In this cold storage case, no back-up controller was used and hence the actual power consumed is identical as the control setpoints.

3.1.4 Parameter Tuning for Physics Informed Neural Network Architectures

Besides different data scenarios, we also analyze the impact of different parameter configurations for both architectures. The input given to both architectures involves a sequence of past room temperatures and control actions. The length of this sequence, referred to as 'depth', determines the amount of past information available to the model and is an important parameter in the architecture. This information, to a certain level compensates for missing information like solar irradiance or internal heat gains, helping the model to better estimate the hidden state of the building (T_m). Further, the network sizes and hyperparameters (learning rate, type of optimizers) were tuned for a base case of setting $\lambda = 0$ for both architectures. This ensured that the network size and representative power was not constrained by the physics-based regularization, and we can observe supplementary gains in performance after tuning λ.

Both variants of physics informed neural networks were implemented using the Pytorch Lightning package [36] . Table 3 and Table 4 present the hyperparameters chosen for these architectures. 20 models were trained using the same set of hyperparameters and with different seeds between 1 to 20. Both architectures were trained using a batch size of 2048 and with 75 epochs.

Parameter	**Value**
Optimizer	Adam
Learning Rate	0.001
Activation Function	Tanh
Batch Size	2048
Hidden Layers	2
Neurons per layer	64

Table 3. Hyperparameters for PhysReg MLP Architecture

Parameter	**Value**
Optimizer	Adam
Learning Rate	0.001
Activation Function	Tanh
Batch Size	2048
Encoder Module (θ_L)	
Hidden Layers	2
Neurons per Layer	24
Dynamics Module (θ_d)	
Hidden Layers	1
Neurons per Layer	128

Table 4. Hyperparameters for PhysNet Architecture

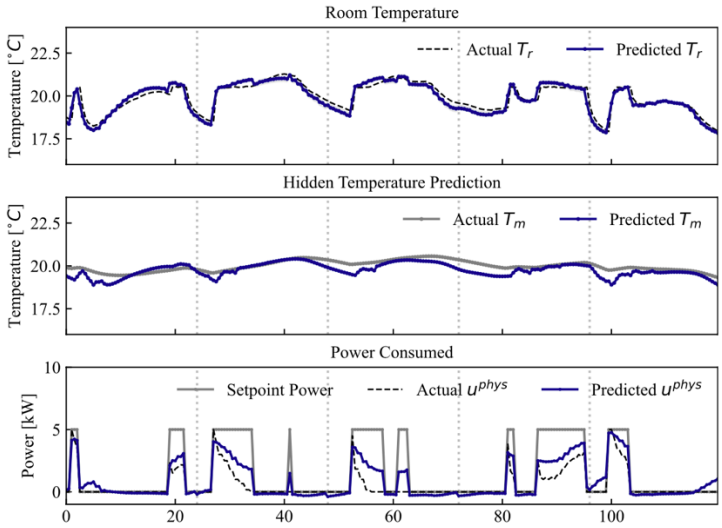
The hyperparameter values were obtained by minimizing the mean absolute error in predicted temperature as a performance metric. Because of the low training sample regime, training multiple models ensures that we obtain a distribution of performance values, thus mitigating the effects of possible outliers due to underfitting.

3.1.5 Results and Discussions

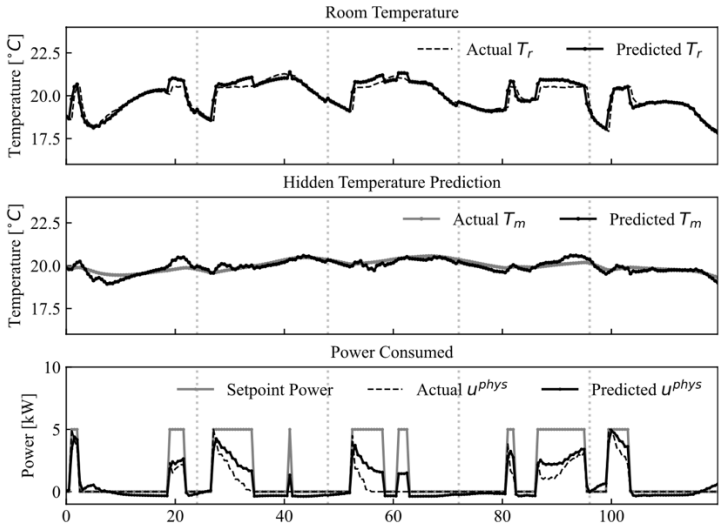
Three different experiments were performed to test our proposed PhysNet and PhysReg MLP architectures (Figure 2) and assess their performance as a control-oriented model.

Architecture Validation

The aim of our first experiment was to validate the performance of the proposed physics informed neural network architectures in determining the quality of the hidden state estimates. For this purpose, simulated data was used for training and validation. The validation dataset, shown in Figure 5 (top), contains 240 samples for which we computed the Mean Absolute Error (MAE) of predicted room temperature (T_r) and predicted hidden state (T_m). A fixed training size of 120 days (5,760 samples) was used along with a fixed depth value of 8. Figure 7 shows the predictions for both architectures.



(a) PhysNet Architecture



(b) PhysReg MLP Architecture

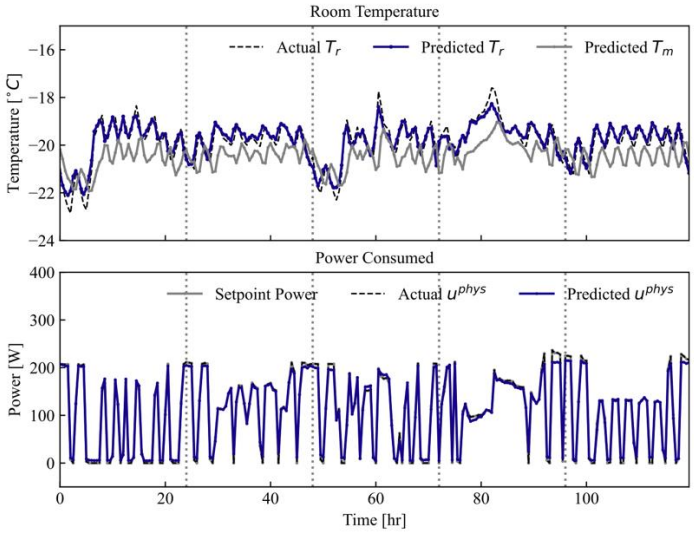
Figure 7. Prediction results for the two physics informed neural network architectures for the simulated data scenario

We note that for both cases, the room temperature and action predictions follow the actual values closely, indicating a good prediction performance. Additionally, the estimates of T_m track the actual values of hidden states, thus demonstrating the effectiveness of our proposed architectures for the given prediction task. Table 5 shows the MAE values for room temperature (T_r) and hidden state (T_m) predictions for this experiment.

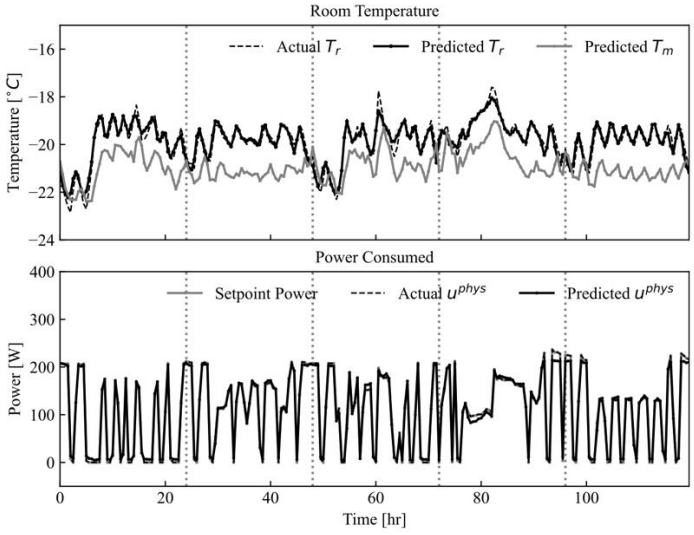
	MLP	**PhysReg MLP**	**PhysNet**
T_r	0.209 °C	0.197 °C	0.226 °C
T_m	1.413 °C	0.385 °C	0.436 °C

Table 5. Comparison of MAEs for room temperature (T_r) and hidden state (T_m) for the proposed architectures

Table 5 presents error values in ℃. A conventional MLP with the same hyperparameters as the PhysReg model was used to benchmark the performance of PhysNet and PhysReg MLP architectures. For all three networks, the mean errors are less than 0.25℃, indicating a good performance. Comparing the architectures, the PhysReg model performs the best with an absolute error of 0.197℃. However, there is a significant difference between mean errors for the hidden state, where conventional MLPs cannot estimate this state due to lack of target values, thus performing poorly in this metric. The physics informed neural network architectures perform 60 − 70% better than the conventional MLPs with an absolute error of less than 0.5℃. These results demonstrate that PhysNet and PhysReg MLP architectures can be used effectively to predict room temperature and hidden state and hence are more suitable for control oriented thermal modeling of a building.



(a) PhysNet Architecture



(b) PhysReg MLP Architecture

Figure 8. Prediction results for physics informed neural network architectures for the real-world data scenario.

Following these results, both physics informed neural network architectures were trained on real-world data obtained from a cold storage unit. Similar to the previous case, a training data size of

120 days was used and performance was validated on 5 test days, including a benchmark by a conventional neural network. Figure 8 shows the performance of PhysNet and PhysReg MLP architectures on the real-world data set.

We note that both architectures accurately predict the room temperature and power consumption values for the 5 test days along with an estimate of the hidden state of the system (T_m). The results shown in Figure 7, Figure 8 and Table 5 validate the performance of the proposed physics informed neural network architectures for the task of modeling the thermal behavior of a building.

Performance vs. Training Data Size

The second set of experiments analyzed the impact of training data size on the performance of physics informed neural network models. The motivation for using physics informed neural networks was to leverage prior knowledge to train models faster and more efficiently. To validate this, models were trained on real-world training data of varying size, sampled from the main training set. Each model was then tested using MAE as the performance metric on the test data set of 240 samples (5 days) shown in Figure 6 (bottom). Two different test configurations were used, depending on the prediction horizon. Our architectures enable one-step ahead prediction. To obtain predictions for longer horizons, a recursive strategy was used, where the model output was fed back to the model as input to generate multi-step forecasts. This strategy mimics a tree search algorithm used in model based RL techniques like [25] . The two test configurations used a prediction horizon of 3 hours (6 steps) and 12 hours (24 steps). The performance of physics informed neural networks was further compared to a conventional neural network and a persistence forecast model for both these configurations. Figure 9 shows the model performance for different training data sizes for the real-world data scenario.

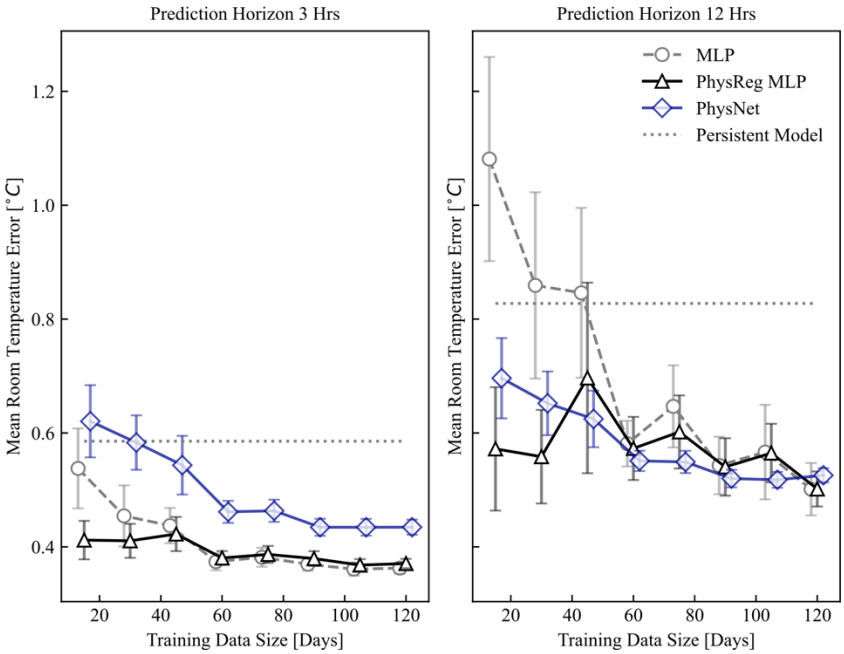


Figure 9. Mean room temperature prediction error for varying training data size. The plots represent mean error of 20 trained models and the error bars represent ± standard deviation.

We note that for a prediction horizon of 12 hours, both PhysNet and PhysReg MLP architectures perform better than the conventional MLP. For smaller training data sizes (15-45 days) the predictions for physics informed neural network architectures attain an MAE that is at least 15% lower than MLP. This difference decreases sharply with increasing training size, where for higher training sizes (> 90 days) the performance of all three architectures is similar. Contrary to this, for a shorter prediction horizon, the conventional MLP outperforms the PhysNet architecture, and performs similarly as the PhysReg MLP architecture. Additionally, in both configurations, all three architectures outperform a persistence forecasting model of similar prediction horizon for most training data sizes. This shows that introducing prior knowledge to the neural network architecture aids the network to learn more efficiently and requires less training data to reach equally good (or better) performance.

Performance vs. Prediction Horizon Size

From Figure 9, we note a difference in performance for different prediction horizons. While it is intuitively expected that increasing the prediction horizon will lead to compounding of errors, it is of interest to analyze how this performance degradation evolves for each of the two architectures. This experiment thus analyzes the performance of physics informed neural networks for varying prediction horizons. Because of their relevance for typical control time frames, prediction horizons of {0.5, 3, 6, 12, 18, 24} hours were selected, with each hour corresponding to 2 prediction steps. To include the impact of training data size, two training configurations of 30 days and 90 days were chosen. Like the previous experiment, real-world data was used with 5 test days as shown in Figure 6 (bottom). MAE of room temperature predictions was chosen as the performance metric and the performance was again benchmarked using a conventional MLP and a persistence forecast model with prediction horizon of 30 minutes (equaling 1 time step). Figure 10 presents the results obtained for this experiment.

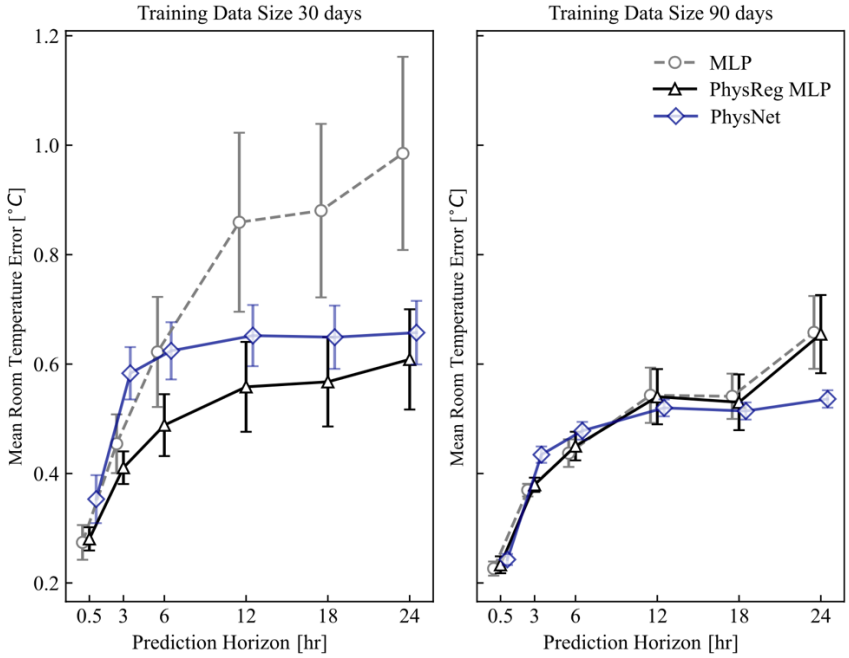


Figure 10. Mean room temperature prediction error for varying prediction horizons. The plots represent the mean error of 20 trained models and the error bars represent ± standard deviation.

We note that for a large training data (120 days), all three architectures perform similarly in terms of mean values. However, the error bars indicate that PhysNet MLP architectures produce results with a tighter distribution. This indicates a stable training performance. For low training sample configurations, the performance of conventional MLP deteriorates rapidly with increase in prediction horizon size, with an error of close to 1°C for the case of 24 hours. While there is a significant decrease in performance for physics informed neural networks, the error margins remain around 0.75°C with a standard deviation of ±0.3°C. This indicates that with less training data, the physics informed neural network models can use prior physics knowledge and lead to trained models that are stable and perform better than conventional MLPs. This is an important feature that can be leveraged in control applications for evaluating longer trajectories in tree searches.

These results demonstrate that introducing prior knowledge into a network leads to better predictions, makes the training process sample efficient and yields models that can be used for developing better control algorithms.

3.1.6. Next steps

Future work will involve two key directions: (i) developing Control Algorithms as part of WP5, and (ii) improving the PhysNet and PhysReg MLP architectures. In (i) we will use these architectures in model based RL algorithms like [25] [26] . The control agent will be capable of learning the model of the building and an optimum control policy simultaneously. Moreover, leveraging the learnt model, the agent can create a schedule for the next hours, making the decision-making process interpretable and allowing human supervisory control. For (ii), we aim to improve the architecture by introducing a direct multi-step forecasting capacity rather than the current one-step prediction setting. This will allow the architecture to produce one shot forecasts for a pre-defined prediction window. Other improvements include assessing the performance benefits of using recurrent neural networks in the model architecture and the role of clustering and transfer learning for scalable model deployment.

3.2 Domestic Hot Water Model

The proposed methodologies have been investigated and tested using Centrica's in-house data from hot water tanks. These data sets contain the tank temperature of the hot water tanks and the power used to heat the tank over an extended period of time.

The parameters of the hybrid digital twin model $\lambda = [\alpha, C, \lambda]$ are learnt using the loss function consisting of the mean squared error difference between the realizations of the tank temperature and the modelled tank temperature, a regularization term to limit the dimensionality of the multilayer perceptron and some cost functions to represent constraints keeping the values of the physical constants within realistic limits ($K^u(\alpha): \alpha < 1, K^l(\alpha): 0.98 < \alpha, K(C): C < 0.1$).

$$argmin_{\alpha,C,\lambda}\, loss = argmin_{\alpha,C,\lambda}\, [\left(T_k - f_\lambda(x)\right)^2 + \beta.|\lambda|^2 + K^u(\alpha) + K^l(\alpha) + K(C)]$$

Stochastic gradient descent is used to minimize the loss function, using the Adam solver [4].

Figure 11 shows the loss function evaluations and this for the training data and the test data during fitting of the hybrid model. The plot indicates the decreasing trend in the loss function and a

converging trend between the training and test loss. The oscillations in the learning curve are normal in a non-linear problem, such as the problem at hand.

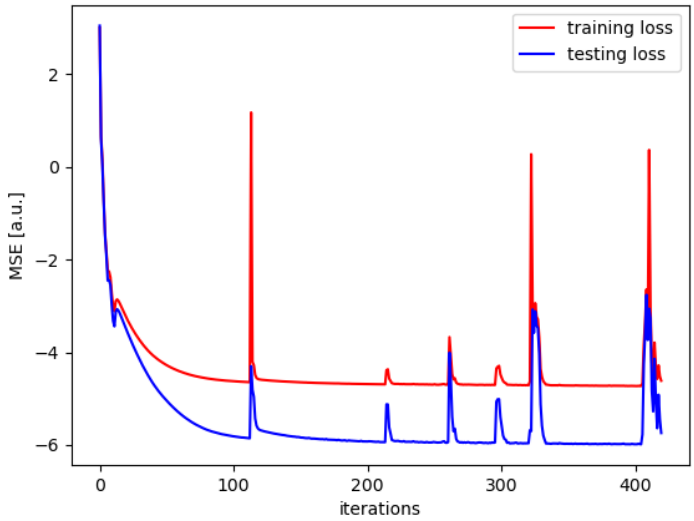


Figure 11. Evolution of mean squared error loss for training and testing data during fitting

Figure 12 shows a forecast of the temperature of the tank using the learnt hybrid model for a given initial temperature compared with the temperature of the tank according to the data set, which shows a reasonable performance.

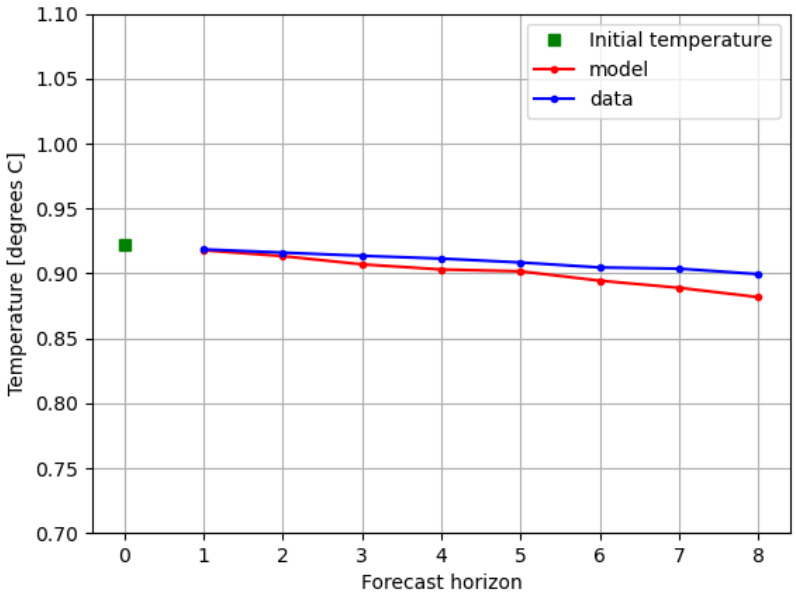


Figure 12. Forecast of the tank temperature for a given initial temperature of the tank

The model also performs as expected in a control context where the model is applied in a model predictive controller. The learnt model is used to forecast the temperature of the tank over a given forecast horizon if certain actions q_k^{hp} are taken. A hard constraint should avoid that the temperature drops below the threshold temperature of the tank:

$$\text{argmin}_{q_k^{hp}} \sum_{k\in[k,k+H]} C\left(q_k^{hp}\right)$$

$$T_{k+1} = \alpha.T_k + C.q_k^{hp} + f_\lambda\left(D_k,\ H_k, dT_k^{(1)}, dT_k^{(2)}, T_k\right)\forall k \in [k, k+H]$$
$$T_k > T^{thres}\ \forall k$$
$$T_0 = T^{init}$$

Figure 13 shows the control actions resulting from applying the linear model in a model predictive control algorithm. The graph shows that the heater is switched on (green dots) at moments when prices are low and switched off when prices are high.

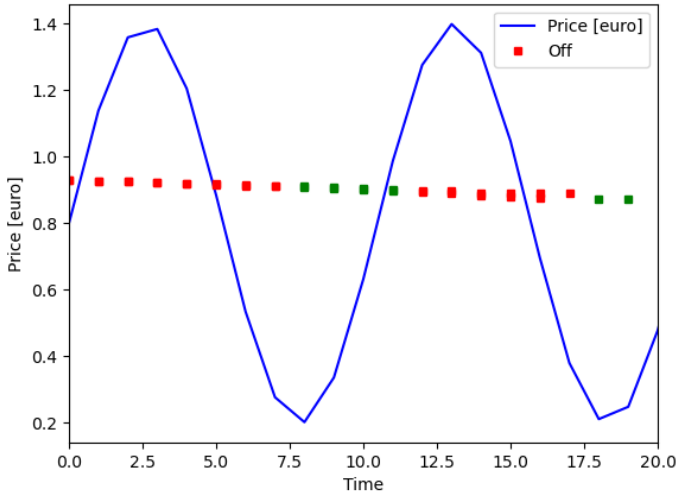


Figure 13. Control actions of the heater as a function of the price signal applying the hybrid model in a model predictive controller (green = on, red = off)

3.3 Thermal Loads & Assets Dynamic Simulator

3.3.1. Data acquisition and description

DomX has developed a smart monitoring and control system which involves the installation of a heating controller that acts as a bridge between the thermostat and the boiler. The domX controller collects various data regarding indoor and outdoor metrics, like temperature and humidity as well as metrics related to boiler activity, e.g., the water temperature inside the boiler, the temperature of the water (inlet) which is entering from the home radiators to the boiler and the boiler modulation level (% of max boiler modulation capacity). In addition, the domX controller runs a control loop algorithm to control the boiler activation patterns and adapt the actual temperature of the water circulated through the installed radiators based on indoor, setpoint and outdoor temperatures. This mechanism aims towards improving the boiler efficiency and the perceived user comfort whilst reducing the energy consumption and costs. This functionality also integrates user preference by exposing a heating balance setting, which defines how aggressively the adaptive algorithm will prioritize faster water heating to reach boiler setpoint temperature over the reduced consumption mode.

These values have been captured for several months. Since the raw data exist in various sampling frequencies, some simple pre-processing steps were performed to polish the data, i.e., (a) resampling all measurements to 1 second intervals, (b) removing erroneous values using rule-based outlier detection (e.g., known extreme temperature values) and (c) handling missing values by linear interpolation or filling. To summarize, all the metrics which were considered as variables of the heating system are:

- **Boiler modulation level**: The percentage of maximum boiler modulation capacity;
- **Boiler temperature**: The current temperature of the water inside the boiler;
- **Inlet temperature**: The current temperature of the water entering the boiler from the radiators;
- **Boiler setpoint**: The target water temperature the boiler is instructed to reach;
- **Indoor temperature**: The current room temperature inside the house;
- **Indoor setpoint**: The target room temperature the heating system must reach;
- **Outdoor temperature**: The current temperature outdoors;
- **Water**: Binary variable which indicates whether domestic hot water has been requested;
- **Heating balance**: User setting of the scale which controls the economy-comfort trade-off. Ranges from 0 (most efficient) to 10 (most aggressive);
- **Bypass**: Binary variable which indicates whether the adaptive algorithm is active or not. In case this is 0, the legacy mode is active which always sets the boiler setpoint to a fixed temperature typically within the range of 65-80°C.

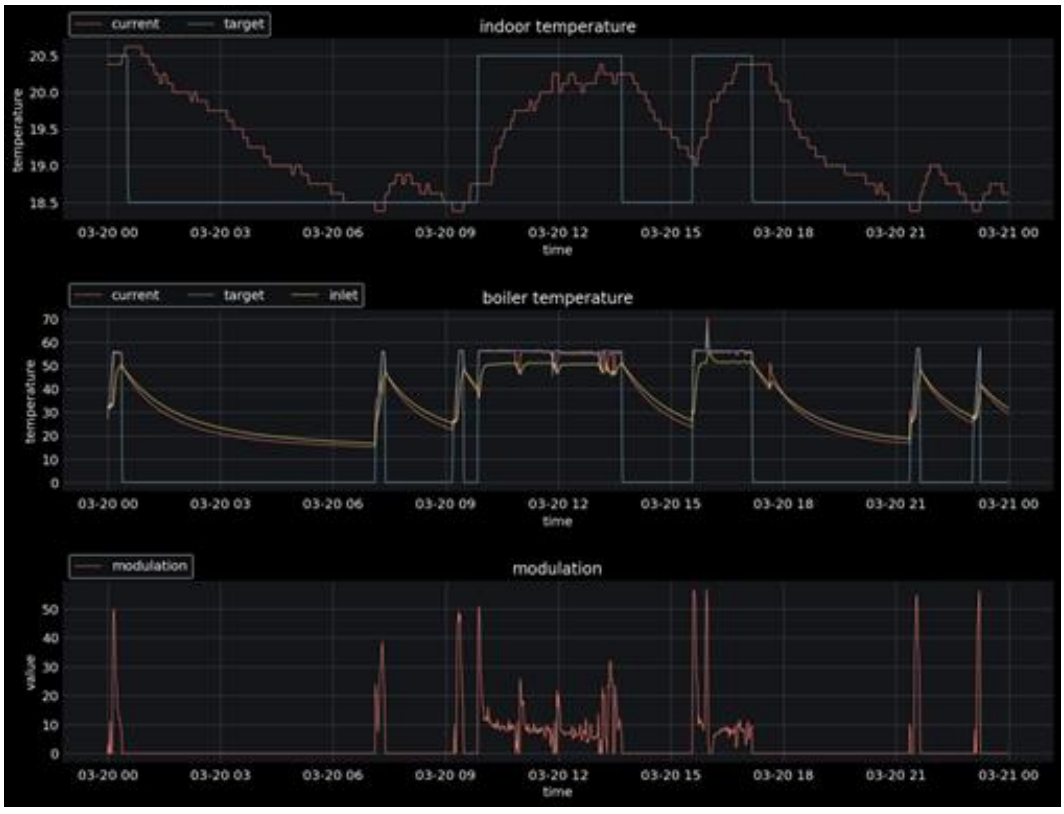


Figure 14. Example of a typical residential heating system.

3.3.2. Boiler modulation forecasting experiments

In the first set of experiments the boiler modulation level was selected as the variable to be forecasted while others were set as the predictor variables. This setting was chosen as the most

direct approach for simulating the heating system and in order to obtain insight about design choices in the methodology and how they impact the forecasting of consumed energy. A typical household's heating system is shown in Figure 14. What is influencing the modulation level directly are the temperatures related to the boiler and specifically the boiler temperature and the boiler setpoint: as long as the setpoint is higher that the current temperature the boiler needs to spend energy to heat up the water. Presumably, the outside weather is also a factor for the water temperature with a reasonable impact mainly when the system has been inactive for a long time and the water is cooling down. The set of indoor temperature and indoor setpoint values define whether heating is required through the radiators. One could argue that this requirement for heating can be expressed equivalently either by the indoor temperatures or the boiler temperatures, regardless for our first attempt we opted to get input from all variables. In future versions where we intend to produce forecasts for the indoor setpoint based on user historical data, we may assume the boiler temperatures as intrinsic variables which are hidden and attempt to associate the consumption with the indoor temperature and the indoor setpoint directly in order to emulate the behaviour of the boiler.

The data have been acquired for several domX clients and span from December of 2020 to April of 2021 for most of the households. We decided to split the data into training, validation and test sets accordingly:

- Training set: 12/2020 – 02/2021 (3 months);
- Validation set: 01/03/2021 – 15/03/2021 (1/2 month);
- Test set: 16/03/2021 – 16/04/2021 (1 month).

The training set is the data which we train the models on, the validation data are used during training to monitor the validation loss upon which the early stopping function decides when training stops and the test data are observations never seen by the models used for the final evaluation.

We decided to run the experiments using three different values for the time-series temporal sampling (timedelta, **dt**), specifically assuming that all time-series values are captured every 10 seconds, 1 minute or 5 minutes. Those are all possible settings that we may adopt in the future. At each time step we assume that every feature in all previous steps are known including the modulation and our intention is to build an accurate predictor of the boiler modulation one step into the future given current and previous values of all features. Our aim in this initial set of experiments is to compare model performance in terms of the root mean squared error (**RMSE**) averaged across all the predicted boiler modulation values of the test set, which we can then use at a later stage as the core module of a more complex dynamic simulator.

In order to get some indication about how far into the past is beneficial for each model to look we tested using both **single-step input** (input from the previous time step) and **multi-step** versions of the models which can accept input from multiple previous values of all features. Additionally, to check that the models are actually learning something from the training process, we evaluate them all against a baseliner which just copies into the future the previous boiler modulation value unchanged. The baseline model is essentially a model which predicts no change.

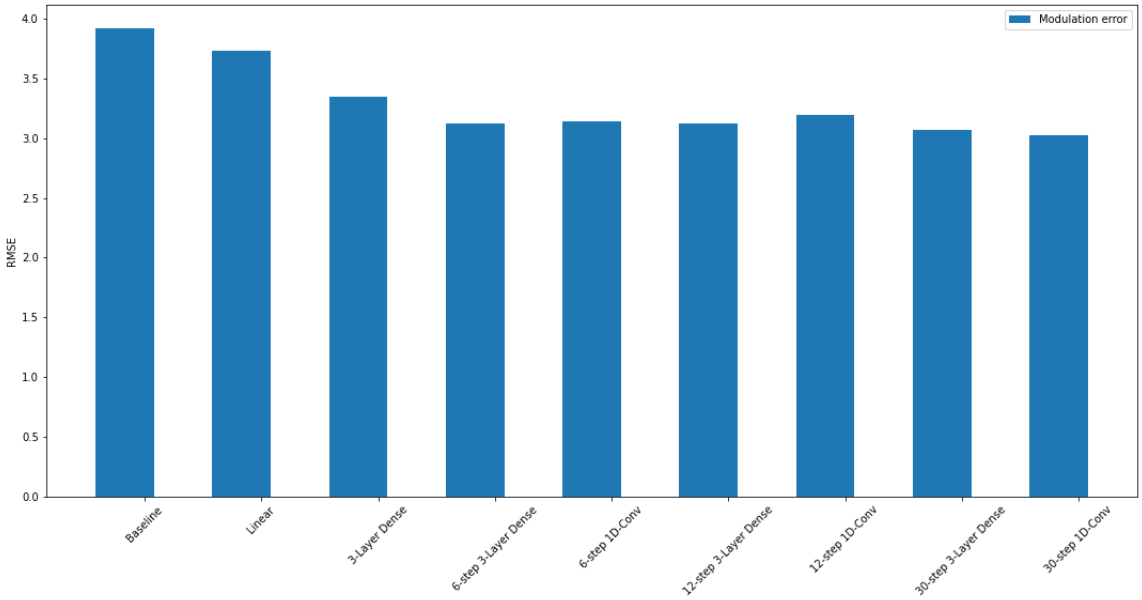


Figure 15. Test set RMSE, dt=10 seconds

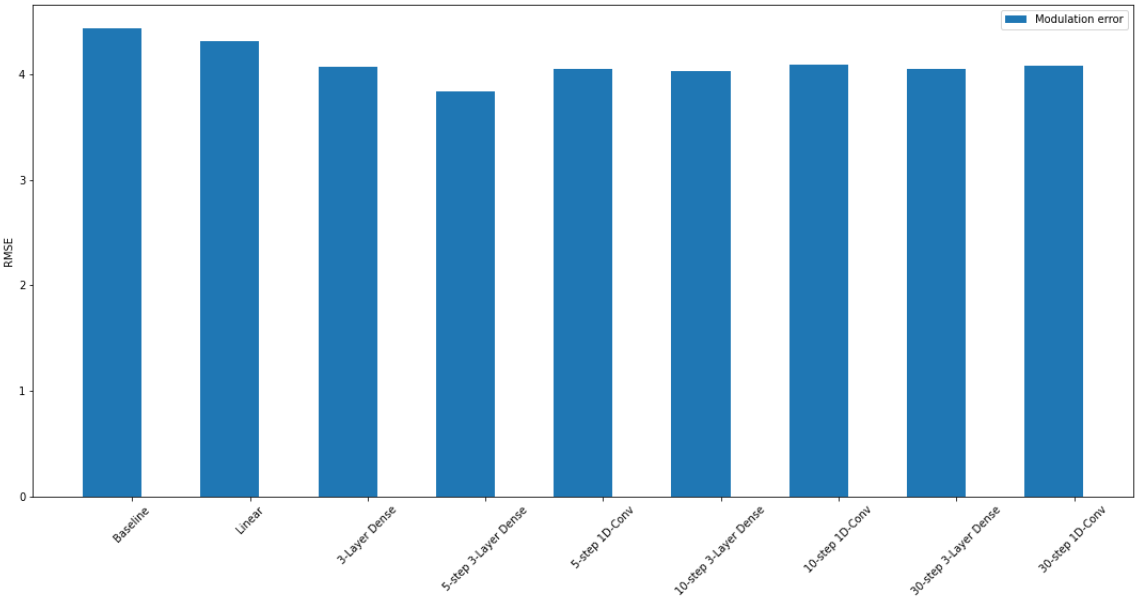


Figure 16. Test set RMSE, dt=60 seconds (1 minute)

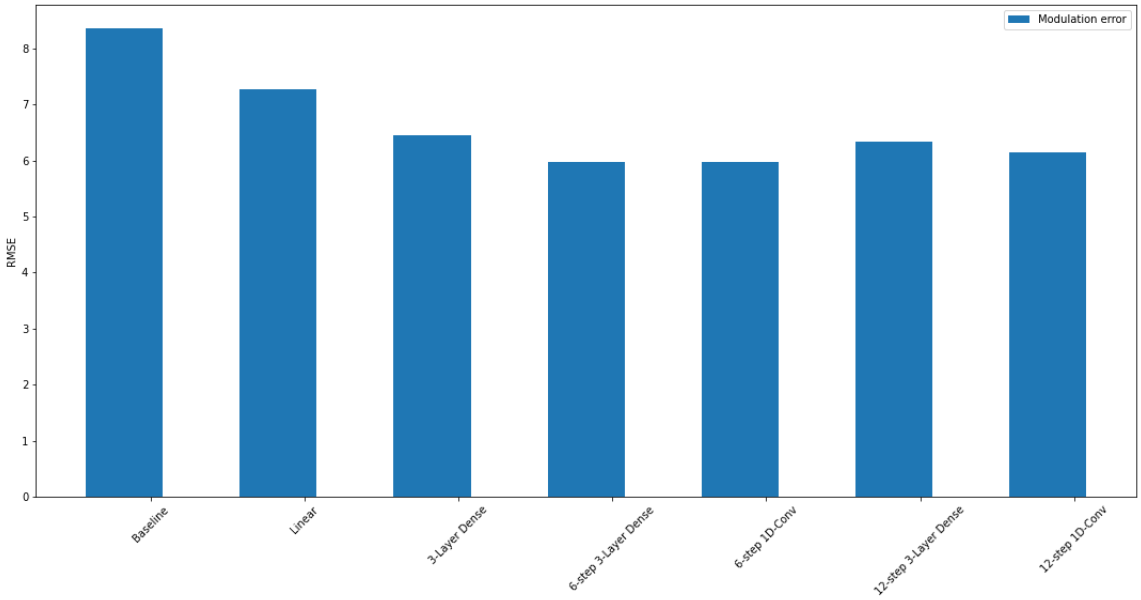


Figure 17. Test set RMSE, dt=300 seconds (5 minutes)

Figure 15, Figure 16 and Figure 17 show the test set RMSE for the three different timedelta settings. First, we can observe that every trained model performs better than the "no-change" baseline which is a sanity check for our methodology that the training process is working as expected. Starting from the linear model we can observe in all charts the drop in RMSE compared to the baseline, which is more evident in the 5-minute timedelta. This is expected, since the modulation is a series that varies rapidly with time and simply copying the last 5-minute value will result in much worse performance than copying the previous 10-second value. Another interesting observation can be made about the dense models which outperform the linear ones in every setting. This validates that the problem is more complex than the capacity of a linear model. Also, regarding multi-step input, it can be seen that it is beneficial to add about 5 or 6 steps into the past and any additional information from the past is either redundant or, interestingly, may even worsen performance. Finally, it can be concluded that there is no real benefit on performance when using a 1D convolutional network, but such a network could still be considered as an alternative option as it produces a similar performance more efficiently, since they tend to converge faster than dense networks.

4. Conclusions

This deliverable presented the first results on digital twin models for groups of flexible devices in energy communities. These digital twins will be used in WP5 to implement and validate flexibility valorisation techniques and algorithms to deliver flexibility services on community and system level.

In particular the following digital twins that are under development were described in detail:

- A data-driven modeling approach using physics informed neural networks to model the temporal evolution of a room temperature, the associated power consumption and temperature of the building thermal mass;
- A hybrid digital twin model for domestic hot water tanks which uses a similar approach. A physical model of the dynamics of the temperature of the tank is combined with a data-driven model;
- A pure data-driven approach using neural networks to forecast the behavior and performance of thermal assets (heating system, building and occupants) for diverse household heating scenarios.

In the second version of this deliverable (to be released in M30), an update will be provided on the presented digital twin models, with a presentation of further optimized models, trained with (larger amounts of) BRIGHT pilot data and data from TNO's HESI lab, and an evaluation of their impact on the development of flexibility services in WP5.

References

[1] T. Q. Pean, J. Salom, R. Costa-Castello, Review of control strategies for improving the energy flexibility provided by heat pump systems in buildings, Journal of Process Control 74 (2019) 35–49. doi:10.1016/j.jprocont.2018.03.006.

[2] J. Drgona, J. Arroyo, I. C. Figueroa, D. Blum, K. Arendt, D. Kim, E. P. Olle, J. Oravec, M. Wetter, D. L. Vrabie, et al., All you need to know about model predictive control for buildings, Annual Reviews in Control (2020).

[3] J. Cigler, D. Gyalistras, J. Siroky, V. Tiet, L. Ferkl, Beyond theory: the challenge of implementing model predictive control in buildings, in: Proceedings of 11th Rehva world congress, Clima, Vol. 250, 2013.

[4] S. Brandi, M. S. Piscitelli, M. Martellacci, A. Capozzoli, Deep reinforcement learning to optimise indoor temperature control and heating energy consumption in buildings, Energy and Buildings 224 (2020) 110225.

[5] A. Foucquier, S. Robert, F. Suard, L. Stephan, A. Jay, State of the art in building modelling and energy performances prediction: A review, Renewable and Sustainable Energy Reviews 23 (2013) 272–288.

[6] G. Gholamibozanjani, J. Tarragona, A. De Gracia, C. Fernandez, L. F. Cabeza, M. M. Farid, Model predictive control strategy applied to different types of building for space heating, Applied energy 231 (2018) 959–971.

[7] D. Perera, D. Winkler,N.-O. Skeie, Multi-floor building heating models in matlab and modelica environments, Applied Energy 171 (2016) 46–57.

[8] E. Zacekova, Z. Vana, J. Cigler, Towards the real-life implementation of mpc for an office building: Identification issues, Applied Energy 135 (2014) 53–62.

[9] F. Ferracuti, A. Fonti, L. Ciabattoni, S. Pizzuti, A. Arteconi, L. Helsen, G. Comodi, Data-driven models for short-term thermal behaviour prediction in real buildings, Applied Energy 204 (2017) 1375–1387.

[10] J. Tarragona, A. L. Pisello, C. Fernandez, A. de Gracia, L. F. Cabeza, Systematic review on model predictive control strategies applied to active thermal energy storage systems, Renewable and Sustainable Energy Reviews 149 (May) (2021). doi:10.1016/j.rser.2021.111385.

[11] J. Siroky, F. Oldewurtel, J. Cigler, S. Privara, Experimental analysis of model predictive control for an energy efficient building heating system, Applied energy 88 (9) (2011) 3079–3087.

[12] I.Hazyuk, C.Ghiaus, D.Penhouet, Optimal temperature control of intermittently heated buildings using Model Predictive Control: Part I - Building modeling, Building and Environment 51 (2012) 379–387. doi:10.1016/j.buildenv.2011.11.009.

[13] D. Sturzenegger, D. Gyalistras, M. Morari, R. S. Smith, Model predictive climate control of a swiss office building: implementation, results, and cost–benefit analysis, IEEE Transactions on Control Systems Technology 24 (1) (2015) 1–12.

[14] M. Bhardwaj, S. Choudhury, B. Boots, Blending mpc & value function approximation for efficient reinforcement learning, arXiv preprint arXiv:2012.05909 (2020).

[15] A. Kathirgamanathan, M. De Rosa, E. Mangina, D. P. Finn, Data-driven predictive control for unlocking building energy flexibility: A review, Renewable and Sustainable Energy Reviews 135, arXiv:2007.14866, doi:10.1016/j.rser.2020.110120.

[16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Pan- neershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of go with deep neural networks and tree search, Nature 529 (2016) 484–503.

[17] J. R. Vazquez-Canteli, Z. Nagy, Reinforcement learning for demand response: A review of algorithms and modeling techniques, Applied energy 235 (2019) 1072–1089.
[18] R. Sutton, A. Barto, An introduction to reinforcement learning, Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions (2011). doi:10.4018/978-1- 60960-165-2.ch004.
[19] L. Yang, Z. Nagy, P. Goffin, A. Schlueter, Reinforcement learning for optimal control of low exergy buildings, Applied Energy 156 (2015) 577–586.
[20] G. Ceusters, R. C. Rodriguez, A. B. Garcia, R. Franke, G. Deconinck, L. Helsen, A. Nowe, M. Messagie, L. R. Camargo, Model-predictive control and reinforcement learning in multi- energy system case studies, arXiv preprint arXiv:2104.09785 (2021).
[21] Z. Wang, T. Hong, Reinforcement learning for building controls: The opportunities and challenges, Applied Energy 269 (2020) 115036.
[22] M. Liu, S. Peeters, D. S. Callaway, B. J. Claessens, Trajectory tracking with an aggregation of domestic hot water heaters: Combining model-based and model-free control in a commercial deployment, IEEE Transactions on Smart Grid 10 (5) (2019) 5686–5695.
[23] S. Privara, J. Cigler, Z. Vana, F. Oldewurtel, C. Sagerschnig, E. Zacekova, Building modeling as a crucial part for building predictive control, Energy and Buildings 56 (2013) 8–22.
[24] E. Atam, L. Helsen, Control-oriented thermal modeling of multi zone buildings: Methods and issues: Intelligent control of a building system, IEEE Control Systems Magazine 36 (3) (2016) 86–111. doi:10.1109/MCS.2016.2535913.
[25] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, D. Silver, Mastering Atari, Go, chess and shogi by planning with a learned model, Nature 588 (7839) (2020) 604–609. arXiv:1911.08265, doi:10.1038/s41586-020- 03051-4.
[26] D. Hafner, T. Lillicrap, J. Ba, M. Norouzi, Dream to control: Learning behaviors by latent imagination, arXiv preprint arXiv:1912.01603 (2019).
[27] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707. doi:10.1016/j.jcp.2018.10.045.
[28] S. Greydanus, M. Dzamba, J. Yosinski, Hamiltonian neural networks, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 32, Curran Associates Inc., 2019.
[29] W. Degroote, S. Van Hoecke, G. Crevecoeur, Physics-Based Neural Network Models for Prediction of Cam-Follower Dynamics Beyond Nominal Operations, IEEE/ASME Transactions on Mechatronics PP (c) (2021) doi:10.1109/TMECH.2021.3101420.
[30] S. Cai, Z. Wang, S. Wang, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks for heat transfer problems, Journal of Heat Transfer 143 (6) (2021) 1–15. doi:10.1115/1.4050542.
[31] J. Drgona, A. R. Tuor, V. Chandan, D. L. Vrabie, Physics-constrained deep learning of multi-zone building thermal dynamics, Energy and Buildings 243 (2021) 110992.
[32] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (8) (2013) 1798–1828. arXiv:1206.5538, doi:10.1109/TPAMI.2013.50.
[33] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, A. Lerchner, Understanding disentangling in β-vae, arXiv preprint arXiv:1804.03599 (2018).
[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, arXiv preprint arXiv:1312.5602 (2013).

[35] E. Vrettos, E. C. Kara, J. MacDonald, G. Anders- son, D. S. Callaway, Experimental Demonstration of Frequency Regulation by Commercial Buildings-Part I: Modeling and Hierarchical Control Design, IEEE Transactions on Smart Grid 9 (4) (2018) 3213–3223. arXiv:1605.05835, doi:10.1109/TSG.2016.2628897.

[36] W. Falcon, The PyTorch Lightning team, PyTorch Lightning (3 2019). doi:10.5281/zenodo.3828935.