energies

MDPI

Article

Modular and Transferable Machine Learning for Heat Management and Reuse in Edge Data Centers

Rickard Brännvall [1,2,*], Jonas Gustafsson [1] and Fredrik Sandin [2]

[1] ICE Data Center, RISE Research Institutes of Sweden AB, 973 47 Luleå, Sweden
[2] EISLAB, Luleå University of Technology, 971 87 Luleå, Sweden
* Correspondence: rickard.brannvalll@ri.se

Abstract: This study investigates the use of transfer learning and modular design for adapting a pre-trained model to optimize energy efficiency and heat reuse in edge data centers while meeting local conditions, such as alternative heat management and hardware configurations. A Physics-Informed Data-Driven Recurrent Neural Network (PIDD RNN) is trained on a small scale-model experiment of a six-server data center to control cooling fans and maintain the exhaust chamber temperature within safe limits. The model features a hierarchical regularizing structure that reduces the degrees of freedom by connecting parameters for related modules in the system. With a RMSE value of 1.69, the PIDD RNN outperforms both a conventional RNN (RMSE: 3.18), and a State Space Model (RMSE: 2.66). We investigate how this design facilitates transfer learning when the model is fine-tuned over a few epochs to small dataset from a second set-up with a server located in a wind tunnel. The transferred model outperforms a model trained from scratch over hundreds of epochs.

Keywords: edge data center; heat management; heat reuse; modular machine learning; transferable machine learning; recurrent neural network; transfer learning; meta-learning

Citation: Brännvall, R.; Gustafsson, J.; Sandin, F. Modular and Transferable Machine Learning for Heat Management and Reuse in Edge Data Centers. *Energies* **2023**, *16*, 2255. https://doi.org/10.3390/en16052255

Academic Editors: Lioua Kolsi, Walid Hassen and Patrice Estellé

Received: 2 February 2023
Revised: 18 February 2023
Accepted: 23 February 2023
Published: 26 February 2023

1. Introduction

The number of data centers (DCs) around the globe is growing with the expanding digitization of society and the increasing demand for edge computing capacity. Thus, the amount of electric energy converted to heat by decentralized computing is also increasing, and the diversity of operating conditions is growing. Cooling units and fans remove heat from the chip, server, and rack levels [1], which further increases the total electricity consumed. Reusing excess heat has considerable potential for improving the energy efficiency of the DC industry, both in large-scale facilities and smaller edge data centers. Edge DCs are expected to be prevalent in cities and co-located in residential or commercial buildings where they can contribute to space heating demands, for example, in urban horticulture, as illustrated in Figure 1. Upcoming EU regulation will require larger DCs to report energy efficiency, including agreed-on heat reuse metrics although it is still unclear how this would apply to decentralized computing infrastructure, such as edge DC networks.

It is a challenging problem to optimize the overall electricity consumption of a system as complex as a modern edge DC while simultaneously meeting space heating constraints and avoiding damage to the equipment caused by overheating. Machine learning-based approaches that attempt to address this problem have received considerable attention recently (see, e.g., [2] or [3]). However, it remains to be demonstrated that such solutions are cost-efficient and scalable, given the diversity of operating conditions and components available on the market. Training modern machine learning models requires access to large amounts of data and may demand substantial resources in terms of human labor and electric energy. These methods will not scale well if the data collection and training procedure need to be carried out from scratch every time the model is applied in a new

context or configuration. This motivates the following investigation of modular architecture and transfer learning.

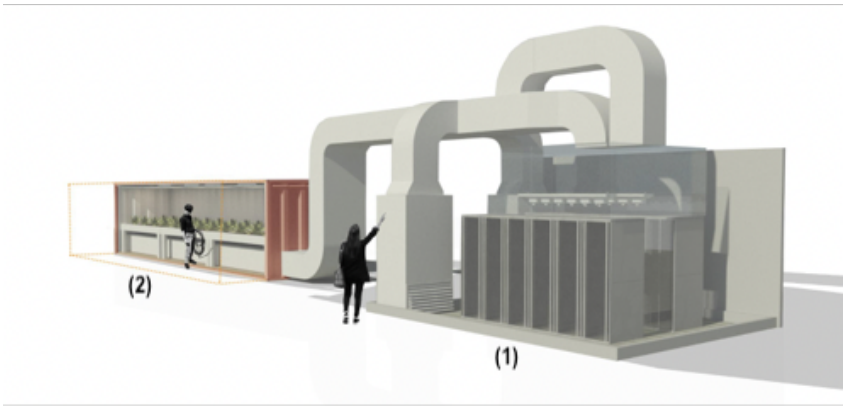

Figure 1. Example illustration of edge data center (1) heat reuse for a small urban greenhouse (2).

Contribution. Modular architectures for data center models describe physically identifiable system components (such as servers, fans, and aisles) and how they can be combined for a specific edge data center configuration. We allow the parameters for each identified component to be learned from data while simultaneously training a higher-level model that generalizes between components of the same type that share meta parameters. This regularizes the training process and facilitates transfer learning in a different configuration and context, as illustrated in Figure 2. Our experiment showed that after fine-tuning on a small dataset for only a few epochs, the transferred model can outperform a model trained from scratch over hundreds of epochs.

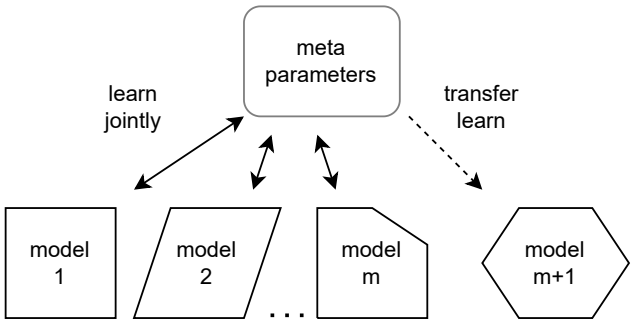

Figure 2. Hierarchical set-up transfer learning which can adapt to local circumstances (represented in the figure as odd shaped.

Outline. The remainder of this section reviews related work. Section 2 describes the experiment set-up, Section 3 details the model, while Section 4 presents the results from the machine learning exercise. Finally, Section 5 discusses the results and Section 6 concludes.

Related Work

At the level of an individual computer server, it is the control over the speed of the cooling fan and the computational workload that typically can be considered for thermal management purposes. An efficient server fan controller aims to hold server component temperatures at a given set point while also using a minimum of cooling power, as, for example, studied for closed loop settings in Wang et al. [4], Han and Joshi [5], Wang et al. [6], and specifically for edge computing servers cooling in Li et al. [7]. Controls based on physical models of the system face considerable challenges that spring from non-trivial long-range spatial and temporal dependencies. Computational Fluid Dynamics (CFD) can be used to model both static thermal properties and dynamic flows in detail, such as demonstrated in data center settings in Iyengar et al. [8], and more recently in

Wibron et al. [9]. However, the computational cost of such simulation is prohibiting for use in online closed-loop control. Pardey et al. [10] reviews compact models for the thermal inertia of servers and methods for their calibration. These predict exhaust temperatures in response to internal heating and time-varying ambient temperatures, such as those developed in parallel by VanGilder et al. [11] and Erden et al. [12] to use as simple, compact components for heat-producing servers and racks for larger CFD-based models.

Lucchese et al. [13] propose a control-oriented, non-linear, thermal model of individual servers and identify model parameters by CFD simulations of an idealized server set-up. Further simulations are then used to (in silico) both validate the model and test a Receding Horizon Control (RHC) strategy, aiming to keep IT component temperatures below thermal bounds while minimizing fan energy use. Building on this work, Eriksson et al. [14] first describe means for collecting detailed data from Open Compute Servers used by Facebook [15] and then write a control-oriented model for the thermal dynamics of the CPU and RAM as functions of computational load and mass flow produced by server fans.

VanGilder et al. [16] propose a compact model for data center applications with various chilled-water cooling systems. It includes a heat exchanger term in series with an "additional mass" term, which captures the thermal inertia of the cooling system. The first term is either a discretized numerical model of a simple counterflow heat exchanger or a quasi-steady-state model for applications when accuracy can be traded for model simplicity. This compact model is developed further in [17] that adds "additional mass" components for the room, plenum, walls, floor, ceiling, and water storage tank in order to better represent the complete thermal mass of a data center. The model is used to simulate a chiller failure and compared with a real-world incident in a 300 kW data center.

A similar modeling approach with thermally connected nodes in a network is followed by Lucchese and Johansson [18], but with improved models of the internal components of the server such that the thermal mass of each CPU and its associated aluminum fin heat exchanger are included. This allows finer details of the components' transient temperatures to be explained. Combined with a simple model of the thermal leakage for the hot chip, the thermal network is then used in an RHC strategy to control the cooling fans under an objective function that penalizes energy consumption (from both sources) while keeping the CPUs within temperature constraints. This thermal network-based model is again employed in [19] to examine heat recovery from Open Compute Windmill (V2) servers. Here the model-based controller is given the objective to maximize exhaust air temperature and encourage fan control signal smoothness, all while keeping CPU temperatures within limits for safe operation. Brannvall et al. [20] investigates a similar problem for a six-server cluster, and in a subsequent work [21] builds a model predictive control for an application where the excess heat is reused for drying fruit. Alternatively, type-2 fuzzy system have been proposed to handle abrupt changes of the reference signal or uncertainty in the parameters, such as, for example, in [22].

Geyer and Singaravel [23] proposes a component-based machine learning method for thermal management of a building design and argues that this extends the reusability and generalization of a model compared to monolithic designs. Gokhale et al. [24] uses physics-informed neural networks for control-oriented thermal modeling that is guided by building physics. For a data center, Berezovskaya et al. [25] describes a modular design with model components for CPU, server fans, chiller, and economizers. As is the case for the work presented in the present article, they demonstrate how to adjust parameters for individual servers to time-series measurements, although their model does not connect parameters for components of the same type together by regularization.

Transfer learning. When data are scarce, one can sometimes take a model trained on data from a related task and then fine-tune the model by training it on the target task. This procedure is an example of transfer learning (see, for example, a recent review Zhuang et al. [26]), which aims to address the difference in data distributions of the source and target tasks. On the other hand, meta-learning usually accumulates experience not only on one but often on multiple tasks to improve the efficiency of learning a new task. It

has been demonstrated to be efficient, for example, in image classification [27]. Bayesian learning involves other methods that facilitate transfer learning (see, e.g., book [28]), such as the use of hierarchical priors that provide probabilistic links between related system variables.

Control. When an empirical dynamical model has been obtained by system identification, as described in the works referenced above, it can be used for model predictive control (MPC). MPC is achieved by optimizing an objective function (relevant to the control problem) over a finite-time horizon. It obtains the control signal for the current time slot while taking likely system variable trajectories at future time slots into account. The optimization is repeated at subsequent time steps, which also moves the prediction window forward—that is why MPC is also sometimes called receding horizon control (RHC). There are many excellent review articles and books on MPC, see, for example, Rawlings et al. [29] and references therein.

Proportional-Integral-Derivative (PID) control is a model-free alternative to MPC-based approaches. It is popular in industrial applications because of its simplicity and relative robustness (see, for example, the review by Åström and Hägglund [30]), but unlike MPC, it does not have the capacity to anticipate future variable trajectories by a predictive model. Fuzzy proportional integral control was applied to constrain the gain and output values for cooling fan control by [31], with reported energy efficiency improvements for a Industry 4.0 data center. We do not however consider PID control further in this work.

2. Materials

Two experimental set-ups are considered in the following. The first is schematically illustrated in Figure 3a and comprises six servers, each equipped with two CPUs and onboard cooling fans. The work by the fans can be set programmatically via a custom-made control board that overrides the factory control logic. The fans create an airflow that transports heat away from the CPUs and other components in the servers into a shared chamber, where the air mixes before it passes into the outside room. The six OCP windmill v2 servers are housed by three server chassis arranged in a stack. The set-up was sealed to eliminate airflow outside of servers. Open source software (stress-ng: https://wiki.ubuntu.com/Kernel/Reference/stress-ng accessed on 25 February 2023) controlled the computational load on the servers according to pre-programmed schedules. The second experimental set-up is a small wind tunnel with room for one or two servers. Only the central section of the 5-meter-long wind tunnel is illustrated in Figure 3b.

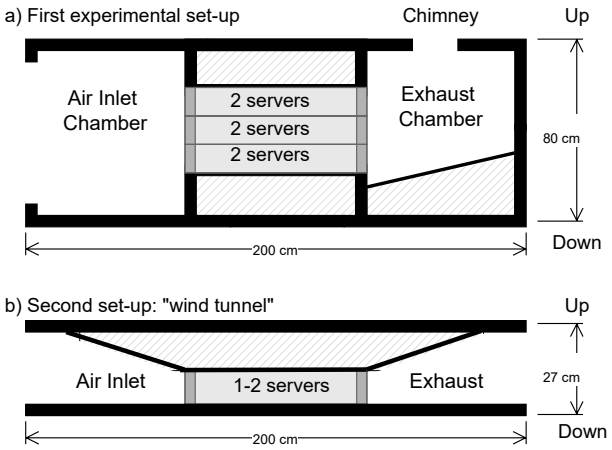

Figure 3. Two set-ups used in the scale-model experiments to investigate transfer learning between different hardware configurations: (**a**) wood box with a chamber that is heated by the exhaust air from a small server cluster, and (**b**) the central 2-meter section of a 5-meter-long wind tunnel.

The two set-ups are used in a scale-model experiment to investigate how transfer learning can be leveraged by first pre-training a model on the small-scale data center model

of Figure 3a to improve the performance on the wind tunnel setup of Figure 3b by adapting the pre-trained model to local hardware configurations and heat management conditions.

2.1. Fan Control Board

The fan control board is a custom-designed printed circuit board that enables external control of local server fans. It is specifically designed to fit OCP windmill v2 servers as a shim which is installed between the OCP backplane and the fan connectors. This enables the server fans to either be controlled by the server (same as the factory default) or manually controlled by the user. The board also allows the user to shut off the fans, which is not possible in the factory configuration.

To control the fans, the user sends a Modbus RTU command from the server via the USB port. The user can specify if the board should operate in the default factory mode, where the server has control of the fans, or if the user's control signal should take precedence instead. This is performed by toggling a hardware multiplexer to choose between sending the pulse width modulation (PWM) signal from the server or the fan controller. If the user has control, the fans can be controlled by sending a signal from 0–320, corresponding to 0–100% PWM. Along with control, many parameters are being measured on the fan control board, including fan speed, fan power, server PWM signal, user PWM signal, board temperature, and air temperature.

2.2. Data Collection

Time series for the modeled quantities, i.e., CPU temperatures (two per server) and chamber temperature measured at a central point, were collected at five-second sampling intervals together with the input variables server load and PWM percentage. Figure 4 shows example data from a step-response experiment that tested different combinations of fan control signal U and computation load P.

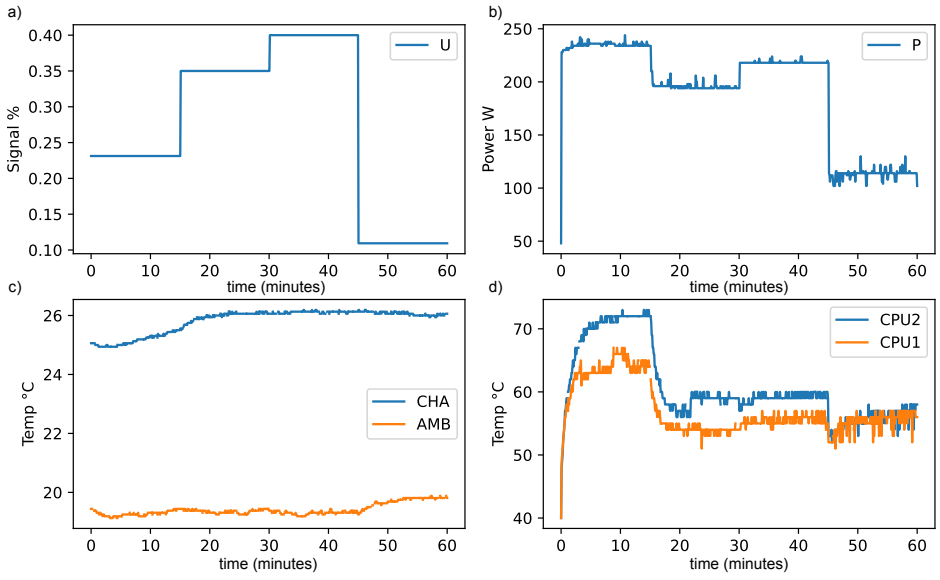

Figure 4. Example of one-hour experimental time series in panels; (**a**) top-left panel shows fan control signal (U), (**b**) top-right panel shows computational load (P), (**c**) bottom-left panel shows chamber (CHA) and ambient (AMB) temperatures, and (**d**) bottom-right panel shows CPU temperatures.

We make some observations from Figure 4 to gain an understanding of the system that we want to model and its transient behavior. CPU temperatures react quickly to changes in computational load and fan speed. They are observed to take values from below 40 °C when servers are idle to above 80 °C when they are fully utilized. The second CPU can be about 10 °C warmer than the first.

The chamber temperature changes much slower and keeps within a more limited range. In our experiments, it was observed to take values from 25 °C to above 35 °C depending on the ambient temperature, that is, the temperature of the room from which the server fans take in cooling air. The ambient temperature varies slowly between 16 and 20 °C over a day.

A visual inspection of Figure 4 tells us that CPU temperature typically appears to change over time-scales of a few minutes or less, while changes for chamber instead occur over tens of minutes. A rough calculation of response times to the step-wise changes supports this intuition, with the CPU taking, on average, 1.5 min before two-thirds of the trough-to-peak distance is traversed. The corresponding 2/3 response time for the chamber is about 20 min.

Sensors capture chamber and ambient temperatures at 0.1 °C resolution, while the sensor resolution is only 1 °C for the CPU measurements. All observations feed through a data collection chain developed at the lab as described by Gustafsson et al. [32].

2.3. Experiment Design

The load per server can be controlled between 50W at idle and 250W at full computational load. Different artificial loads were placed on the server while the fan speed was varied to generate data for training the machine learning algorithms. The experiment sequences were composed of multiple steps when the load and fan signal was held constant. All servers were exposed to the same sequences. The steps varied in length between 1 and 15 min, with the computational load and the fan signal strength determined by Latin hyper-cube sampling [33] in order to traverse the control variable space.

Experiments were also conducted to test the machine learning-based control algorithms. For these tests, pre-determined load sequences were again run on the servers while the server fans were regulated with the control algorithm.

3. Method

Figure 5 is a conceptual illustration of the server model described further in this section. First, we make a few short comments on notation and nomenclature. Note that although the framework for transfer learning illustrated in Figure 2 can accommodate many different source models, we only explore the single-source single-target set-up in this work (see Figure 6) as we only had access to two test-beds.

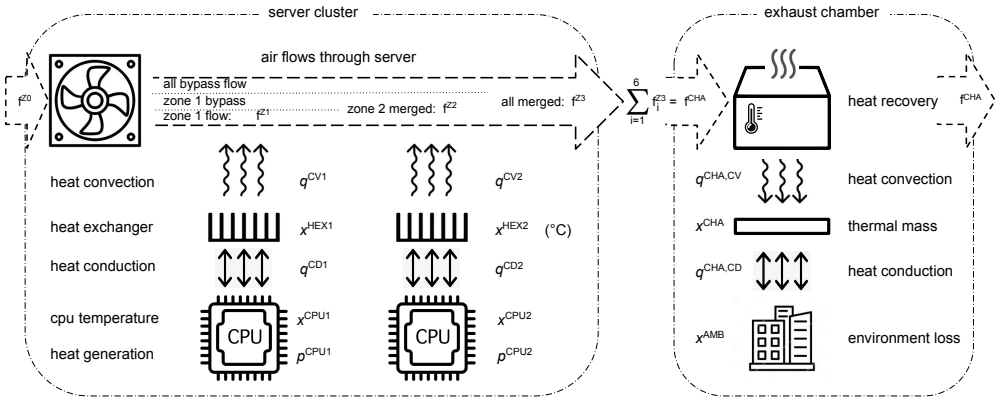

Figure 5. Conceptual illustration of the physics-informed thermal network model that is implemented as the custom cell of the physics-informed data-driven recurrent neural network (PIDD RNN).

3.1. Notation

The temperatures of the system are denoted with the letter x, computational load by p, the server fan control signal by u, and an unobserved unit-free mass flow f. Superscript

labels are used to indicate the position of the component for which the temperature is observed, such that for CPU1 we write

$$x_{i,t}^{\mathrm{CPU1}},$$

where subscripts i and t determine, respectively, for which server and at what time the observation was made.

In addition to physical position, the superscript is also used to distinguish related quantities, for example, the temperatures,

$$x_{i,t}^{\mathrm{Z1,IN}} \quad \text{and} \quad x_{i,t}^{\mathrm{Z1,OUT}}$$

which are the temperatures of the air that enter and exit zone 1 in the server (which houses CPU 1). Similarly, it is used to enumerate trainable model parameters, for example

$$\theta_i^{\mathrm{CPU1,D}} \quad \text{and} \quad \theta_i^{\mathrm{Z1,RCD}}$$

for the lump thermal capacitance of CPU1 and its conductive thermal resistance (in zone 1) with the heat exchanger. Note that the (lowercase) Greek letter θ (with different subscripts and superscripts) is used throughout the text to denote trainable parameters.

Capital letter Θ denotes the set of all trainable parameters for the six servers and the chamber module of the first experiment. We find these to the left in the conceptual illustration of parameter transfer of Figure 6, which belongs to the six-server experimental set-up—where [SRV] and [CHA] are used to represent any superscript associated with the server or the chamber, respectively. To the right in the same figure, we show the corresponding parameters set, $\tilde{\Theta}$, that are the target for fine-tuning on the wind tunnel experiment.

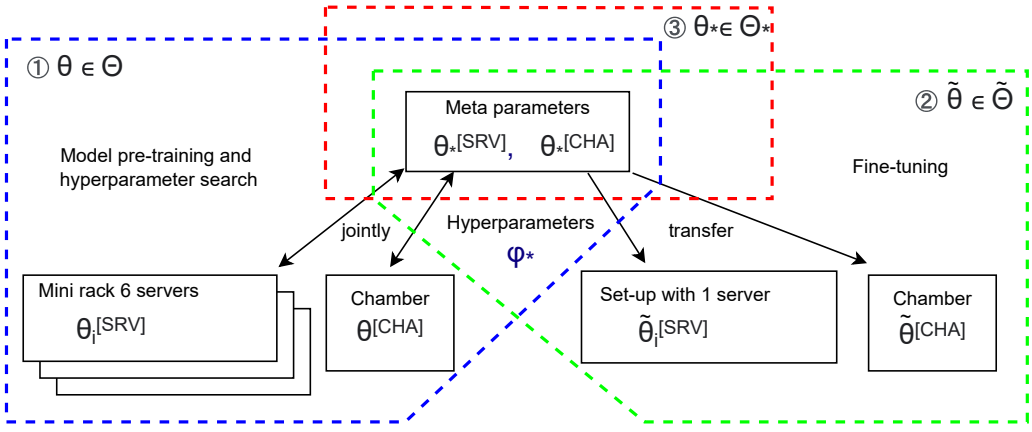

Figure 6. Conceptual illustration of the transfer learning procedure where in pre-training (1: blue area), the model for the six server set-up (with parameter set Θ) and the meta parameter set Θ_* are first trained together. Transfer learning (2: green area) is then obtained when the single server model (with parameters in set $\tilde{\Theta}$) is fine-tuned under regularization by the meta parameters. Note that the meta parameters (3: red area) are updated in pre-training (indicated by two-way arrows) but remain fixed during fine-tuning (one-way arrows). Hyper-parameters ϕ are also transferred.

One level up in the hierarchical model of Figure 6, we find the meta parameter set Θ_*, which contains generalized model parameters. These lack (server) index and are subscripted by a star symbol. The meta-model facilitates regularization and transfer learning.

3.2. Thermal Model

The server fan cooling system is complex with long-range spatial and temporal dependencies that are hard to model precisely. Instead, we use a physics-informed data-driven models [34].

Lucchese and Johansson [18] models the server as a network of interacting nodes that represent the different components of the system: extraction fans, self-heating components such as CPU and RAM, and passive components such as thermal sinks. The specifics of such a thermal network are determined by the relevant mass flows and heat exchanges that occur within the server; limited in the model to heat sources, convection, and conduction. The temperature dynamics of a generic node j are given in differential equation form by

$$d_j \dot{x} = p + q_j^{\mathrm{CV}} + q_j^{\mathrm{CD}}, \tag{1}$$

where d_j is the lumped thermal capacitance of the node, p a self-heating term, and q_j^{CV} and q_j^{CD} convective and conductive heat transfer, respectively.

The general thermal network model outlined above was developed for an individual server with two CPUs, each equipped with aluminum heat sink, and by assuming that:

1. Conductive heat transfer only occurs between a CPU and its heat sink;
2. Only heat sinks are affected by convection of the cooling air-flow f_j.

where f_j is the strength of the airflow passing node j. Furthermore, in that model, the heat conduction is approximated using Fourier's law and convection by Newton's law. That is, conduction is taken as being proportional to the temperature difference between two nodes, with the coefficient of proportionality being the (inverse) thermal resistance.

Additionally, we connect the outlet nodes of the server thermal networks with an outlet chamber node, assuming it has a thermal mass affected by convection and conduction (also modeled by Equation (1)). As illustrated in Figure 5 the airflow into the chamber is the sum of the air flows through each server. The distribution of airflow within each server is modeled such that net mass flow is preserved by using a categorical soft-max function of the fan speed with trainable parameters. The Appendix A has details and equations for the thermal model.

As we have six servers of the same type for the first experiment, it is reasonable to assume that they share some characteristics, although there may be individual variations between servers due to their placement in the rack, wear from years of service, or differences already present when delivered from factory. Therefore, we let each server have its own set of model parameters but tie all parameters of the same function in nodes together in a hierarchical construct (illustrated by the blue area in the left half of Figure 6). This is similar to setting a prior on all parameters in the server model, and we can, therefore, also refer to these generalized parameters as prior parameters. These structural relations reduce the number of degrees of freedom in the system and regularize the loss function of the training procedure.

3.3. Physics-Informed Data-Driven RNN

We refer to our model as physics-informed data-driven RNN, or in short, PIDD RNN, to borrow a term from Li et al. [35] for models that incorporate physics into machine learning model construction. It is implemented as a custom RNN cell, by sub-classing the corresponding object in the machine learning software package that we use.

The state and input variables of the PIDD RNN cell are:

$$\text{State}: \quad x_{i,t}^{\mathrm{HEX1}}, x_{i,t}^{\mathrm{CPU1}}, x_{i,t}^{\mathrm{HEX2}}, x_{i,t}^{\mathrm{CPU2}}, x_t^{\mathrm{CHA}}, \tag{2}$$

$$\text{Input}: \quad p_{i,t}, u_{i,t}, \tag{3}$$

for i being the index of the server, i.e., i is 1, 2, 3, 4, 5, or 6. The state variables are passed on to the next RNN cell as time is stepped forward. The initial state and the input

variables are taken from the observation time series. As we do not have sensor data for the heat exchangers, the steady-state temperatures (obtained from Equation (A9) from the Appendix A) are used to initialize these hidden variables.

For clarity, we note that it is the thermal network with trainable parameters of Figure 5, and described in Section 3.1, Section 3.2, and the Appendix A, that are implemented in place of the weighted transformations and activation functions of a conventional RNN cell.

3.4. Alternative Models

For comparison, two alternative models are also tested: one based on a conventional RNN and one on the linear state-space model (SSM), both taking the same state variables and control variables as input to predict the next state. Architectures with up to two hidden layers are tested for the conventional RNN, with the non-linearity taken to be the rectified linear unit (relu). The SSM model also follows a conventional formulation and allows full coupling between all state variables, although the observed variables do not directly depend on the external or control variables (i.e., the D matrix sometimes seen in the SSM formulation is not included).

$$X_t = AX_{t-1} + BU_{t-1} \tag{4}$$

$$Y_t = CX_t \tag{5}$$

A comparison is also made to the persistence model, i.e., the model which always predicts the next state to be the same as the current state. It is implemented as the RNN cell that only does an identity multiplication on the state and ignores any other inputs.

3.5. n-Step PEM

In a traditional time-series framework, we would proceed by letting the prediction error be described by a particular probability distribution and obtain estimates for model parameters by Maximum Likelihood. For analytic and computational convenience this would often mean that we make one-step ahead predictions and assume Gaussian errors.

In this study, however, we leverage on the flexibility of the open source software package for expressing non-standard cost functions and write down a n-step PEM objective.

For each component, observation time $0 < t < N - n$ and prediction horizon $0 < \tau \leq n$, we make a n-step prediction and calculate the prediction errors

$$\varepsilon^{\text{CPU1}}_{i,t,\tau} = x^{\text{CPU1}}_{i,t,\tau} - \tilde{x}^{\text{CPU1}}_{i,t+\tau} \tag{6}$$

where $\tilde{x}^{\text{CPU1}}_{i,t+\tau}$ is the actual observation corresponding to the prediction $x^{\text{CPU1}}_{i,t,\tau}$. The errors for CPU2, chamber (CHA), and server outlet (OUT) are similarly written.

The loss function sums these errors in an error norm of our choice, e.g., the squared L_2-norm for $\|X\|_\eta = X^2$

$$l^{\text{PEM}}_{t,\tau} = \omega^{\text{CHA}} \|\varepsilon^{\text{CHA}}_{t,\tau}\|_\eta + \omega^{\text{OUT}} \sum_{i=1}^{k} \|\varepsilon^{\text{OUT}}_{i,t,\tau}\|_\eta + \sum_{i=1}^{k} \|\varepsilon^{\text{CPU1}}_{i,t,\tau}\|_\eta + \sum_{i=1}^{k} \|\varepsilon^{\text{CPU2}}_{i,t,\tau}\|_\eta \tag{7}$$

for constants ω^{CHA} and ω^{OUT} that determine the relative weighting between deviations for CPUs, and chamber and outlet, respectively. For this study, we take $\omega^{\text{CHA}} = 12$ to give as high relative strength to the chamber as to the aggregated error of all CPUs. The server outlet temperatures are considered less important and receive weight $\omega^{\text{OUT}} = 0.1$ as they are not used in the target control problem.

The losses are aggregated for all observation times and prediction horizons

$$l = \frac{1}{Nn} \sum_{t=0}^{N-n} \sum_{\tau=1}^{n} \frac{l^{\text{PEM}}_{t,\tau}}{\omega^{\text{CHA}} + \omega^{\text{OUT}} + 2k} + \gamma h(\Theta) \tag{8}$$

adding a penalty through a regularizing function h that depends on the parameters of the model Θ.

3.6. Regularization

The hyper-parameter γ in (8) controls the strength of the regularization that directly targets the model parameters and is associated with a measure of dispersion in the distribution of the top-level prior parameters. For a square penalty function $h(\Theta)$, we would have that γ is (inversely) proportional to the variance of a Gaussian top-level prior. In this work, we are content with just pointing out the similarity to hierarchical Bayes [28] but do not take it further to a full formal model.

Next, the actual penalty functions $h(\Theta)$ used are then defined specifically for each model in the following paragraphs.

PIDD RNN: This model takes a hierarchical penalty for parameters associated with the server components

$$h(\Theta) = \sum_{\theta \in \Theta} \|\theta - \theta_*\|_\xi, \tag{9}$$

with θ_* being the member of the top-level hierarchical parameter set Θ that corresponds to the server or chamber level parameter θ. These are auxiliary parameters that tie all model parameters together in a hierarchical structure, such that, for example, the zone 1 convective resistance parameters, $\theta_i^{\mathrm{Z1,RCD}}$, for all servers are regularized by the auxiliary parameter $\theta_*^{\mathrm{Z1,RCD}}$ at the higher level in the hierarchy. The top-level hierarchical parameters are learned together with all other model parameters (as illustrated in Figure 6).

As the parameters of the model have different units, we use a customized norm for the penalty

$$\|x - y\|_\xi = 2\frac{|x-y|}{x+y} + \left(2\frac{|x-y|}{x+y}\right)^4, \tag{10}$$

which is similar to a percentage difference when the parameters compared are close (dominated by the first term) but boosts the penalty when the parameters are of different orders of magnitude (dominated by the second term).

Conventional RNN: We apply L_2-regularization on the kernel coefficient of the RNN cell.

State Space Model: We apply in L_2-regularization of the A and B matrices towards the persistence model.

Persistence model: Losses are not considered for the persistence model as it has no parameters and is not subject to training.

Noise layer: An additional means of regularization is provided through the addition of Gaussian noise to all RNN initialization values, that is, all initial CPU and chamber temperatures. This is because it is expected that long-term values for the state would depend more on the aggregated power inserted into the system and on the heat transferred away by the mass flow and less on the starting values.

The strength of this regularization is controlled by the standard deviation of the noise distribution, σ, which we treat as another hyper-parameter by which we can explore the effect of the perturbation and, by extension, on whether model performance on validation/test set is improved by encouraging the training procedure to discount starting values.

Other hyper-parameters: The length of the prediction window, n, the learning rate, r, and the number of training epochs, K, are other hyper-parameters associated with the training procedure. The choice of error norm for the PEM loss is also treated as a hyper-parameter. We test mean squared error (MSE), mean absolute error (MAE), and the Huber loss (Hub), which can be understood as a cross-over between MSE and MAE loss [36].

We denote the set of hyper-parameters for a model by the Greek letter ϕ and write ϕ_* for the model with the best performance on the validation set.

4. Results

4.1. Model Training

Manual derivation of the gradients over a n-step prediction would be a tedious task, but as the software package already supports the necessary functions and norms also for automatic differentiation we can immediately expose the model to the data through a built-in gradient-based optimizer. More specifically we use the ADAM optimizer by Kingma and Ba [37].

Observation data from a total of 54 h of experiments were available for analysis. From this 4 h were set aside as a validation set for hyper-parameter tuning, and an 8-hour section was set aside for out-of-sample testing. Models were trained for up to 1000 epochs on the training set with different combinations of hyper-parameters taken from a predefined grid of values.

Cross-model comparison of the score on the PEM loss function (8) is not meaningful, as the loss function is different for each model variation. Instead, we use RMSE, which is related to the loss and furthermore can be calculated separately for each component of the system.

For CPU, the RMSE is calculated for the first 1.5 min (i.e., 18 timesteps), while the chamber RMSE uses the last 5 min of the 15-minute prediction window. These windows were selected based on the relevant time-scale identified from inspecting the data as discussed in conjunction with Figure 4.

Hyper-parameter selection. A grid search over hyper-parameters $\phi = \{\gamma, \sigma, n, r, K\}$ was performed to select the hyper-parameters ϕ_* for which the candidate model shows the best weighted average RMSE on the validation set, where the chamber was given weight 6 compared to CPUs (as this gives equal weight to the chamber and each of CPU1 and CPU2 after summing over all servers). Note that deviations for predicted outlet temperatures were ignored for the RMSE.

Performance metrics. A comparison of the models' RMSE calculated on the test set is displayed in Table 1. It is evident that the PIDD RNN model outperforms both the conventional RNN and the SSM on both CPU and chamber total RMSE. All three models show an improvement over persistence on CPU scores, although both non-trivial alternative models fail to do better at describing the chamber dynamics. We note that the PIDD RNN model only shows a modest improvement over persistence on chamber RMSE.

Table 1. RMSE for prediction on the test set (°C) for PIDD RNN model compared with three alternative models.

Model	CPU1	CPU2	CHA	TOT
PIDD RNN	2.41	2.32	0.34	1.69
Persistence	6.93	7.54	0.38	4.95
Standard RNN	3.6	4.11	1.72	3.18
Standard SSM	3.04	2.98	1.97	2.66

Residual analysis. The low resolution (temperatures only in whole °C) complicates residual analysis as it induces a stratification pattern for the difference between the one-step ahead predicted temperature and its corresponding measured value. Tests for whiteness (normality) are, therefore, of limited use. We note that the auto-correlation of residuals for both chamber temperatures and CPU temperatures in the PIDD RNN model quickly diminishes to become insignificant (95% confidence intervals by Bartlett's standard formula) within 4–6 time steps (30 s). This is however not the case for the conventional RNN and SSM, which exhibits significant auto-correlation out to more than 60 timesteps (5 min).

Uncertainty envelopes. The uncertainty of the estimates grows with the length of the prediction horizon. This is visualized for CPU temperature in Figure 7 that plots the standard deviation of the prediction error against the prediction horizon. While there is very little bias in the estimate, the standard deviation for the PIDD RNN model predictions

grows gradually for the first couple of minutes and approaches a stable level. In contrast, the stdev envelope for the Persistence model is much wider, indicating that our model captures some important dynamics of the server system. The corresponding plot for the chamber does not show such a strong dominance of the PIDD RNN model (see Figure 8).

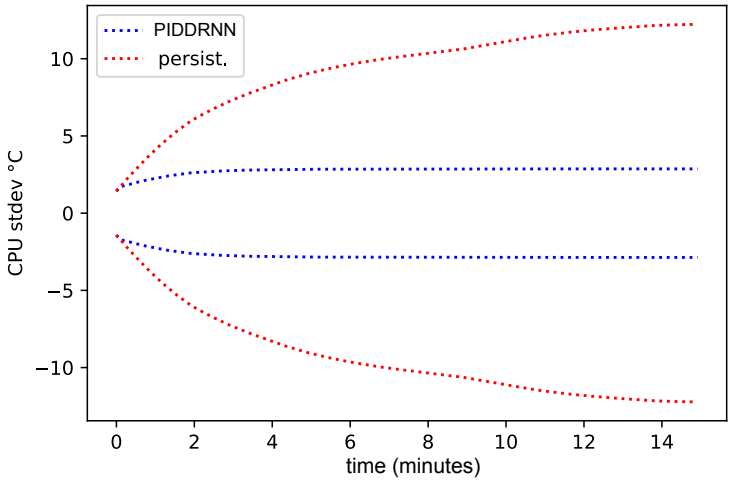

Figure 7. The standard deviation of the CPU test-set prediction error increases with the length of the prediction horizon. For the PIDD RNN model, its envelope quickly reaches a stable width, visibly narrower than that for the persistence model.

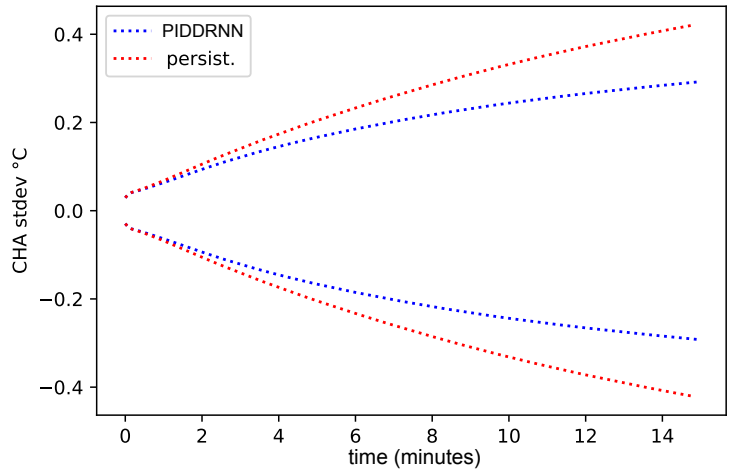

Figure 8. The width of the chamber test set stdev envelopes increases with the prediction horizon without reaching a stable level. The PIDD RNN model performs somewhat better than the persistence reference model.

Long term predictions. Figure 9 and 10 shows predictions over 2000 consecutive time-steps initialized only with starting values for CPU and chamber temperatures. Visual inspection of the predicted versus the true time series does indicate that the model is able to robustly describe the important thermal dynamics of the system, especially for the CPU where the large swings in component temperature (30–80 °C), caused by changing computational load and fan speed, appear to be accurately predicted by the model. Changes in the chamber temperature are slower and more limited in range (28–31 °C) mean it is difficult to make conclusions.

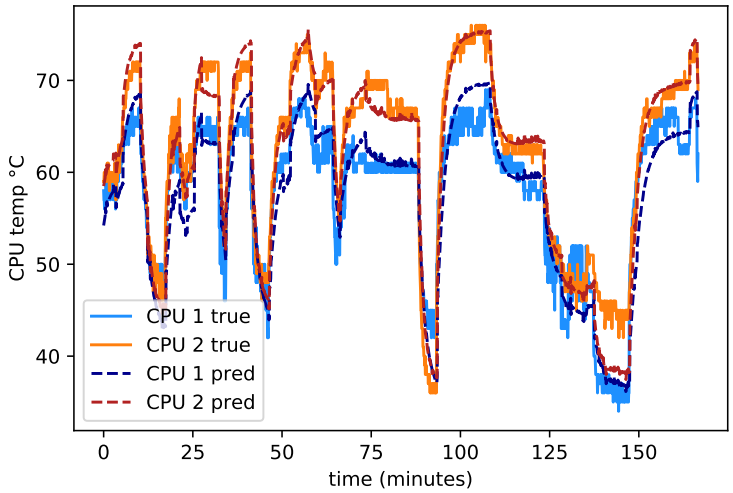

Figure 9. Long-term predictions for CPU temperatures on the test set. The PIDD RNN model was initialized at t=0, and then received only computational load and fan signal experiment sequences.

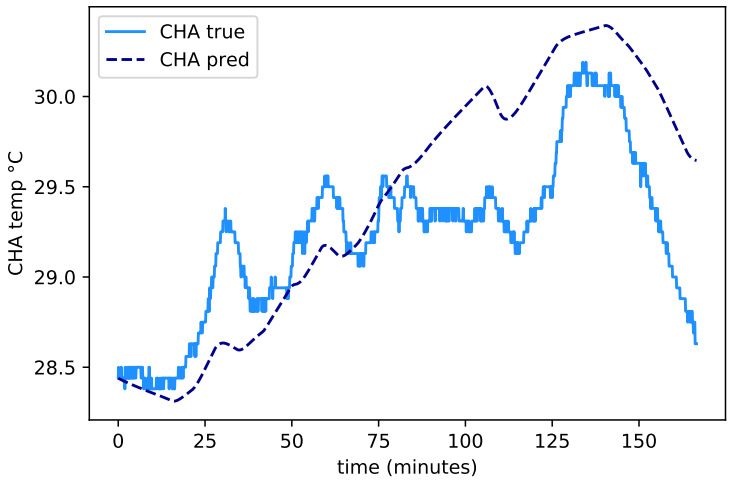

Figure 10. Corresponding test set long-term predictions for chamber temperature.

4.2. Fan Control

In this section, we explore different strategies for managing the system's temperature, assuming that the computation load is an external (stochastic) process outside of our control. It is then the work of each server fan (PWM) that must be dynamically set such that the CPUs do not overheat.

The starting point for each control strategy is the formulation of a loss function that penalizes system temperature deviations from a predefined set point. Secondary objectives are related to the cost of electricity used for cooling and smoothness of the control signal, as the pulsing of the fan (turning it on and off) tends to consume more power and cause increased wear and shorter life span of components.

Let U_t be the control signal for all servers over the next n time steps. This is a two-dimensional array taking prediction horizon τ on one axis and server index i on the other. The PIDD RNN model can then be used to make a n-step ahead prediction of the future temperatures conditional on this assumed fan control signal, the observed system temperatures, and what we know about the computational load. The flexibility of the model would allow for many alternative control strategies, for example, targeting a preferred CPU temperature, chamber temperature, or delta over the ambient temperature. To make a concrete example, we will in this section consider a scenario where we (1) cool the server CPUs such that for each server there is only a need for cooling when the hotter of its CPUs has a temperature above the setpoint $setpointx$, and (2) target a fixed chamber temperature.

We denote the deviations from setpoint as follows (using a similar notation as for the PEM loss)

$$\epsilon_{i,t,\tau}^{\mathrm{CPU1}} = \mathrm{relu}\left(x_{i,t,\tau}^{\mathrm{CPU1}} - \hat{x}^{\mathrm{CPU1}}\right) \tag{11}$$

and similarly for CPU 2. Note that this uses the asymmetric relu function such that no deviation is registered if the CPU temperature is below the setpoint.

For the chamber, we have

$$\epsilon_{t,\tau}^{\mathrm{CHA}} = x_{t,\tau}^{\mathrm{CHA}} - \hat{x}^{\mathrm{CHA}} \tag{12}$$

and sum over deviations according to

$$l_{t,\tau}^{\mathrm{DEV}} = \hat{w}^{\mathrm{CHA}} \|\epsilon_{i,t,\tau}^{\mathrm{CHA}}\|_\eta + \sum_{i=1}^{k} \|\epsilon_{i,t,\tau}^{\mathrm{CPU1}}\|_\eta + \sum_{i=1}^{k} \|\epsilon_{i,t,\tau}^{\mathrm{CPU2}}\|_\eta \tag{13}$$

and finally, over the entire prediction horizon

$$l_t^{\mathrm{MPC}} = \frac{1}{n} \sum_{\tau=1}^{n} \frac{l_{t,\tau}^{\mathrm{DEV}}}{\hat{w}^{\mathrm{CHA}} + 2k} + \Lambda(U_t), \tag{14}$$

with Λ_t a combination of different penalty terms that depend on the different signals

$$\Lambda_t = \frac{\lambda^{\mathrm{SIG}}}{nk} \sum_{\tau=1}^{n} \sum_{i=1}^{k} \left(u_{i,t,\tau}\right)^2 + \frac{\lambda^{\mathrm{SIM}}}{k} \sum_{i=1}^{k} \left(u_{i,t,1} - u_{i,t,0}\right)^2 + \frac{\lambda^{\mathrm{SRV}}}{nk} \sum_{\tau=1}^{n} \sum_{i=1}^{k} \left(u_{i,t,\tau} - \bar{u}_{t,\tau}\right)^2. \tag{15}$$

Here the first term gives the penalty for the strength of the signal, the second for abrupt changes compared to the prevailing value of the signal $u_{i,t,0}$, and the third term for variance away from the server average $\bar{u}_{t,\tau}$.

A fourth term that promotes smoothness of the signal could have been added, but instead, we incorporate that naturally by constraining the signal functional form, which specifically is chosen to be piece-wise linear with fixed pivots at $\tau_0 = 0$, $\tau_1 = 6$, and $\tau_2 = 30$ time-steps.

The optimization to find U_t is performed by gradient descent as both the n-step predictive model and the cost function is encoded as a differentiable computational graph in TensorFlow, where model parameters Θ are now held constant. With U appropriately initialized, a few tens of iterations are sufficient to approximate the control sequence and can fit within the time step of a realistic online application. The action at the present time step is then the $\tau = 0$ entry in the n-step control signal sequence. This procedure is repeated for each time step t.

The penalty hyper-parameters λ^{SIG}, λ^{SIM}, and λ^{SRV}, should be chosen such that one finds an acceptable trade-off between set-point deviance and signal regularization.

Two different MPC formulations were tested on the experimental set-up, each given control over the internal server fans as a random, artificial load sequence was run on the servers.

CPU-focused controller. We set $\hat{w}^{\mathrm{CHA}} = 0$ to zero out the chamber weight and obtain a controller with the objective only to keep CPU temperatures below the limit given by the CPU temperature set-point at 75 °C while having penalties $\lambda^{\mathrm{SIG}} = 10$ and $\lambda^{\mathrm{SIM}} = 10$ for the signal strength and shape, respectively.

Figure 11 displays results from a one-hour experiment with server-focused control. The upper panel shows CPU temperatures; the second CPU starts out at around 80 °C but is brought down below the 75 °C set-point within minutes, where it is kept for the remainder of the experiment. The lower panel shows the fan control signal, which is relatively stable but shows spikes when the computational load on the server changes.

Average temperatures are 64.9 °C and 72.0 °C for CPU1 and CPU2, respectively. The average control signal is 22.5%.

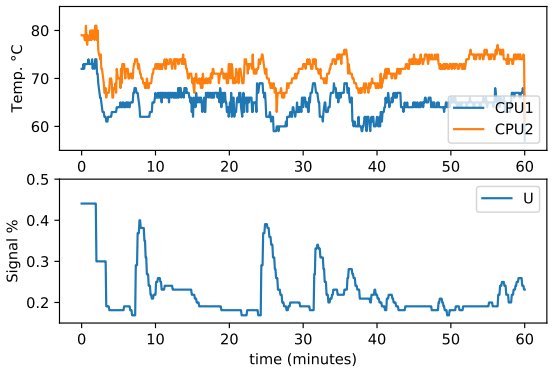

Figure 11. Results for control scenario (a) CPU focused control, which shows temperatures below threshold (**upper** panel) and control signal (**lower**).

Chamber-focused controller. Here the chamber is given equal weight to each CPU, $\hat{w}^{\mathrm{CHA}} = 1$, with a chamber set-point at 28 °C. The other MPC parameters are the same as for the Server focused controller.

Figure 12 shows results from the corresponding experiment with chamber-focused control. The upper panel shows the chamber temperature, which hovers around a 28.2 °C average over the experiment—not far from the 28 °C set-point. The fan control signal displayed in the lower panel is relatively stable around a 48.8 % average. Average CPU temperatures are 57.3 °C and 60.5 °C for CPU1 and CPU2, which is well below the set-point limit at 75 °C.

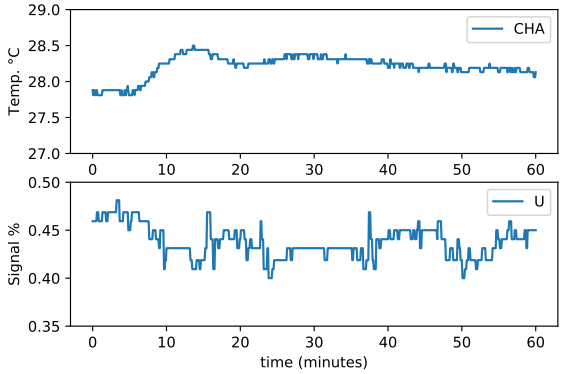

Figure 12. Results for control scenario (b) chamber focus which shows chamber temperatures close to set-point (**upper**) and control signal (**lower**).

The chamber-focused temperature controller could be used where one targets a specific output temperature, for example, a smart heating application for a room, a greenhouse, or a food-drying plant.

4.3. Transfer Learning

To investigate the transfer learning capacity of the model, it is fine-tuned (according to the illustration in Figure 6) to a smaller but similar experimental set-up in our data center experimental research facilities. The set-up consists of a wind tunnel used for testing individual or pairs of computer servers under controlled environmental conditions with respect to temperature, pressure, and humidity of the cooling airflow; see Sarkinen et al. [38] for a

detailed description. A single OCP Windmill v2 server was used for the transfer learning experiment (the same server type as for training the original model). An exit chamber where the server exhaust air mixes are also present in the wind tunnel set-up, although it is of different dimensions and design compared to the first experiment.

Sensor data from an experiment running a 5 h sequence that varied computational load and fan speed on a single server was used for fine-tuning the transferred model over up to 100 additional training epochs. Meta parameters, θ_*, were directly transferred. Half the data were used as training data and half as test data. Hyper-parameters, ϕ, were taken as same as when training the source model—hence there was no need for a validation set. Model parameters related to server components were initialized at the average of those for the six servers of the source cluster, while the chamber parameters were directly transferred to the target model.

Table 2 compares the performance of the directly transferred model with a model fine-tuned over progressively more epochs, and finally with a model re-trained entirely from scratch. Each result is an average of over 10 repeated training experiments. Already after a few epochs, we see that fine-tuning shows clear improvements in RMSE over the direct transfer model (first row) and re-start (last row). Figure 13 visualizes in log-scale the RMSE scores averaged over both CPUs and chamber. We note that the model that was re-initialized and trained from scratch does not match the transferred and fine-tuned model even after 100 training epochs, except for CPU2 which sees a slight improvement over direct transfer in Table 2 (crossing occurs after about 50 epochs).

Table 2. RMSE results for transfer learning (°C).

Mode	Epoch	CPU1	CPU2	Chamber
transfer	0	2	2.02	0.85
finetune	1	1.84	1.67	0.72
finetune	10	1.88	1.53	0.74
finetune	100	1.9	1.47	0.66
re-start	100	2.2	1.96	1.63

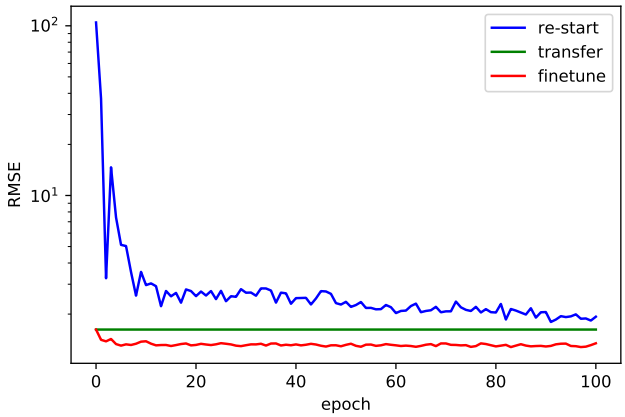

Figure 13. Total test set RMSE by epoch for transfer learning example.

A closer look at how the parameters are updated revealed that it is the chamber that is most affected by fine-tuning. It is also the physical set-up of the exit chamber that is most different compared to the original experiment. The server model is the same for both set-ups, which is a plausible explanation for why their related model parameters change less with fine-tuning.

5. Discussion

The hierarchical formulation with meta parameters that are used to connect related components of a modular design provides an intuitive and efficient means for exploiting

transfer learning when implementing the model on a server set-up of a similar design. It is similar to soft-parameter sharing in multi-task learning [39] in that parameters are not shared explicitly but instead tied together by regularization that constrains the distance between the weights of models for different tasks. We are inclined to consider this a type of meta-learning as the set-up generalizes the training of target models across potentially many different configurations [27,35]. The PIDD model can thus support modular designs where components are combined according to local conditions.

The physics-informed structure should be more robust and generalize well to unseen conditions even with fewer training samples, as is also argued by [24]. The results from our experiments indicate that this carries through also to transfer learning.

Our study shares the use of a component-based approach for modeling thermal dynamics with [23] and their aim to improve the reusability and generalization of the models. Like [24], the proposed PIDD model is guided by physics and utilizes a neural network to model the complex thermal dynamics of the system. In contrast, the PIDD model specifically targets edge data centers and incorporates a hierarchical modular structure.

Our study differs from [25] in that we focus on edge data centers rather than large-scale data centers and propose a physics-informed data-driven model with a hierarchical modular structure that connects related components through meta-parameters. Additionally, our study places greater emphasis on energy efficiency through heat reuse.

In summary, our study builds upon the ideas in [23–25], but proposes a novel approach to thermal management of edge data centers that combines physics-informed modeling, hierarchical modular design, and transfer learning. This design facilitates transfer learning, allowing the model to be adapted to different local conditions, which extends the reusability and generalization of the model compared to previous work.

Limitations. While we believe that the study presents a promising approach to transfer learning for modular edge data center systems, there are several limitations to consider:

Limited experimental setup: the study was conducted on a small-scale experimental setup with only six servers, which may not be representative of larger data centers. Further work is needed to evaluate the proposed approach's reliability and scalability on larger edge data centers with different server hardware, more realistic load profiles, and space heating requirements.

Transfer learning to more diverse conditions: while the approach shows promise in facilitating transfer learning when implementing the model on a server set-up of a similar design, further testing is required to determine the transferability to more diverse conditions.

Modeling assumptions: the proposed model is based on a physics-informed data-driven approach, which makes several assumptions about the system's thermal behavior. The model's performance may be affected by the accuracy of these assumptions, and further work is required to evaluate the model's robustness to different system configurations and operational conditions.

Computational complexity: while the proposed model can execute in real-time to control the system under complex constraints, the computational complexity may be an issue for models of larger data centers. We did not investigate the scalability of the proposed approach, and further work is needed to evaluate the model in larger settings.

6. Conclusions

This study investigates the use of transfer learning and a modular design to facilitate the adaption of a pre-trained model to different local conditions for edge data centers. The objective is to improve energy efficiency and optimize the overall electricity consumption of a system while meeting local space heating constraints. A Physics-Informed Data-Driven Recurrent Neural Network (PIDD RNN) was trained on a scale-model experiment representing the principal thermal dynamics of a small data center.

The PIDD RNN model performs better than other alternative models, with a lower RMSE value of 1.69 compared to a persistence baseline (RMSE: 4.95) and non-trivial

alternative models based on a conventional RNN or State Space Model (RMSE: 3.18 and 2.66, respectively). It was subsequently used for model predictive control of the cooling fans to maintain the exhaust chamber at a set-point temperature while ensuring safe operation thermal limits. The proposed PIDD RNN model can execute in real-time to control the system under complex constraints.

The hierarchical formulation with meta parameters used to connect related components provides an efficient means for exploiting transfer learning when implementing the model on a server set-up of an alternative design. Fine-tuning on a small data-set obtains low test-set RMSE scores after only a few epochs that are not reached even after hundreds of epochs for a re-initialized model trained from scratch.

The study is limited to a small-scale experimental setup, which may not be representative of larger data centers. There is therefore need for further testing to determine the transferability of the proposed approach to more diverse conditions and to investigate the potential complexity of implementing the model in larger data center settings.

Future work. Further work is required to evaluate the reliability and scalability of the proposed modeling and transfer learning approach, particularly by considering a larger edge data center with different server hardware, more realistic load profiles, and space heating requirements. A natural extension would also include more than one source in the pre-training stage by employing data from various edge data center experiment test beds. Similarly, the design should also be tested for transfer to more diverse conditions.

Author Contributions: Conceptualization, R.B. and J.G.; methodology, R.B.; software, R.B.; validation, R.B., J.G. and F.S.; investigation, R.B.; writing—original draft preparation, R.B.; writing—review and editing, J.G. and F.S.; visualization, R.B.; supervision, J.G. and F.S.; funding acquisition, J.G. All authors have read and agreed to the published version of the manuscript.

Funding: Vinnova through the Celtic Next project AI-NET Aniara with project-ID C2019/3-2.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations and Notation

The following abbreviations are used in this manuscript:

CFD	Computational Fluid Dynamics
DC	Data Center
MPC	Model Predictive Control
MSE	Mean Square Error
MAE	Mean Average Error
OCP	Open Compute Project
PEM	Prediction Error Minimization
PWM	Pulse Width Modulation
PID	Proportional-Integral-Derivative
PIDD	Physical Informed Data Driven
RHC	Receding Horizon Control
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SSM	State Space Model
ReLU	Rectified Linear Unit (activation function)
Hub	Huber (loss function)
AMB	ambient (in experiments)

CHA	chamber (in experiments)
CPU1	1st CPU (in experiments)
CPU2	2nd CPU (in experiments)
HEX1	heat exchanger 1 (in experiments)
HEX2	heat exchanger 2 (in experiments)
SRV	server (in experiments)

Notation: The temperatures of the system are denoted with the letter x, computational load by p, the server fan control signal by u, and an unobserved unit-free mass flow f. Superscript labels are used to indicate the position of the component for which the temperature is observed, such that for CPU1 we write $x_{i,t}^{\text{CPU1}}$, where subscripts i and t determine, respectively, for which server and at what time the observation was made.

In addition to physical position, the superscript is also used to distinguish related quantities, for example, the temperatures, $x_{i,t}^{\text{Z1,IN}}$, and $x_{i,t}^{\text{Z1,OUT}}$, which are the temperatures of the air that enter and exit zone 1 in the server (which houses CPU 1). Similarly, it is used to enumerate trainable model parameters, for example $\theta_i^{\text{CPU1,D}}$ and $\theta_i^{\text{Z1,RCD}}$, for the lump thermal capacitance of CPU1 and its conductive thermal resistance (in zone 1) with the heat exchanger.

Note that the (lowercase) Greek letter θ (with different subscripts and superscripts) is used throughout the text to denote trainable parameters. Capital letter Θ denotes the set of all trainable parameters for the six servers and the chamber module of the first experiment.

The following table is a guide to the mathematical notation:

x_t^{AMB}	Ambient temperature at time t
$x_{i,t}^{\text{CPU1}}$	Temperature of CPU 1 for server i at time t
$p_{i,t}^{\text{CPU1}}$	Power input through CPU 1 for server i at time t
$x_{i,t}^{\text{HEX1}}$	Temperature of heat exchanger of CPU 1 for server i at time t
$q_{i,t}^{\text{CD1}}$	Conductive heat transfer in zone 1 server i at time t
$q_{i,t}^{\text{CV1}}$	Convective heat transfer in zone 1 server i at time t
$x_{i,t}^{\text{Z1,IN}}$	Temperature of air that enter zone 1 of server i at time t
$x_{i,t}^{\text{Z1,OUT}}$	Temperature of air that exit zone 1 of server i at time t
$f_{i,t}^{\text{Z1}}$	Mass (air) flow in zone 1 server i at time t
$R_{i,t}^{\text{CV1}}$	Convective thermal resistance in zone 1 of server i at time t
$x_t^{\text{CHA,IN}}$	Temperature of air entering exhaust chamber at time t
x_t^{CHA}	Temperature of lump mass of exhaust chamber at time t
$q_t^{\text{CHA,CD}}$	Conductive heat transfer in chamber at time t
$q_t^{\text{CHA,CV}}$	Convective heat transfer in chamber at time t
$\theta_i^{\text{CPU1,D}}$	Trainable parameter for lump thermal capacitance (CPU 1)
$\theta_i^{\text{Z1,RCD}}$	Trainable parameter for conductive thermal resistance (zone 1)
θ_i^{F0}	Trainable regression intercept parameter for fan airflow for server i
θ_i^{F1}	Trainable regression slope parameter for fan airflow for server i
$\theta_i^{\text{Z1,L0}}$	Trainable regression intercept parameter for zone 1 airflow for server i
$\theta_i^{\text{Z1,L1}}$	Trainable regression slope parameter for zone 1 airflow for server i
$\theta^{\text{CHA,D}}$	Trainable parameter for lump thermal capacitance of chamber
$p_{i,*}^{\text{CPU1}}$	Steady state input power for CPU 1
$q_{i,*}^{\text{CD1}}$	Steady state flow heat for zone 1 of server i
$x_{i,*}^{\text{CPU1}}$	Steady state temperature of CPU 1 of server i
$x_{i,*}^{\text{HEX1}}$	Steady state temperature for heat exchanger 1 of server i
w^{CHA}	Parameter that control the relative weight of chamber set-point
τ	Parameter for the length of the model predictive control horizon
λ^{SIG}	Penalty parameter for signal strength
λ^{SIM}	Penalty parameter for signal smoothness
λ^{SRV}	Penalty parameter for signal heterogeneity
γ	Hyper-parameter for strength of hierarchical regularization
σ	Hyper-parameter for strength of noise smoothing
n	Hyper-parameter for length of prediction window
r	Hyper-parameter for learning rate
K	Hyper-parameter for number of training epochs

Note that variables labelled with CPU1 (HEX1,CV1, or CD1) also have an CPU2 (HEX2, CV2, or CD2) analogue. Similarly, only Z1 labels (for zone 1) are tabled—analogues here exist for other zones: Z0, Z2, and Z3. Hierarchical (trainable) parameters are denoted with subscript $*$ in place of server index i (and for chamber). Temperature set-points are marked with a hat, for example, $\hat{x}^{CPU1}$ and $\hat{x}^{CHA}$. Trainable parameters for the second set-up instead are marked above with the ˜ sign.

Appendix A. Model Details

This appendix contains a more detailed description of the components of the physics-informed model. The reader can refer to Figure 5 for visual guidance and notation.

Appendix A.1. Server Model

Recall Equation (1) from the main text, which is the differential equation for the temperature dynamics of a generic node j subject to heat convection and conduction that was also used by Lucchese and Johansson [18]

$$d_j \dot{x} = p + q_j^{\mathrm{CV}} + q_j^{\mathrm{CD}} \tag{A1}$$

We rewrite it for each component of our system in our own notation, and then apply Euler's method to obtain approximations for the discrete time step Δt. For the first CPU, CPU1, of server i

$$x_{i,t+1}^{\mathrm{CPU1}} = x_{i,t}^{\mathrm{CPU1}} + \theta_i^{\mathrm{CPU1,D}} \left(p_{i,t}^{\mathrm{CPU1}} + q_{i,t}^{\mathrm{CD1}} \right), \tag{A2}$$

with a time-constant related to the time-step Δt and the lumped thermal capacitance of the continuous model

$$d_i^{\mathrm{CPU1}} = \theta_i^{\mathrm{CPU1,D}} \Delta t. \tag{A3}$$

For the first on-chip heat exchanger, HEX1,

$$x_{i,t+1}^{\mathrm{HEX1}} = x_{i,t}^{\mathrm{HEX1}} + \theta_i^{\mathrm{HEX1,D}} \left(q_{i,t}^{\mathrm{CV1}} - q_{i,t}^{\mathrm{CD1}} \right), \tag{A4}$$

where the time-constant is separate from CPU1 since their lumped thermal capacitance may be different. Note also the changed sign of the conductive heat transfer term. The equations for CPU2 and HEX2 within the same server are entirely analogous, with their own time-constants, self-heating and heat transfer terms.

Heat transfer. The conductive heat transfer is written

$$q_{i,t}^{\mathrm{CD1}} = -\frac{x_{i,t}^{\mathrm{CPU1}} - x_{i,t}^{\mathrm{HEX1}}}{\theta_i^{\mathrm{Z1,RCD}}}, \tag{A5}$$

assuming constant thermal conductive resistance.

The convective heat transfer is determined in relation to the temperature of the mass flow entering the zone $x_{i,t}^{\mathrm{Z1,IN}}$

$$q_{i,t}^{\mathrm{CV1}} = -\frac{x_{i,t}^{\mathrm{HEX1}} - x_{i,t}^{\mathrm{Z1,IN}}}{R_i^{\mathrm{CV1}}(f_i^{\mathrm{Z1}})}, \tag{A6}$$

with convective thermal resistance assumed to depend on the mass flow $f_{i,t}^{\mathrm{Z1}}$ according to

$$\left(R_i^{\mathrm{CV1}}(f_i^{\mathrm{Z1}}) \right)^{-1} = \frac{f_{i,t}^{\mathrm{Z1}}}{\theta_i^{\mathrm{Z1,RCV0}} + \theta_i^{\mathrm{Z1,RCV1}} f_{i,t}^{\mathrm{Z1}}}, \tag{A7}$$

for constants $\theta_i^{\mathrm{Z1,RCV0}}$ and $\theta_i^{\mathrm{Z1,RCV1}}$ that determine the non-linear dependence of the mass-flow on the strength of the convective heat transfer [40].

Steady state. From Equation (A2) we note that a steady state for the CPU and HEX temperatures are reached if the power added from the computational load is balanced by the heat transferred by convection between the two components, or

$$p_{i,*}^{\mathrm{CPU1}} = -q_{i,*}^{\mathrm{CD1}} = -\frac{x_{i,*}^{\mathrm{CPU1}} - x_{i,*}^{\mathrm{HEX1}}}{\theta_i^{\mathrm{Z1,RCD}}}, \tag{A8}$$

that is, for a HEX temperature given by

$$x_{i,*}^{\mathrm{HEX1}} = x_{i,*}^{\mathrm{CPU1}} - \theta_i^{\mathrm{Z1,RCD}} p_{i,*}^{\mathrm{CPU1}}, \tag{A9}$$

with a corresponding equation that relates CPU2 and HEX2 for each server.

The steady state relations are used in the training procedure to determine starting values for HEX temperatures, as these are not observed in the experiment, and instead must be inferred from CPU temperatures.

Mass flow. A model for the airflow is also provided in [18], which however is not used in the work presented here. Instead we use a formulation in terms of the soft-max function.

The total airflow entering (zone 0 of) server i depends on the speed at which we run its fans. We assume a linear relation with the fan control signal $u_{i,t}$ according to

$$f_{i,t}^{\mathrm{Z0}} = \theta_i^{\mathrm{F0}} + \theta_i^{\mathrm{F1}} u_{i,t}, \tag{A10}$$

for trainable regression parameters θ_i^{F0} and θ_i^{F1}. This flow is split into three parts where the first fraction $L_{i,t}^{\mathrm{Z1}}$ goes directly to zone 1, the second fraction $L_{i,t}^{\mathrm{Z2}}$ to zone 2 and the third $L_{i,t}^{\mathrm{Z3}}$ bypass all electronic components. We assume no re-circulation flows, and hence, due to mass conservation, the total flow to each zone is now

$$f_{i,t}^{\mathrm{Z1}} = L_{i,t}^{\mathrm{Z1}} f_{i,t}^{\mathrm{Z0}}, \quad f_{i,t}^{\mathrm{Z2}} = \left(L_{i,t}^{\mathrm{Z1}} + L_{i,t}^{\mathrm{Z2}}\right) f_{i,t}^{\mathrm{Z0}}, \quad \text{and} \quad f_{i,t}^{\mathrm{Z3}} = f_{i,t}^{\mathrm{Z0}}, \tag{A11}$$

where we assumed

$$L_{i,t}^{\mathrm{Z1}} \propto \exp\left(\theta_i^{\mathrm{Z1,L0}} + \theta_i^{\mathrm{Z1,L1}} u_{i,t}\right), \tag{A12}$$

and corresponding equations for zone 2 and zone 3, normalized by

$$L_{i,t}^{\mathrm{Z1}} + L_{i,t}^{\mathrm{Z2}} + L_{i,t}^{\mathrm{Z3}} = 1, \tag{A13}$$

which completes the air flow model. Note that the use of the soft-max function makes our formulation similar to multinomial logistic regression.

Air temperatures. The temperature of the airflow entering zone 0 is that of the ambient air. As this is assumed to go directly to zone 1 we then have

$$x_{i,t}^{\mathrm{Z1,IN}} = x_t^{\mathrm{AMB}}. \tag{A14}$$

The airflow that exits zone 1 has been heated by convection

$$x_{i,t}^{\mathrm{Z1,OUT}} = x_{i,t}^{\mathrm{Z1,IN}} - \frac{q_{i,t}^{\mathrm{CV1}}}{\rho c f_{i,t}^{\mathrm{Z1}}}. \tag{A15}$$

The airflow entering zone 2 is a mixture of air that passed zone 1 and air that come directly from the ambient. We write its temperature

$$x_{i,t}^{\mathrm{Z2,IN}} = \lambda_{i,t}^{\mathrm{Z1}} x_{i,t}^{\mathrm{Z1,OUT}} + \lambda_{i,t}^{\mathrm{Z2}} x_t^{\mathrm{AMB}}, \tag{A16}$$

and for the exit air

$$x_{i,t}^{\mathrm{Z2,OUT}} = x_{i,t}^{\mathrm{Z2,IN}} - \frac{q_{i,t}^{\mathrm{CV2}}}{\rho c f_{i,t}^{\mathrm{Z2}}}, \tag{A17}$$

and finally for the temperature of the airflow in zone 3

$$x_{i,t}^{Z3} = (1 - \lambda_{i,t}^{Z3}) x_{i,t}^{Z2,OUT} + \lambda_{i,t}^{Z3} x_t^{AMB}. \tag{A18}$$

Appendix A.2. Chamber Model

The air flow entering the chamber is the sum of all air flows from servers

$$f_t^{CHA} = \sum_{i=1}^{6} f_{i,t}^{Z0}, \tag{A19}$$

and its temperature is taken as a weighted average

$$x_t^{CHA,IN} = \frac{\sum_{i=1}^{6} f_{i,t}^{Z0} * x_{i,t}^{Z3}}{f_t^{CHA,IN}}. \tag{A20}$$

For the chamber we assume a single lump mass whose temperature is described by

$$x_{t+1}^{CHA} = x_t^{CHA} + \theta^{CHA,D} \left(q_t^{CHA,CV} + q_t^{CHA,CD} \right), \tag{A21}$$

that interacts with the airflow from the servers through convection

$$q_t^{CHA,CV} = -\frac{x_t^{CHA} - x_t^{CHA,IN}}{\theta^{CHA,RCV0} + \theta^{CHA,RCV1} f_t^{CHA}} f_t^{CHA}, \tag{A22}$$

and which additionally loses heat to the environment by conduction

$$q_t^{CHA,CD} = -\frac{x_t^{CHA} - x_t^{AMB}}{\theta^{CHA,RCD}}. \tag{A23}$$

This completes the detailed description of the Thermal model. Section 3.3 and forward describes how it is used for inference and training after it is integrated in a RNN-like model structure.

References

1. Khalaj, A.; Halgamuge, S.K. A Review on efficient thermal management of air- and liquid-cooled data centers: From chip to the cooling system. *Appl. Energy* **2017**, *205*, 1165–1188. [CrossRef]
2. Athavale, J.; Yoda, M.; Joshi, Y. Comparison of data driven modeling approaches for temperature prediction in data centers. *Int. J. Heat Mass Transf.* **2019**, *135*, 1039–1052. [CrossRef]
3. Manaserh, Y.M.; Tradat, M.I.; Bani-Hani, D.; Alfallah, A.; Sammakia, B.G.; Nemati, K.; Seymour, M.J. Machine learning assisted development of IT equipment compact models for data centers energy planning. *Appl. Energy* **2022**, *305*, 117846. [CrossRef]
4. Wang, Z.; Bash, C.; Tolia, N.; Marwah, M.; Zhu, X.; Ranganathan, P. Optimal fan speed control for thermal management of servers. In Proceedings of the ASME InterPack Conference 2009, IPACK2009, San Francisco, CA, USA, 19–23 July 2009; Volume 2, pp. 709–719.
5. Han, X.; Joshi, Y. Energy reduction in server cooling via real time thermal control. In Proceedings of the Annual IEEE Semiconductor Thermal Measurement and Management Symposium, San Jose, CA, USA, 18–22 March 2012; pp. 74–81.
6. Wang, X.; Liu, Y.; Tian, T.; Li, J. Directly air-cooled compact looped heat pipe module for high power servers with extremely low power usage effectiveness. *Appl. Energy* **2022**, *319*, 119279. [CrossRef]
7. Li, J.; Zhou, G.; Tian, T.; Li, X. A new cooling strategy for edge computing servers using compact looped heat pipe. *Appl. Therm. Eng.* **2021**, *187*, 116599. [CrossRef]
8. Iyengar, M.; Hamann, H.; Schmidt, R.R.; Vangilder, J. Comparison between numerical and experimental temperature distributions in a small data center test cell. In Proceedings of the 2007 ASME InterPack Conference, IPACK 2007, Vancouver, BC, Canada, 8–12 July 2007; Volume 1, pp. 819–826.
9. Wibron, E.; Ljung, A.L.; Lundström, T. Computational Fluid Dynamics Modeling and Validating Experiments of Airflow in a Data Center. *Energies* **2018**, *11*, 644. [CrossRef]
10. Pardey, Z.; Demetriou, D.; Erden, H.; VanGilder, J.; Khalifa, H.; Schmidt, R. Proposal for standard compact server model for transient data center simulations. *Ashrae Trans.* **2015**, *121*, 413–421.

11. VanGilder, J.; Healey, C.; Pardey, Z.; Zhang, X. A compact server model for transient data center simulations. *Ashrae Trans.* **2013**, *119*, 358–370.
12. Erden, H.; Ezzat Khalifa, H.; Schmidt, R. Transient thermal response of servers through air temperature measurements. In Proceedings of the International Electronic Packaging Technical Conference and Exhibition, Burlingame, CA, USA, 16–18 July 2013; Volume 2. [CrossRef]
13. Lucchese, R.; Olsson, J.; Ljung, A.L.; Garcia-Gabin, W.; Varagnolo, D. Energy savings in data centers: A framework for modelling and control of servers' cooling. *IFAC-PapersOnLine* **2017**, *50*, 9050–9057. [CrossRef]
14. Eriksson, M.; Lucchese, R.; Gustafsson, J.; Ljung, A.L.; Mousavi, A.; Varagnolo, D. Monitoring and modelling open compute servers. In Proceedings of the IECON 2017—43rd Annual Conference of the IEEE Industrial Electronics Society, Beijing, China, 29 October–1 November 2017; pp. 7177–7184.
15. Open Compute Project. Open Compute Project, 2019. Available online: https://www.opencompute.org/about (accessed on 25 February 2023).
16. VanGilder, J.W.; Healey, C.M.; Condor, M.; Tian, W.; Menusier, Q. A Compact Cooling-System Model for Transient Data Center Simulations. In Proceedings of the 2018 17th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), San Diego, CA, USA, 29 May–1 June 2018. [CrossRef]
17. Healey, C.; VanGilder, J.; Condor, M.; Tian, W. Transient Data Center Temperatures after a Primary Power Outage. In Proceedings of the 2018 17th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), San Diego, CA, USA, 29 May–1 June 2018; pp. 865–870. [CrossRef]
18. Lucchese, R.; Johansson, A. On energy efficient flow provisioning in air-cooled data servers. *Control. Eng. Pract.* **2019**, *89*, 103–112. [CrossRef]
19. Lucchese, R.; Johansson, A. On server cooling policies for heat recovery: Exhaust air properties of an Open Compute Windmill V2 platform. In Proceedings of the 2019 IEEE Conference on Control Technology and Applications (CCTA), Hong Kong, China, 19–21 August 2019; pp. 1049–1055. [CrossRef]
20. Brannvall, R.; Sarkinen, J.; Svartholm, J.; Gustafsson, J.; Summers, J. Digital Twin for Tuning of Server Fan Controllers. In Proceedings of the 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), Helsinki, Finland, 22–25 July 2019. [CrossRef]
21. Brännvall, R.; Mattson, L.; Lundmark, E.; Vesterlund, M. Data Center Excess Heat Recovery: A Case Study of Apple Drying. In Proceedings of the ECOS 2020: Proceedings of the 33rd International Conference on Efficiency, Cost, Optimization, Simulation and Enviromental Impact of Energy Systems. ECOS 2020 Local Organizing Committee, Osaka, Japan, 29 June–3 July 2020; pp. 2165–2174.
22. Xia, L.; Chen, G.; Wu, T.; Gao, Y.; Mohammadzadeh, A.; Ghaderpour, E. Optimal Intelligent Control for Doubly Fed Induction Generators. *Mathematics* **2023**, *11*, 20. [CrossRef]
23. Geyer, P.; Singaravel, S. Component-based machine learning for performance prediction in building design. *Appl. Energy* **2018**, *228*, 1439–1453. [CrossRef]
24. Gokhale, G.; Claessens, B.; Develder, C. Physics informed neural networks for control oriented thermal modeling of buildings. *Appl. Energy* **2022**, *314*, 118852. [CrossRef]
25. Berezovskaya, Y.; Yang, C.W.; Mousavi, A.; Vyatkin, V.; Minde, T.B. Modular Model of a Data Centre as a Tool for Improving Its Energy Efficiency. *IEEE Access* **2020**, *8*, 46559–46573. [CrossRef]
26. Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; He, Q. A Comprehensive Survey on Transfer Learning. *Proc. IEEE* **2021**, *109*, 43–76. [CrossRef]
27. Finn, C.; Abbeel, P.; Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 1126–1135.
28. Gelman, A.; Hill, J. Multilevel regression. In *Data Analysis Using Regression and Multilevel/Hierarchical Models*; Cambridge University Press: Cambridge, UK, 2007; pp. 235–236. [CrossRef]
29. Rawlings, J.; Mayne, D.; Diehl, M. *Model Predictive Control: Theory, Computation, and Design*; Nob Hill Publishing: San Francisco, CA, USA, 2017.
30. Åström, K.J.; Hägglund, T. The future of PID control. *Control. Eng. Pract.* **2001**, *9*, 1163–1175. [CrossRef]
31. Ko, J.S.; Huh, J.H.; Kim, J.C. Improvement of Energy Efficiency and Control Performance of Cooling System Fan Applied to Industry 4.0 Data Center. *Electronics* **2019**, *8*, 582. [CrossRef]
32. Gustafsson, J.; Fredriksson, S.; Nilsson-Mäki, M.; Olsson, D.; Sarkinen, J.; Niska, H.; Seyvet, N.; Minde, T.B.; Summers, J. A demonstration of monitoring and measuring data centers for energy efficiency using opensource tools. In Proceedings of the e-Energy 2018—Proceedings of the 9th ACM International Conference on Future Energy Systems, Karlsruhe Germany, 12–15 June 2018, pp. 506–512.
33. McKay, M.D.; Beckman, R.J.; Conover, W.J. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics* **1979**, *21*, 239. [CrossRef]
34. Rai, R.; Sahu, C.K. Driven by Data or Derived Through Physics? A Review of Hybrid Physics Guided Machine Learning Techniques With Cyber-Physical System (CPS) Focus. *IEEE Access* **2020**, *8*, 71050–71073. [CrossRef]
35. Li, Y.; Wang, J.; Huang, Z.; Gao, R.X. Physics-informed meta learning for machining tool wear prediction. *J. Manuf. Syst.* **2022**, *62*, 17–27. [CrossRef]

36. Huber, P.J. Robust Estimation of a Location Parameter. *Ann. Math. Stat.* **1964**, *35*, 73–101. [CrossRef]
37. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations (ICLR), Banff, AB, Canada, 14–16 April 2014.
38. Sarkinen, J.; Brännvall, R.; Gustafsson, J.; Summers, J. Experimental Analysis of Server Fan Control Strategies for Improved Data Center Air-based Thermal Management. In Proceedings of The Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm 2020), Orlando, FL, USA, 21–23 July 2020.
39. Duong, L.; Cohn, T.; Bird, S.; Cook, P. Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), Beijing, China, 26–31 July 2015. [CrossRef]
40. Moffat, R.J. Modeling Air-Cooled Heat Sinks as Heat Exchangers. In Proceedings of the Twenty-Third Annual IEEE Semiconductor Thermal Measurement and Management Symposium, San Jose, CA, USA, 18–22 March 2007; pp. 200–207.