algorithms

MDPI

Article

Model Predictive Evolutionary Temperature Control via Neural-Network-Based Digital Twins

Cihan Ates *,†, Dogan Bicat †, Radoslav Yankov, Joel Arweiler, Rainer Koch and Hans-Jörg Bauer

Institute of Thermal Turbomachinery, Karlsruhe Institute of Technology (KIT), 76137 Karlsruhe, Germany; rainer.koch@kit.edu (R.K.); hans-joerg.bauer@kit.edu (H.-J.B.)

* Correspondence: cihan.ates@kit.edu; Tel.: +49-72160844703

† These authors contributed equally to this work.

Abstract: In this study, we propose a population-based, data-driven intelligent controller that leverages neural-network-based digital twins for hypothesis testing. Initially, a diverse set of control laws is generated using genetic programming with the digital twin of the system, facilitating a robust response to unknown disturbances. During inference, the trained digital twin is utilized to virtually test alternative control actions for a multi-objective optimization task associated with each control action. Subsequently, the best policy is applied to the system. To evaluate the proposed model predictive control pipeline, experiments are conducted on a multi-mode heat transfer test rig. The objective is to achieve homogeneous cooling over the surface, minimizing the occurrence of hot spots and energy consumption. The measured variable vector comprises high dimensional infrared camera measurements arranged as a sequence (655,360 inputs), while the control variable includes power settings for fans responsible for convective cooling (3 outputs). Disturbances are induced by randomly altering the local heat loads. The findings reveal that by utilizing an evolutionary algorithm on measured data, a population of control laws can be effectively learned in the virtual space. This empowers the system to deliver robust performance. Significantly, the digital twin-assisted, population-based model predictive control (MPC) pipeline emerges as a superior approach compared to individual control models, especially when facing sudden and random changes in local heat loads. Leveraging the digital twin to virtually test alternative control policies leads to substantial improvements in the controller's performance, even with limited training data.

Keywords: model predictive control; digital twin; neural network; deep learning; genetic programming; evolutionary algorithm; heat transfer; temperature control; data driven control; data-driven engineering

Citation: Ates, C.; Bicat, D.; Yankov, R.; Arweiler, J.; Koch, R.; Bauer, H.-J. Model Predictive Evolutionary Temperature Control via Neural-Network-Based Digital Twins. *Algorithms* **2023**, *16*, 387. https://doi.org/10.3390/a16080387

Academic Editor: Sándor Szénási and Gábor Kertész

Received: 28 July 2023
Revised: 8 August 2023
Accepted: 10 August 2023
Published: 12 August 2023

1. Introduction

Model Predictive Control (MPC) represents an advanced control method that distinguishes itself by employing a mathematical system model to anticipate future system behavior and makes proactive decisions in response to expected deviations from a set point. Unlike traditional control methods that reactively rely on past and present system behavior, MPC combines the principles of feedback control and numerical optimization to achieve optimal control outcomes. By continuously optimizing the system model in real-time, MPC determines an optimal trajectory for the manipulated variable.

The essential constituents of MPC encompass three fundamental elements: (i) a predictive model capturing the dynamics of the controlled system, (ii) a trajectory to be tracked and (iii) an optimal controller achieved through continuous optimization. Notably, only the initial value of the optimized output trajectory is implemented in the system, with the prediction and optimization process being repeated at each time step. This adaptive approach enables MPC to dynamically respond to changing conditions and deliver accurate control by considering future system behavior.

As the MPC revolves around the iterative solution of an optimization problem, it necessitates a system model, as well as a mathematical description of the corresponding

control law [1]. These models are traditionally derived from first principles or obtained through system identification techniques using measured data [2]. However, an attractive alternative approach is to directly implement an MPC controller using solely measured data, without relying on prior knowledge of an accurate model [3]. The data-driven approach particularly offers practical advantages in scenarios where (i) obtaining a precise model may be challenging, time-consuming and/or expensive to evaluate; (ii) the process is ill-defined; (iii) the process is time-variant or stochastic in nature. Herein, recent advancements in machine learning facilitates the creation of input–output-based digital twin models (DT) that do not require a thorough mathematical description of the process [4], enabling the implementation of intelligent controllers that can adapt to the system dynamics and change their control policies in real time. These techniques allow for treating the system and the physical process within itself as a black box [5], while maintaining good accuracy, by approximating the mapping from the input to the output space [6].

Examples of using machine learning in MPC cover a broad range of applications. In one of the early works, Liu and Atkeson combined the linear quadratic regulator with unsupervised clustering (k-nearest neighbor) [7]. Other shallow learning applications include Gaussian process modeling for the safe exploration of dynamical systems [8], the optimal energy management in commercial building micro-grids [9], heating, ventilation and air-conditioning (HVAC) control of a hospital surgery center [10]; Bayesian regression for safe predictive learning control [11], statistical time series modeling (ARIMA) for optimal energy management [9], random forests for HVAC systems [12] and support vector machines for milling [13]. Feed-forward neural network (NN) applications within an MPC framework can also be found in various disciplines. Piche et al. [14] implemented an NN to regulate set point changes in a polyethylene reactor, resulting in a 30% improvement in transition speed and a significant reduction in controlled variable fluctuations. The work of Mu and Rees [15] is another early example combining NNs with MPC to control the shaft speed of a gas turbine engine. Gas turbine models were created via a nonlinear autoregressive moving average model with exogenous inputs (NARMAX) and neural networks, enabling an improved control performance compared to PID controllers through various step tests. Afram et al. [16] employed NNs to develop a supervisory MPC for residential heating, ventilation, and air conditioning(HVAC) systems. Their approach successfully reduced the operating costs of the equipment, while ensuring that thermal comfort constraints were not compromised. In comparison to the fixed set-point (FSP) approach, the NN-augmented MPC achieved cost savings ranging from 6% to 73%, depending on the season. Similarly, Li et al. [17] investigated the application of an NN in the context of MPC, focusing on temperature control in a stirred tank reactor. Maddalena et al. [18] used NNs to generate control laws for MPC of voltage-current regulation in DC-DC converters. Similarly, Nubert et al. [19] demonstrated that the computation time of MPC can be drastically reduced with an NN controller for real-world robotic systems. In another study, Shin et al. [20] employed an NN in conjunction with MPC to control a simulated depropanizer in Aspen HYSYS, achieving a remarkable 60% reduction in settling time compared to a traditional PID controller. Nunez et al. [21] utilized a recurrent neural network (RNN) along with a particle swarm optimization (PSO) to model an industrial paste thickening process. The RNN-based MPC successfully maintained the desired concentration of the paste thickener, even in the presence of severe pump failures. Other RNN-based applications include solving a generic nonconvex control problem [22], optimal policy selection [23], fault diagnosis for HVAC systems [24], theory [25] and application [26] of a generic nonlinear system for open-loop simulations, multi-mode process control of a generic system [27], chemical reactor control [28], crystallization processes [29] annealing furnaces [30], N-tank problems [31] and corn production [32]. Achirei et al. [33] very recently introduced a model-based predictive controller that utilized the object map obtained from the convolutional neural network (CNN) detector and light detection and ranging (LIDAR) data to guide an omnidirectional robot to specific positions in a warehouse environment. For a more comprehensive understanding of recent advancements in model predictive control, we rec-

ommend consulting several key papers. Sand [34] offers a detailed comparison of different predictive control methods. In the realm of autonomous systems, Rosolia et al. [35] delve into the realm of data-driven control. For those interested in chemical process systems, Rawlings and Maravelias [36] provide a comprehensive exploration. Schwenzer et al. [37] present a holistic view of model predictive control, while Schweidtmann et al. [38] explore the integration of machine learning techniques in this context.

The literature review on NN-augmented MPC reveals the successful utilization of neural networks as effective approximators in MPC. Recent advancements in deep learning, such as neural networks with memory functions (RNNs) and specialized architectures capable of handling spatial information (CNNs), have further enhanced the representational power of data-driven models. Our contribution introduces a noteworthy progression within the domain of intelligent control strategies, stemmed from the strategic utilization of *ConvLSTM-based* digital twins' spatiotemporal pattern extraction abilities, enabling the successful implementation of a real-time population-based MPC in systems with many controlled variables. In particular, we propose a data-driven intelligent controller that leverages NN-based digital twins for hypothesis testing. Initially, a diverse set of control laws is generated using genetic programming with the digital twin of the system, facilitating a robust response to unknown disturbances. During inference, the trained digital twin is utilized to virtually test alternative control actions for a multi-objective optimization task associated with each control action. Subsequently, the best policy is applied to the system. To evaluate the proposed intelligent control pipeline, experiments are conducted on a multi-mode heat transfer test rig. The measured variable vector comprises high-dimensional infrared camera measurements arranged in a sequence (i.e., 655,360 inputs), while the control variable includes power settings for three fans responsible for convective cooling. Disturbances are induced by randomly altering the set point of local heat loads. The objective is to achieve homogeneous cooling over the surface, minimizing the occurrence of hot spots and energy consumption.

The structure of this paper is outlined below. Section 2 begins by providing an explanation of the experimental setup. Next, the model architecture of the NN-based digital twin is detailed. Then, the genetic programming implementation for generating a diverse control law population is described. Lastly, the design of the experiment used to evaluate the performance of MPC is presented. In Section 3, the predictive capabilities of the digital twin are assessed, followed by an evaluation of the MPC performance in real time test experiments. The paper concludes with a discussion about the current limitations of the approach, and the future directions.

2. Materials and Methods

2.1. Experimental Setup

This case study is motivated by the significant impact that high-temperature technical processes can have on the degradation of components. Accordingly, the proposed approach seeks to develop an intelligent controller using machine learning techniques to enable predictive cooling. The main objective is to generate control laws that facilitate a uniform temperature distribution, thereby minimizing the stresses and deformations arising from the formation of hot spots in the presence of unknown disturbances, or sudden changes in the thermal load.

The physical setup is designed as a multi-mode cooling problem. It consists of the following components (Figure 1):

- A copper plate—selected due to its high thermal conductivity in order to reduce the duration of the experiments;
- The copper plate is coated with a high-emissivity black paint (Nextel Velvet Coating 811-21) for improved signal-to-noise ratio;
- Three heating strips on the backside of the plate arranged in a "*Z*"-like pattern—310 mm × 17 mm, 24 V, 36 W;
- Three fans located on the perimeter of the plate—SUNON, 12 V, 1.62 W;

- A data acquisition module (myDAQ, NI) with an in-house built control unit;
- A mid-wave infrared camera—FLIR SC5000, (512 × 640) pixels;
- A LabVIEW interface for the real-time control of the system.

The infrared camera detects thermal radiation emitted by the copper plate and other components. The detected radiation is dependent on the plate surface temperature only for constant ambient conditions. This is achieved by conducting the tests in an air-conditioned room. This way, changes in the camera signal can be directly attributed to changes in the plate surface temperature.

A single experiment begins from an initial steady state s_0. A heating disturbance is then introduced through the strips and the fan loads are adjusted. The experiment lasts until a new steady state s_1 is reached. Figure 2 depicts the recordings of two experiments (top row and bottom row) from the training dataset. The first six frames show the steady-state temperature distribution reached at the end of the previous operating point, while the final two frames illustrate the new steady state under the new thermal loads and cooling configuration. There are two options for s_0. We either start with the system completely shut down (no heating or cooling), or we carry on with the steady state reached in the previous operating point.

Figure 1. The physical setup is devised as a multi-mode cooling problem, depicted in the upper section. The arrangement of fans around the copper plate is illustrated in the lower left corner, while the configuration of heating strips at the back of the plate, along with randomly placed thermal insulators, is shown in the lower right corner.

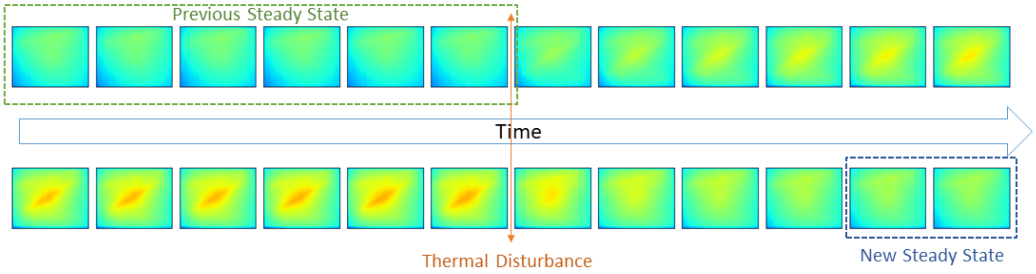


Figure 2. Visualization of two experiments from the dataset.

The following conditions are used to define the second steady state s_1 (Figure 3):

1. The per pixel percentage difference of consecutive frames after 16×16 max filter is less than 1.5%. The application of this max filter is required for two reasons. First, due to thermal inertia, the difference between consecutive frames can be small, and thus we increase the rigidity of the steady-state condition. Second, we reduce the impact of objects that have the same temperature in all frames (e.g., the frame around the plate).
2. The pixels with a 3% deviation in consecutive frames are less than 1% of the total pixels in a frame after a 16×16 max filter.

It is important to highlight that thermal insulation is absent at the slab edges as well as behind the resistance heating strips. The experimental configuration, illustrated in Figure 1, was executed in this manner. Once the system attains a steady state, it does so due to the interplay of forced and natural convection, conduction, and radiative heat transfer processes. In other words, the system was deliberately rendered more susceptible to environmental disturbances and fluctuations.

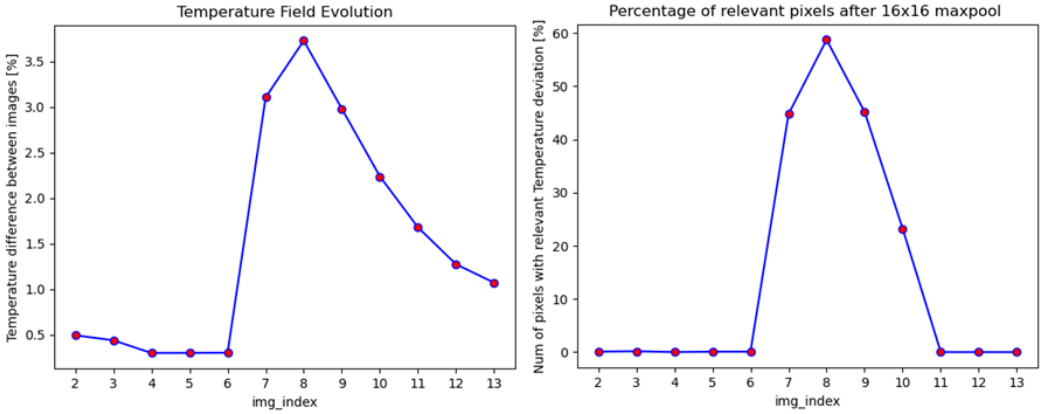


Figure 3. Evolution of consecutive frames over the course of an experiment. (**Left**): Mean pixel value change compared to the first steady-state condition. (**Right**): The percentage of pixels with a deviation larger than 3%.

2.2. Dataset

The training of the digital twin model necessitates a substantial amount of data. In this study, we performed 323 experiments, with each experiment saved as an individual HDF-file. The dataset was split into an 87.5/12.5% training-validation split and the frames were captured at a fixed rate of 1 image per 30 s. Each frame is stored as a grayscale image. The selection of the frame rate was based on preliminary experiments aimed at identifying the system's thermal inertia and response time. A higher frame rate would yield negligible differences between the images, making it challenging for the model to capture the temperature field's evolution. Conversely, longer time intervals may result in the loss of crucial information, such as heat propagation mechanisms and the formation and dissipation of local hot spots.

Within each experiment, the first six frames (2 min and 30 s) represent the initial steady state, denoted as s_0. This allows for the use of up to six frames as an input sequence, ensuring that all subsequent frames after thermal disturbances can be utilized as the ground truth at least once, maximizing the information within the dataset. Furthermore, to cover the parameter space for heating and cooling loads, we randomly sampled fan settings from a 0 to 100% workload with 20% increments and heating strip loads from 0 to 100% workload with 25% increments. In other words, fan settings and heating loads were randomly sampled from a pool of 6^3 and 5^3 possible configurations, respectively. These settings are also saved in the labels of the HDF5-files for post-processing purposes.

It is important to note that different initial conditions, heat loads, and fan settings influence the behavior of the system. Consequently, each configuration requires a varying amount of time to reach a steady-state operation, leading to variations in the sequence lengths across different experiments. Table 1 summarizes the distribution of experiment durations in the dataset.

Table 1. Summary of Experiment Duration in the Dataset.

Duration in Frames	Number of Experiments
8	32
9	103
10	85
11	48
12	35
13	16
14	4

2.3. Digital Twin

2.3.1. Model Architecture

The digital twin serves as the fundamental component of the proposed MPC pipeline. Hence, an extensive parametric study was conducted to identify an appropriate architecture and training protocol (see Appendix A for details). The model is based on Convolutional Long Short-Term Memory (ConvLSTM) cells [6]. Given the thermal inertia and slow evolution of the temperature field, it is anticipated that a smaller kernel size would yield better results. This hypothesis was confirmed through numerical experiments, where models utilizing a 3×3 kernel outperformed those employing a 5×5 kernel. Hence, the standard ConvLSTM cell with a 3×3 convolutional kernel is employed as the fundamental building block of the model. Following the lead of prior studies implementing ConvLSTM-based models, we adopt an auto-encoder structure. This choice offers two significant advantages. Firstly, it allows for the extraction of rich semantic information at a relatively low computational cost. Secondly, the learned compression of input data can considerably reduce the workload associated with the genetic programming-based optimization process, while enabling a high accuracy (mean absolute percentage error, Equation (1)). The architecture of the model is depicted in Figure 4.

The encoder is constructed by stacking seven convolutional layers with an increasing number of channels. Semantic information is extracted and the spatial dimension of the input is compressed by each layer. Various compression strategies, such as max pooling, average pooling, and strided convolution, were compared in the preliminary tests. The best results were achieved using a strided convolution with a stride of two. The structure of a single convolutional layer consists of (i) a ConvLSTM cell with a 3×3 kernel, (ii) a stride of two, (iii) L2 weight regularization and (iv) batch normalization. This structure enables the compression of the input image of size $n \times 512 \times 640 \times 1$ to a $n \times 4 \times 5 \times 256$ tensor, which contains rich semantic features. The parameter n represents the number of frames in the input sequence. The feature tensor is subsequently flattened into a vector for further processing.

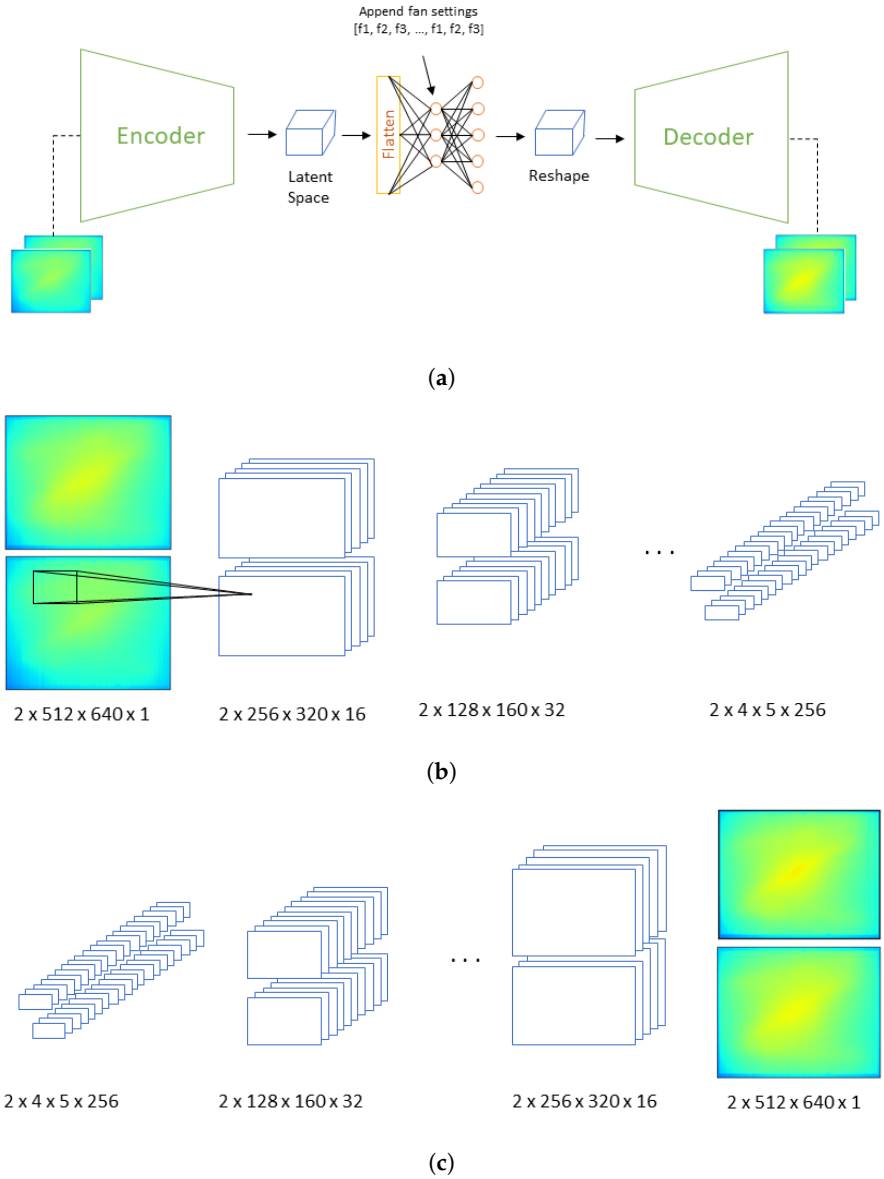


Figure 4. Deployed digital twin model architecture. (**a**) The next sequence predictor architecture with input and output sequence length of two. The fan settings vector is appended to the output of the first dense layer. (**b**) A detailed view of the encoder architecture. The input sequence is compressed to a latent state representation through 7 convolutional layers with (16, 32, 64, 64, 128, 128, 256) channels, respectively. (**c**) A detailed view of the decoder. The output of the second dense layer is reshaped into a $2 \times 4 \times 5 \times 256$ tensor. The reconstruction is conducted through 7 consecutive deconvolutional layers with (256, 128, 64, 32, 16, 8, 1) channels, respectively.

Following the encoder, a small fully connected network comprising two dense layers is employed. Due to the limited size of the training dataset, the number of dense layers is restricted to avoid overfitting, as supported by the parametric study conducted. Each dense layer includes (i) a dense layer with ReLU activation, (ii) a normal initializer, (iii) dropout regularization and (iv) batch normalization. The first dense layer consists of 2048 nodes and employs a dropout rate of 0.2. The optimal parameters were determined experimentally, considering the trade-off between computational burden and model performance. Next, the fan settings vector obtained from the experiment filename is appended. We select this point to introduce the meta-parameters since this is the layer containing the densest representation of the inputs. Hence, it is an ideal concatenation point that can serve as an

input to the GP-based controller. The fan settings vector comprises one hundred repetition of the duty cycle values for each fan. This extension is necessary since the original vector contains only three entries, one for each fan. By appending the initial vector to itself, its relevance to the output of the neural network is increased. This enables the model to learn the impact of the ventilators on the plate's temperature distribution. The size of the second dense layer is predetermined as $m \times 5120$, where m represents the length of the predicted sequence. This ensures that the output is rescaled to $m \times 4 \times 5 \times 256$ to initiate the upscaling of the prediction. To accurately capture the influence of the fans, dropout is disabled in this layer. The activation and initializer used are the same as in the first dense layer.

The final component of the model is the decoder, which mirrors the structure of the encoder. It consists of seven "deconvolutional" layers with a decreasing number of channels. Unlike the encoder, the deconvolutional blocks in the decoder upscale their inputs. Therefore, non-strided convolution and an upsampling layer, which doubles the height and width of the input, are utilized. The structure of the block includes (i) ConvLSTM cell (same as the encoder cell but with a stride of 1), (ii) batch normalization and (iii) upsampling layer. The decoder output has the shape $m \times 512 \times 640 \times 1$ and represents the prediction for the next "m" frames in the sequence.

It is pertinent to highlight that the digital twin model operates as a functional approximator. In essence, this model facilitates the mapping of the temperature distribution across a defined spatial region, over a specific time interval. This mapping takes the form of predicting the temperature field for the upcoming minute based on the temperature distribution observed in the preceding minute—a configuration often referred to as a sequence-to-sequence prediction. It is crucial to emphasize that this mapping encompasses not only the intricate temporal relationships but also the intricate spatial correlations within the field. These predictions are executed on a grid whose spatial resolution mirrors the input dimensions (512×640), preserving the structured nature of the grid and facilitating a seamless translation of insights between the physical domain and the digital representation. This framework, driven by the principles of neural networks, extends the familiar principles of function approximation to the realm of dynamic systems, such as the multi-mode heat transfer setup developed in this work.

2.3.2. Training Protocol

Determining optimal hyperparameters for training neural networks can pose a challenge and often necessitates an empirical approach. In our case, extensive testing was conducted, leading to the derivation of the following list of hyperparameters:

- The batch size was set to 16.
- The optimizer employed was Adam, utilizing a default initial learning rate of 0.001.
- A learning rate decay scheme was employed, wherein $lr_t = lr_{t-1} \times 0.99$ was initiated after the tenth epoch, with decay continuing until a minimum value of 0.000001 was reached.
- Training was conducted for 800 epochs on an NVIDIA GeForce RTX 3080 GPU. Early stopping was implemented with a patience of 100 epochs.
- One hundred copies of the fan settings vector were utilized.

The selection of an appropriate loss function significantly influences the performance of the model. In this study, the mean absolute percentage error (MAPE) was adopted, with the following conventions: $\frac{0}{0} = 0$ and $\frac{a}{0} = \infty$ for all $a \neq 0$ [39]. Equation (1) demonstrates the calculation of the MAPE loss, where n denotes the number of pixels in the image, p represents the predicted value for a given pixel, and gt signifies the ground truth value.

$$LOSS_{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \frac{(p_i - gt_i)}{gt_i} * 100 \quad (1)$$

Preliminary tests indicated that utilizing MAPE as the loss function yielded significantly improved the performance in comparison to the mean absolute error (MAE) or mean squared error (MSE) for both the training and validation datasets.

To maximize the utilization of all available data, the sequence length was limited to two, considering the duration of the experiments in the dataset as described in Table 1. For instance, an experiment comprising 8 frames contributed a single input-ground truth sequence pair, while 9 frames resulted in 2 pairs, and so forth. To prevent the model from memorizing the order of entries in the dataset, all sequence pairs were randomly shuffled.

2.4. Control Policy Generation Using Genetic Programming

The subsequent component of the pipeline involves the utilization of genetic programming (GP) to generate control policies for the fans. GP is a variant of Genetic Algorithms (GA) developed by John R. Koza, where the solution is encoded in a tree structure instead of a string [40–42]. Similar to GA, GP draws inspiration from nature and mimics the evolutionary process by iteratively applying a set of genetic operations on an initially randomly selected pool of candidate solutions [41,43,44]. However, unlike GA, which aims to solve specific optimization tasks, genetic programming focuses on creating a model with a predefined objective [45].

In this study, the controller population is designed with two primary objectives. Firstly, it aims to adjust the fans to achieve a homogeneous temperature field. Secondly, it strives to prevent the occurrence of local hot spots. Evaluating these phenomena can be challenging, and relying solely on a single metric may be insufficient. To address this issue, we propose a combination of three metrics to assess the performance of the controller. The first metric targets the homogeneity of the temperature field by minimizing the standard deviation of the pixel values. A lower standard deviation indicates a more uniform temperature distribution. However, relying solely on this metric is inadequate for effectively penalizing hot spot formation. Hence, we introduce a second loss, referred to as the hot spot loss, which calculates the sum of all positive pixel values after subtracting the mean temperature from each pixel. This loss function encourages strong cooling and discourages the formation of regions with temperatures significantly higher than the system's average temperature. Additionally, we incorporate an auxiliary loss function to penalize excessive fan usage:

$$\begin{aligned} Loss_{STD} &= \sqrt{\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}} \\ Loss_{hotspot} &= \sum_{i=1}^{m} x_i - \mu \\ Loss_{fanload} &= \frac{1}{300}\sum_{i=1}^{3} f_i \end{aligned} \tag{2}$$

where μ represents the mean value, n corresponds to the total number of pixels, and m corresponds to the number of pixels with values larger than μ. To ensure an appropriate evaluation, we scale these three losses to the same order of magnitude and assign weights to emphasize their relative importance. The assigned weights are 5, 5, and 1, respectively. This weight distribution ensures that the fan load loss only becomes relevant when different control laws produce similar temperature distributions.

Control Model Architecture

The integration of the controller into the pipeline requires a trained next-sequence predictor. As explained in Section 2.3.1, the predictive model is compiled as two parts, separated at the output of the initial dense layers. This separation offers a significant advantage: it allows the entire $2 \times 512 \times 640 \times 1$ input sequence to be compressed into a vector consisting of only 2048 data points. This compressed vector is used as the input for the GP-based controllers. By employing this compression technique, the entire input space

can be spanned by deeper trees, enabling the generation of solutions based on the complete temperature field, rather than randomly selected local regions of interest.

Control laws in the form of a 3D vector are generated by each candidate in the population (Figure A1). To align with the expected input dimensions of the second part of the predictor, the vector is duplicated 100 times. Next, the proposed fan settings vector is appended, and predictions are generated using the decoder component of the digital twin model. These predictions are then evaluated against a predefined fitness function, and the corresponding fitness scores are assigned to the respective individuals (Figure A1).

The GP controller undergoes evolutionary training for 5 generations on each training experiment, amounting to a total of 1410 generations. Limiting the training to only 5 generations per sequence prevents overfitting to a specific problem, allowing for the transmission of genes that exhibit generalization capabilities across various heating loads. This can be considered similar to the early stopping policies in NN training. For additional information on the GP controller's configuration and the reasoning behind the chosen approaches, refer to Appendix B. Appendix C further presents the details of the MPC experiment design for a population of control laws.

3. Results

3.1. Testing Digital Twin as a Predictive Model

Before implementing with the GP-based controller on the real experimental setup, the performance of the digital twin is first assessed in two distinct aspects. First, it should be able to accurately predict the next two frames given a certain set of inputs. Second, it should be able to capture the impact of the fan loads on the temperature distribution within a virtual experiment, even if it is not part of its training set. In other words, the learnt model representation of the problem in NN parameters should be able to answer "what if" questions in a reasonable way.

Figures 5 and 6 illustrate some good and bad predictions of the digital twin model on new test experiments with pre-set heat load changes and fan settings. It is worth note here that the digital twin typically performed well for experiment trajectories with around 10 snapshots, while failing to capture the extreme hot spots in very short experiments, which were underrepresented in the training set (see Table 1). For instance, the first experiment in Figure 6 consists of only one executable sequence. As a result, the model never received information regarding the new heat load on the system. Consequently, the prediction is an informed guess, based on the last steady state reached. Similar behavior can be observed in the first predictions for experiments (a) and (b) in Figure 5. Hence, weaker performance is to be expected in such cases. This indicates that input sequences containing only the frames, depicting the steady state reached from the previous experiment, may have a negative influence on the model's predictive capabilities on hot spot risk estimation. Fortunately, we do not parse two consecutive steady-state frames as input to the controller, thus mitigating the effect of such outliers when we evaluate the MPC performance. The reason for the inaccurate predictions for the second experiment in Figure 6 is not clearly identifiable. While it manages to capture the structural evolution of the temperature field, it misses the hot spot formation. One reasonable explanation for this is the effect of sampling through a sparsely populated set of fan settings. Increasing the number of training and validation examples and sampling from a set with smaller intervals may remedy this behavior. In either case, however, the MAPE score was less than 5%, which would still be relatively informative enough to decide upon the best MPC policy given the input sequence.

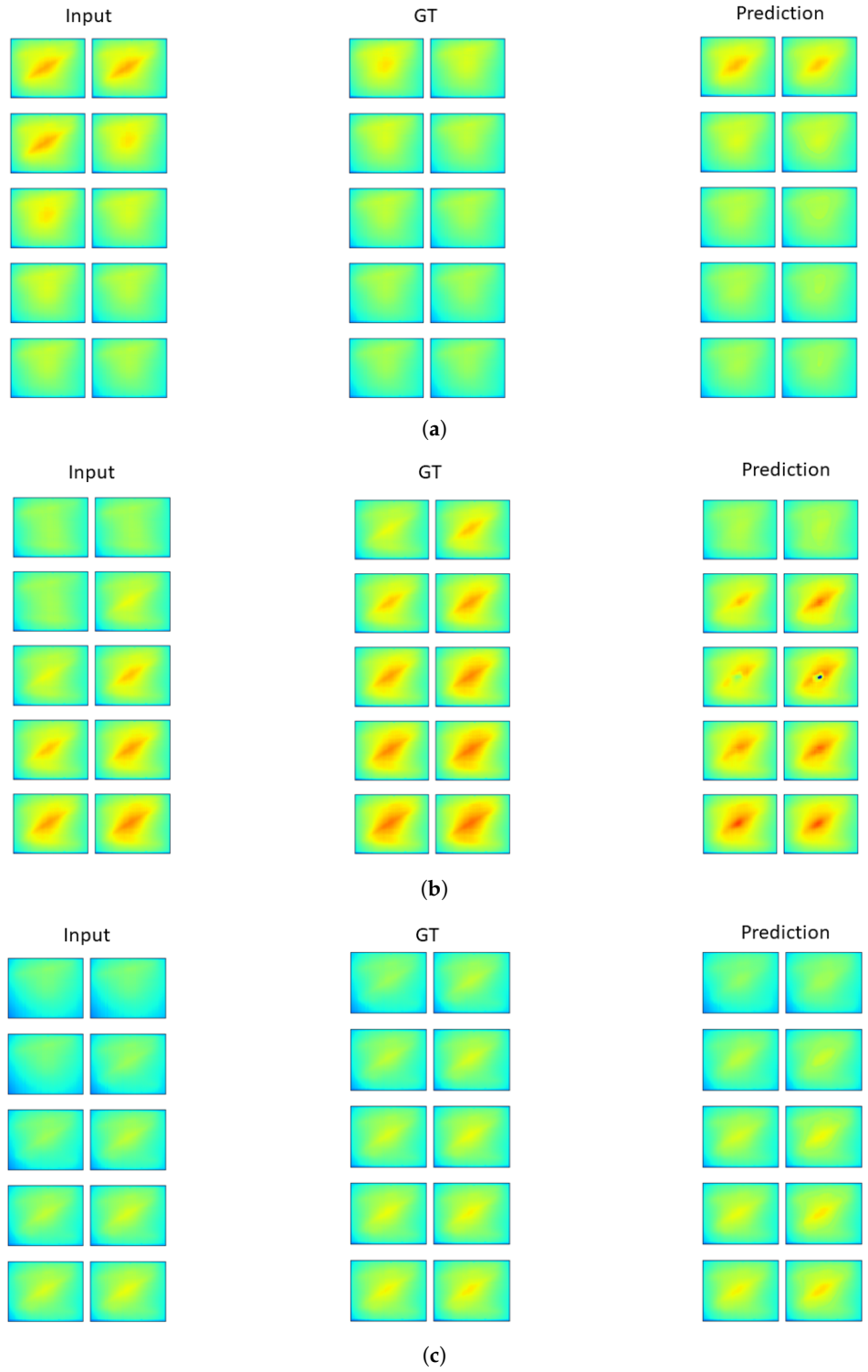


Figure 5. Examples of successful digital twin predictions. GT refers to the ground truth (i.e., experiments). (**a**) Fans [0, 40, 60]; Heating Strips [75, 0, 25]. (**b**) Fans [0, 40, 40]; Heating Strips [75, 100, 100]. (**c**) Fans [60, 80, 100]; Heating Strips [75, 75, 100].

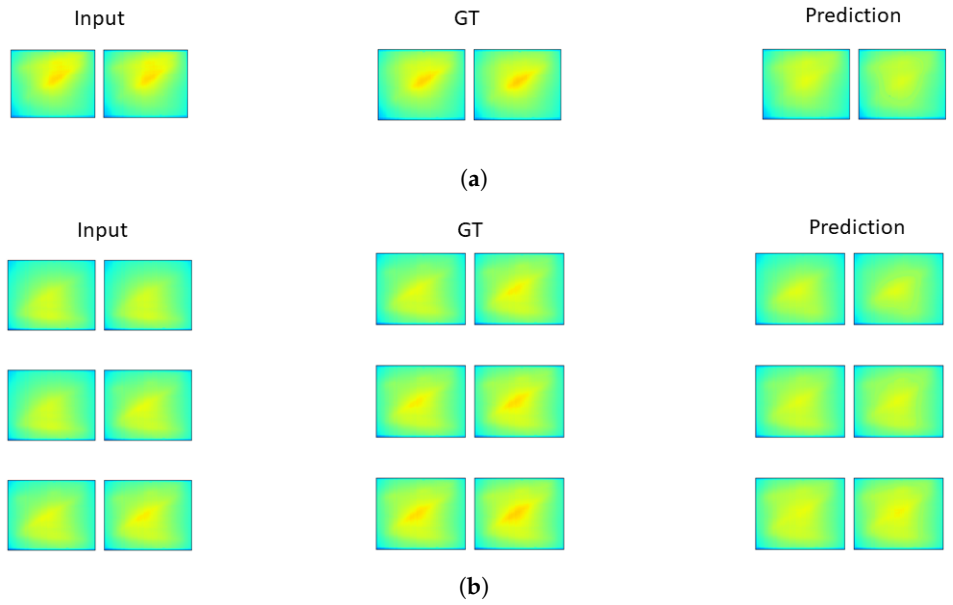


Figure 6. Digital twin predictions missing the hot spot formations. GT refers to the ground truth (i.e., experiments). (**a**) Fans [60, 20, 20]; Heating Strips [25, 100, 25]. (**b**) Fans [80, 0, 40]; Heating Strips [100, 75, 75].

The second assessment for the digital twin is related to its ability to capture the physical relationship between the fan settings and the evolution of the temperature field, as "understanding" the fans' impact is crucial for the performance of the controller. For that purpose, we conducted a set of parametric analysis. Given a sequence of inputs, the digital twin first makes a prediction of the next one minute for a given fan setting (e.g., [0%, 40%, 60%]), for which the ground truth measurements exist. After checking model accuracy (MAPE < 2%), the DT is used to estimate how the temperature field would be if the fans were fully open ([100%, 100%, 100%]), or fully closed ([0%, 0%, 0 %]). Some examples of the DT predictions are shown in Figure 7. While it is difficult to judge the extent to which the model perceives the impact of cooling on the temperature field distribution, one may conclude that it adequately shifts the prediction with changing fan loads. If they are fully opened, there is an increased cooling effect, while turning the fans off leads to the emergence of some hot spots.

It is worth noting here that changing the way the fan settings are parsed to the model can further improve its ability to capture the effect of the fan loads on the temperature distribution. In the current architecture, we clone the fan settings vector 100 times to increase its relative importance. Although this strategy achieves satisfactory results, it may not be the optimum approach. An alternative way would be to append each of the three fan loads as a channel to the input images. In this way, we would allow the encoder to "learn" the impact the fan settings have on the temperature field evolution. For the current implementation, however, it is considered to be unnecessary as the model already performs with high accuracy (mean absolute percentage error, Equation (1)).

Figure 7. Capabilities of the digital twin model for extrapolation of the fan settings. (**a**) Prediction: Fans [0, 40, 60]; Fans on [100, 100, 100]. (**b**) Prediction: Fans [0, 40, 40]; Fans on [100, 100, 100]. (**c**) Prediction: Fans [60, 20, 20]; Fans off [0, 0, 0]. (**d**) Prediction: Fans [80, 0, 40]; Fans off [0, 0, 0].

3.2. Model Predictive Controller Performance

After validating the accuracy of the digital twin model, the performance of the control law population is investigated via following the experimental protocol described in Appendices B and C. The metrics used to evaluate the controller's performance are the same as the loss functions defined in Equation (2).

The greatest advantage of the intelligent controller is that it can leverage the speed of the NN-based predictive model to select the best control law policy among the alternatives for the current state trajectory in real time. Figure 8 portrays the change in the temperature field caused by a significant and sudden increase in the thermal load applied to the system, while Figure 9 shows the temporal evolution of performance metrics. In all GP-based tests, a control law population of 10 is deployed. In both figures, the term "Specialist" denotes a subset of controls particularly trained to handle high load disturbances. "General" refers to control laws learned for the entire operating range. The category labeled as "Random" corresponds to the benchmark case, where fan settings are randomly assigned (for further details, refer to Appendix C). It is clearly observed in Figure 9 that both GP controllers (Specialist, General) significantly outperform the random controller. Interestingly, the General population achieves a reasonably similar performance to the Specialist at significantly lower energy consumption.

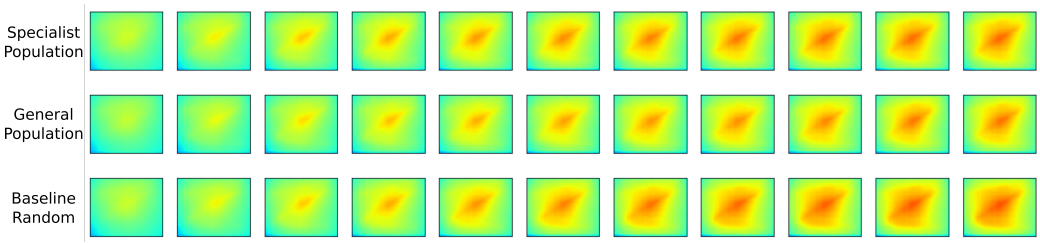


Figure 8. From left to right—Evolution of the temperature field during the experiment: Specialist (**top** row), General (**middle** row) and Random Control (**bottom** row).

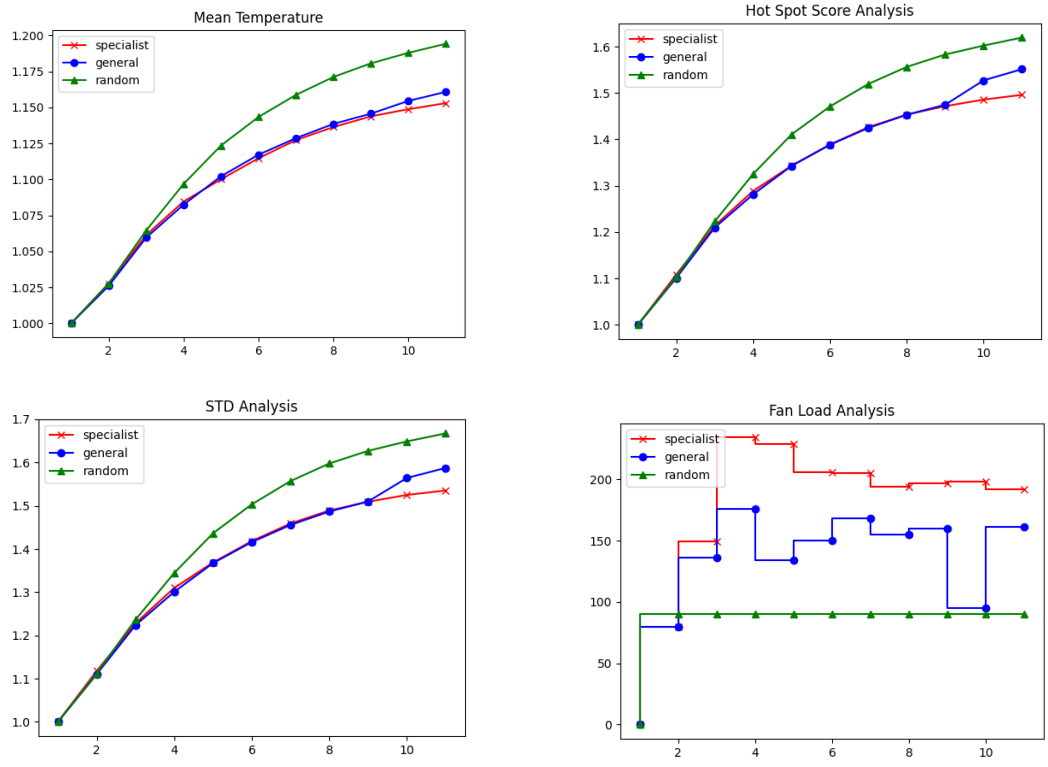


Figure 9. Performance metrics of MPC with 10 individuals at high thermal loads. The x-axis denotes the time, while the y-axis shows the metric.

When the the heat load is raised to medium range from low loads (Figure 10), the Specialist population was found to outperform both the General control law population and the Random cooling, despite the fact that the fan load of the Random case is much higher (Figure 11). If just the fan settings are considered, Specialist MPC is at a disadvantage to eliminate the hot spots on the surface, compared to the random controller. Yet it was found

in repeated experiments that Specialist population-based control outperforms the others with less power usage for the fans. The performances of the GP-based controller were also tested in the settings where the heat load is reduced (Figures 12 and 13). As expected, the Specialist population is the best performer. However, its energy consumption is also higher. A possible explanation is that since a homogeneous temperature field is currently prioritized over efficiency, individuals that perform better on the hot-spot and standard deviation metrics are overtaken, albeit at a higher energetic cost.

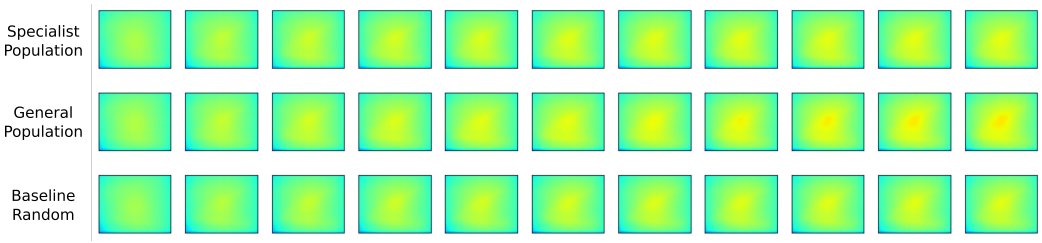


Figure 10. From left to right—Evolution of the temperature field during the experiment: Specialist (**top** row), General (**middle** row) and Random Control (**bottom** row).

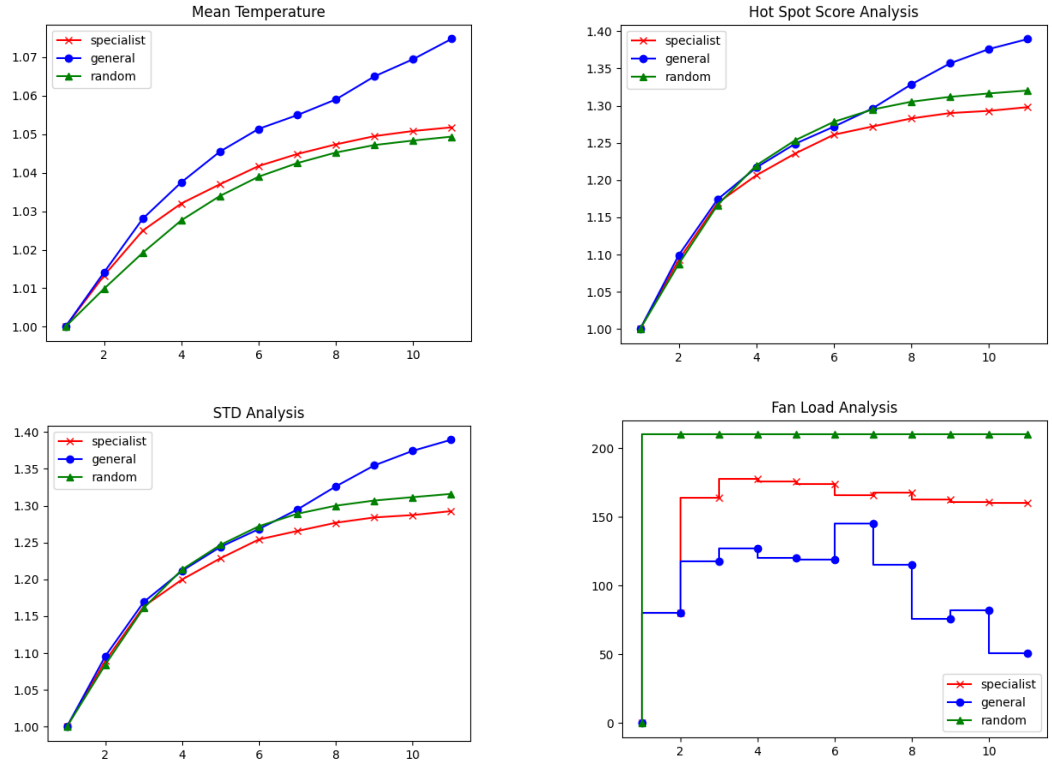


Figure 11. Performance metrics of MPC with 10 individuals at medium thermal loads. The x-axis denotes the time, while the y-axis shows the metric.

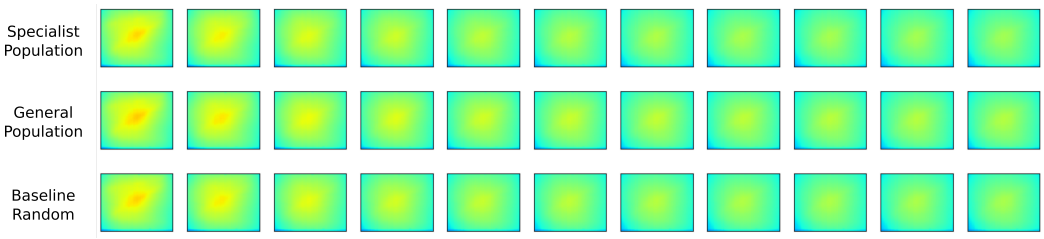


Figure 12. From left to right—Evolution of the temperature field during the experiment: Specialist (**top** row), General (**middle** row) and Random Control (**bottom** row).

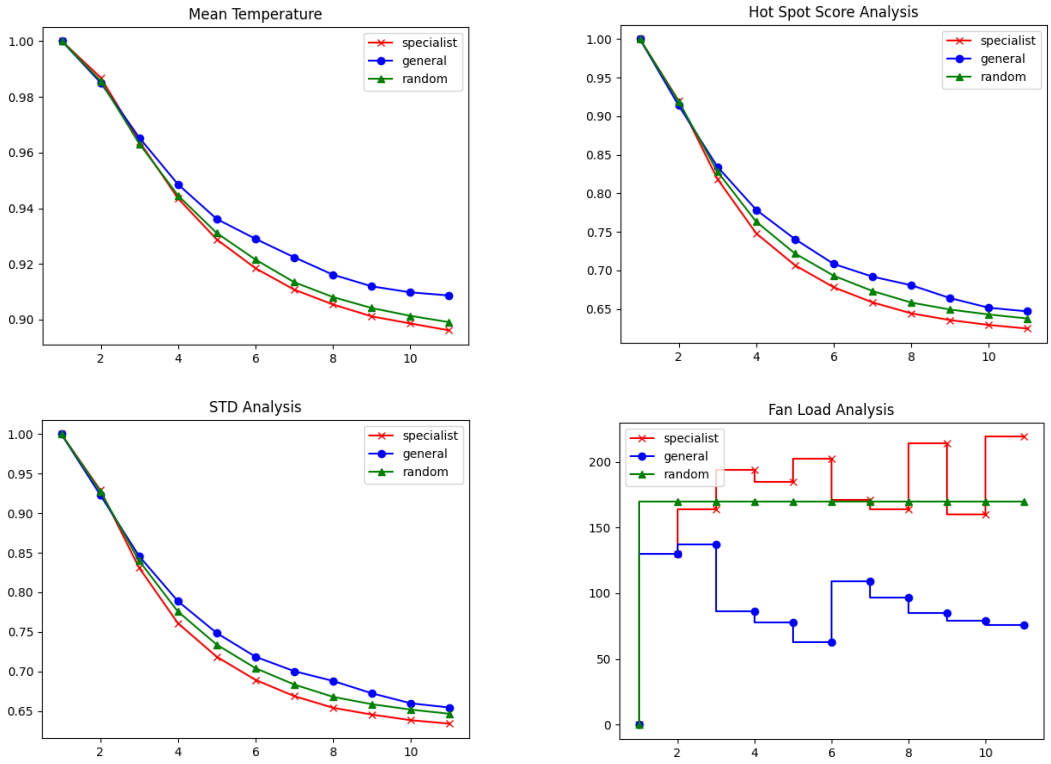


Figure 13. Performance metrics of MPC with 10 individuals at low thermal loads. The x-axis denotes the time, while the y-axis shows the metric.

4. Discussions

Many critical processes of a technical nature occur at high temperatures, leading to the heating of structurally and functionally important components. This heat can significantly deteriorate their properties, especially when coupled with an uneven distribution of temperatures that creates local stresses and deformations. Such processes are often characterized by nonlinearity and stochasticity, making analytical modeling challenging. Fortunately, recent advances in machine learning have provided new opportunities for modeling dynamic systems, even in the absence of precise mathematical descriptions. Consequently, it has become feasible to design controllers that exhibit robust performance and fast response times, even for systems that are stochastic and nonlinear in nature. The objective of this work is to establish a population-based model predictive controller, which tests alternative cooling policies via a virtually trained digital twin on a generic multi-mode heat transfer test rig. The practical aim is to minimize the hot spot formations on the surface, while simultaneously minimizing the overall surface temperature. In accordance, the controlled variables are taken from IR camera measurements, which creates an extremely large input space with more than half a million dimensions. Furthermore, the sudden changes in the heat load distributions on the surface leads to complex, nonlinear transient heat transfer processes, resulting in a significant variation in the time and length scales in the thermal state. In accordance, the controller should be complex enough to respond the drifts in both the system state and the measured variable characteristics. In this work, we propose to use a population of control models within an MPC scheme to respond to these demands. Moreover, the control models in the population were not assumed a priori, but rather learnt via an evolutionary algorithm on measured data. The same training database of experiments were also used to create a digital twin of the process, with which virtual control experiments can be conducted to speed up the evolutionary process. For the studied problem type of image sequence prediction, ConvLSTM-based autoencoder enabled the extraction of a latent representation of the past and current state by using IR camera measurements. More importantly, when fan settings are appended to the vector

representation in the latent space, the autoencoder was shown to learn and interpret the impact of fan settings on the future state trajectories, which is of critical importance for a dynamic MPC problem. The robustness of the population-based controller is one of the key properties of the proposed digital-twin-assisted MPC pipeline. In order to demonstrate its added value, the same high, medium and low heat load tests were also conducted by picking one individual control model from the converged population pool, instead of 10 for the Specialist and the General sub-groups (Appendix D). While selecting one individual from the gene pool led to a better control when the heat load was suddenly decreased to medium and low load range, it resulted in a worse performance in hot spot formation when the load suddenly increased from a low to high range. It should be pointed out at this point that 323 experiments were conducted in the study to create a train/validation dataset, and the whole MPC pipeline was tested on randomly generated disturbances out of 27,000 possible configurations (data density was 1.2%). As a result of this sparsity, it is likely that the state dynamics may not be captured with a single control law, particularly if both the DT and the controller model is learnt from data. However, deploying an ensemble of controller models with a DT enables the testing of alternative control policies virtually and deploys the best approach. Furthermore, with an evolutionary approach, it is also possible to trigger the creation of new offspring models, if the current population starts to fail in suppressing the hot spot formations. In the current work, we only deployed 10 of the best individuals from the whole gene pool around 300 converged solutions, based on their performance on a small subset of the state space (<1%). Although the performance of 10 individuals was better than the benchmark case, utilizing the whole population within MPC would lead to much better performance. In MPC experiments, the time interval to make a decision after testing the controller models was set to be less than 30 s. In the current code implementation, the tree model compiling of the GP model was run in a serial mode, hence it limited the application to a maximum of 10 individuals. Therefore, it is strongly recommended to parallelize the controller testing for a more robust implementation. The task of speeding up the candidate evaluation problem and sampling from a larger pool of candidates remains open for future work.

Enhancing the accuracy of the predictive model is a paramount objective for future contributors. Expanding the size of the training and validation datasets is imperative to comprehensively evaluate the architecture's potential. Moreover, refining the data-cleaning process and incorporating experiments with longer durations could booster the model's performance and reliability. Tailoring the pipeline to the specific requirements of the problem at hand is another crucial aspect to consider. For instance, another potential improvement is to extend the length of the sequence for the ConvLSTM autoencoder to fully take advantage of the long-term memory capacity of the model, particularly if the proposed methodology is applied to a different problem. Additionally, we investigate the integration of fan settings as channels within the input images and explore the utilization of symmetric skip connections.

Overall, the results strongly suggest that taking advantage of the ability to test multiple control laws in real-time leads to a significant improvement in the controller's performance. The results clearly indicate that DT-assisted MPC produces effective and efficient control laws even with sparse training data. The fact that the specialist populations consistently outperform random controllers, highlights the potential for the application to more sophisticated problems.

5. Conclusions

This study highlights the significant potential that emerges from combining a population-based control strategy with neural networks to construct a robust and dynamic *Model Predictive Control* framework suitable for addressing complex and nonlinear challenges. The effectiveness of our approach is demonstrated through extensive real-time experiments conducted within a multi-mode heat transfer scenario, where the measured variable vector encompasses high-dimensional infrared camera measurements organized

as a sequence (655,360 inputs). We utilize evolutionary algorithms to generate a diverse set of control laws from empirical data, allowing for adaptability to complex and transient heat transfer dynamics. Importantly, our digital twin-enhanced population-based MPC outperforms individual control models, particularly in scenarios involving sudden and stochastic shifts in localized thermal loads. The digital twin, engineered through ConvLSTM-based spatiotemporal pattern extraction, assumes a pivotal role in virtually testing alternative control policies, thereby substantially heightening the controller's responsiveness, even when confronted with limited data availability. Differentiating from traditional methods constrained by the nonlinear and stochastic aspects of complex systems, our data-driven approach harmonizes the capabilities of neural networks, genetic programming and digital twin technology. This blend not only demonstrates the practical efficacy of our contribution, but also highlights the broader potential of these methods across various domains.

Author Contributions: Conceptualization, C.A. and D.B.; methodology, C.A., D.B., J.A. and R.Y.; software, C.A., D.B. and R.Y.; formal analysis, C.A., D.B., J.A. and R.Y.; writing—original draft preparation, C.A., D.B. and R.Y.; writing—review and editing, J.A., R.K. and H.-J.B.; visualization, C.A., D.B. and R.Y.; supervision, C.A., R.K. and H.-J.B.; project administration, C.A. and H.-J.B.; funding acquisition, C.A. and H.-J.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data will be available upon request. The code can be accessed via https://github.com/cihan-ates/model_predictive_control (accessed on 1 August 2023).

Acknowledgments: We acknowledge support by the KIT-Publication Fund of the Karlsruhe Institute of Technology. The authors also thanks Patrick Zengerle and particularly Michael Lahm from ITS Workshop for their support in building the experimental test rig.

Conflicts of Interest: The authors declare no conflict 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.

Nomenclature

The following abbreviations are used in this manuscript:

ARIMA	Autoregressive moving average model
CNN	Convolutional neural network
ConvLSTM	Convolutional Long Short-Term Memory
DT	Digital twin
FSP	Fixed set-point
GA	Genetic algorithm
GP	Genetic programming
HDF	Hierarchical data format
HVAC	Heating, ventilation and air conditioning
LIDAR	Light detection and ranging
MAE	Mean absolute error
MAPE	Mean absolute percentage error
MSE	Mean squared error
MPC	Model predictive control
NARIMAX	Nonlinear Autoregressive moving average model with exogenous inputs
NN	Neural networks
PID	Proportional–integral–derivative controller
PSO	Particle swarm optimization
RNN	Recurrent neural network
ReLU	Rectified linear unit
STD	Standard deviation

Appendix A. Background of the Deployed Digital Twin Model

While data-driven methods have an impressive potential for application in the field of digital twin creation, it is important to note that the architecture and performance are heavily dependent on the nature of the problem to be solved. Thus, a thorough understanding of the system and the underlying fundamental physical laws could contribute to a more precise problem formulation. In turn, this can facilitate the selection of a more adequate architecture for the approximation of the system. The problem that the model used in this paper is going to solve falls within the subcategory of the image sequence prediction.

Given that the system's state is represented by infrared (IR) camera images, the predictive task undertaken by the digital twin becomes a challenging task of estimating conditioned image sequences. This entails the need to capture both the spatial structures within the images and the temporal relationships between consecutive frames. When working with image data, CNNs are widely regarded as the preferred choice due to their strong performance and efficiency. Conversely, RNNs have demonstrated success in handling time-series data. Thus, a combination of CNN and RNN architectures is necessary to address the image sequence prediction problem effectively. In recent years, architectures incorporating the *Convolutional Long Short-Term Memory* (ConvLSTM) module have emerged as successful solutions for such tasks [6,46,47]. The ConvLSTM memory cell has a very similar structure to the standard LSTM. However, the fully connected matrix multiplications are replaced by convolutional operators [47]. This simple modification has two significant implications. First, it reduces the redundancy in the model. Second, by setting the convolutional kernel to a value larger than one, one can capture complex "spatiotemporal motion patterns" [6]. An interesting point to highlight here is the robustness of LSTM-based temporal modeling. For instance, in a recent work, the concept has been further extended to a reversed sequence-to-sequence mapping technique that is applicable for long time-horizon forecasting in dynamical systems [48]. The applicability of the approach was also shown to model spiking (biological) pyramidal neurons in hippocampal CA1 [48].

The typical ConvLSTM architecture resembles an Encoder-Decoder architecture. In the original implementation of this architecture, Shi et al. [6] try out several models with varying depths and widths. This approach consistently outperforms the fully connected LSTM. In another example, ref. [49] construct a next-frame prediction model adopting ConvLSTM. The encoder extracts high-level features and encodes them into a fixed-size vector, while the decoder reads the vector and transforms it into the prediction for the next frame's state [49].

Considering the practical importance of multivariate time series prediction, several improvements to the original ConvLSTM architecture have been proposed. Ref. [50] demonstrated that symmetric skip connections between the encoder and decoder parts of the model can significantly improve its image restoration capabilities. Others try to combine ConvLSTM cells with conventional convolutional modules. For instance, ref. [51] applied a ConvLSTM network to the fully compressed feature map of a five-layer convolutional encoder to predict subsurface flows. This allows them to extract rich features through the convolutional encoder alongside the long-term temporal evolution of the flow with a relatively compact model. Alternatively, ref. [52] applied a standard 2D convolution in parallel with a ConvLSTM layer. In this way, he preserves the original ConvLSTM implementation where the input dimensions remain constant, while simultaneously compressing the inputs through a standard convolutional encoder. This architecture allows the addition of more layers, and thus extracts more features, without a dramatic increase in the number of total parameters. As a result, the model can generalize better and process longer sequences. Ref. [53] adopted a similar approach, however, they argued that separating the convolutional autoencoder from the ConvLSTM network may further increase the network's performance. Furthermore, they proposed an improved training protocol. The autoencoder was first trained independently. Consequently, latent space representations are used for the training of the ConvLSTM network. As a last step, the entire network was trained together

for fine-tuning [53]. Finally, ref. [47] proposed an inception-inspired ConvLSTM, where each convolution was implemented with a different kernel size, thus extracting features at different scales. Overall, the previous work indicates that ConvLSTM-based models can achieve good results for image sequence prediction. Furthermore, the architecture can be optimized according to the task at hand.

In this work, a relatively simpler design approach was considered. The image space of the case of interest was found to be relatively homogeneous, and the duration of the experiments ranged between three and five minutes. As a result, the main objectives could be identified as follows: (i) accurate predictions of the next frames, (ii) low computational cost to be useful to the controller, and (iii) avoidance of information loss during image reconstruction. Consequently, a ConvLSTM-based approach was deemed sufficient due to its adequate performance, flexibility, and straightforward implementation. Details of the deployed architecture are provided in the next section.

Appendix B. GP Controller Hyperparameters and Operators

Table A1. Hyperparameters of the deployed genetic programming approach.

Parameter/Operator	Value/Policy	Argument
Mutation Probability	0.05	To prevent the loss of good solutions while maintaining diversity in the gene pool.
Crossover Probability	0.85	To avoid unnecessary population shrinkage and prevent excessively fast convergence.
Tree Depth	15–25	Shallow trees would only utilize a small portion of the inputs and would be insufficient for generating sophisticated control laws. Deeper trees, however, require longer computational times for evaluation.
Selection Strategy	Tournament selection	This strategy is widely used and has shown acceptable results. According to [41], all selection strategies can generate satisfactory outcomes, except for roulette, which is not suitable for minimization tasks.
Tournament Size	2	A smaller tournament size preserves greater variety in the gene pool.
Population Size	300	A larger initial population ensures a more diverse gene pool. However, it also leads to longer training times. To capitalize on the processing power of our GPU unit, we explore a broader set of initial candidates.
Output Filter	Sigmoid	The outputs of the trees are scaled to values between 0 and 1 using the sigmoid function.

Furthermore, we selected the following mathematical operations for the nodes of the trees:

- Linear operations—summation, addition, subtraction, multiplication and negation;
- Trigonometric operations—sine and cosine—these operators are used to scale the floating point numbers in the tree. This prevents an "explosion" of the values in either direction (positive or negative), resulting in only two possible modes of operation for the fans—either 0% or 100% load;
- Regrouping operations—create a 3D vector from three values—this is a hard-coded function for the output of the tree, which should result in a 3D vector with one value for the duty cycle of each fan.

Appendix C. MPC Experiment Design

The next step is to transfer the controller from the virtual to the physical domain and assess its performance on the experimental setup. Figure A2 depicts the final procedure for the MPC experiments with one, or multiple control models.

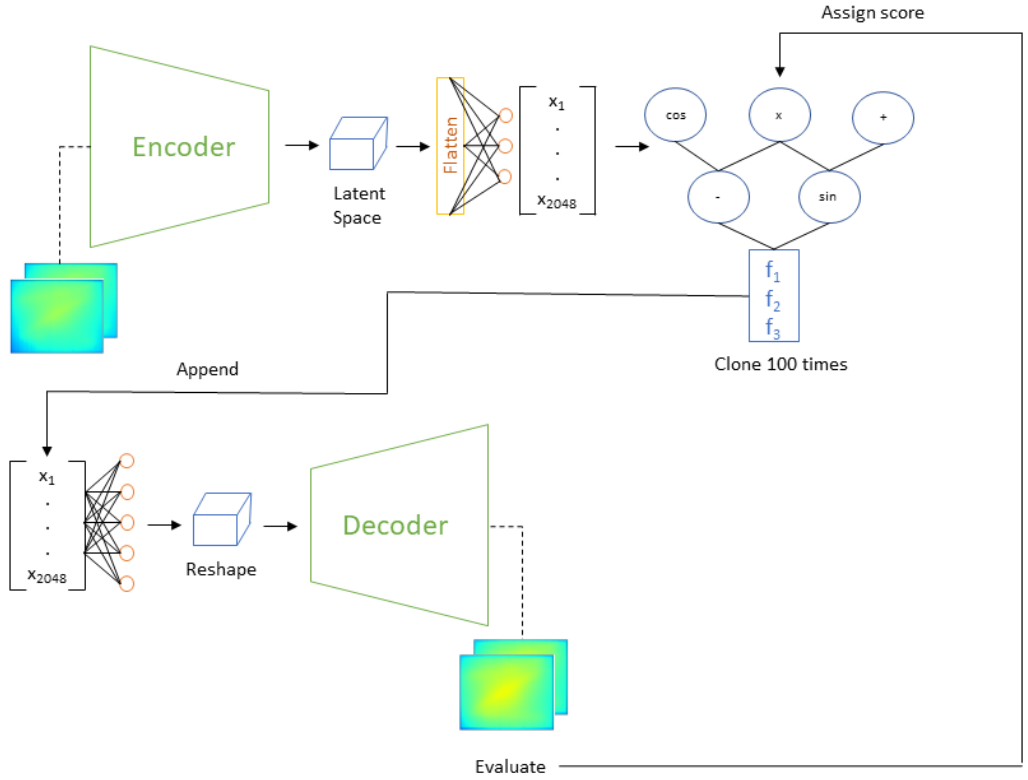
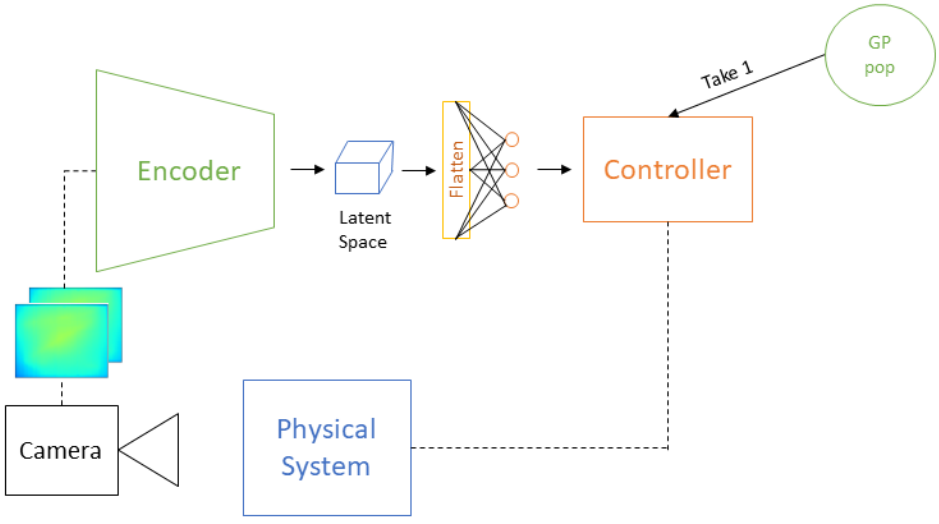
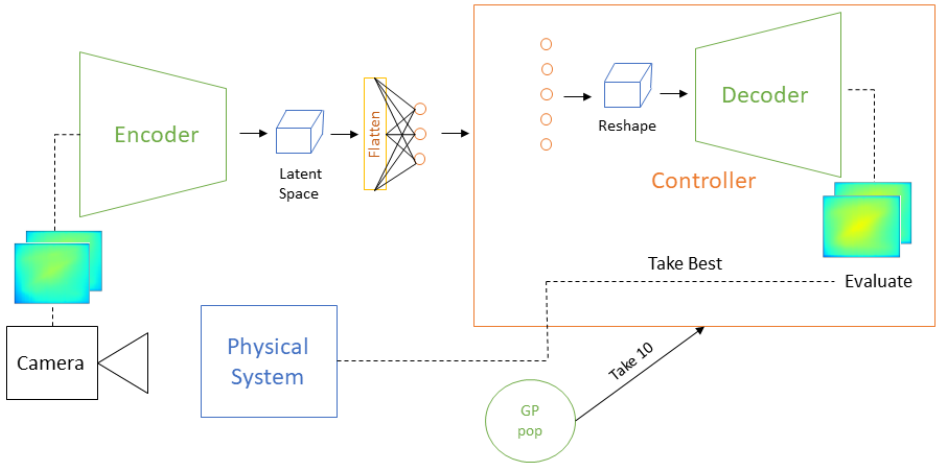


Figure A1. The training pipeline for the GP controller.



(**a**)

Figure A2. *Cont.*



(**b**)

Figure A2. Experimental protocol for MPC. (**a**) Experiment pipeline with a single individual. (**b**) Experiment pipeline with population-based MPC.

MPC tests were conducted using three distinct control policies: (i) specialist control models based on heat loading, (ii) general-purpose control models, and (iii) a simple control model utilized as a benchmark. The specialist groups were formed by selecting the top 10 performers from the final population in experiments with low (total load < 100), medium (100 < total load < 200), and high heat load conditions (200 < total load < 300). As a result, three specialist populations were created, each corresponding to one of the heating load groups. The general group consisted of randomly chosen individuals from the final population.

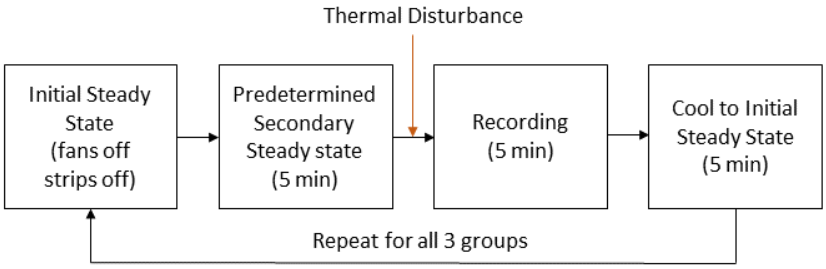


Figure A3. Standardized protocol for the Performance Evaluation Experiments.

To ensure a proper evaluation of the controller's performance, it is essential to maintain comparability among experiments within each group. To achieve this, a standardized workflow is followed, as depicted in Figure A3. The workflow includes the following steps:

1. Cooling to the initial state: All experiments begin from the same starting point by cooling the system to the initial state. This step ensures consistency across experiments.
2. Recreating a predetermined steady state: To simulate the control of a dynamic system and replicate a realistic scenario, the system is preheated to a predetermined secondary steady state. This step further enhances the reliability of the evaluation.
3. Fixed experiment duration: Each experiment is conducted for a fixed duration of 5 min, with a frame captured every 30 s. This extended monitoring period allows for a comprehensive observation of the evolution of the temperature field.

In the MPC tests, three different thermal load scenarios are investigated:

- High heat loading: the load on the heating strips was suddenly increased from [50%, 25% and 0%] to [75%, 100% and 75%], while the fans were open at [20%, 40% and 20%]. The benchmark control law resulted in a fan setting for the cooling experiment of [70%, 0% and 20%] after the set point change.

- Medium heat loading: the heating strip loads were suddenly raised from [25%, 0% and 50%] to [25%, 50% and 70%], while the fans were open at [30%, 20% and 30%] during the second steady state. In this situation, the benchmark control law adjusted the fan settings to [30%, 80% and 100%].
- Low heat loading: the thermal load was abruptly reduced from [75%, 75% and 50%] to [0%, 25% and 25%], while the fans were open at [50%, 80% and 0%]. In this case, the benchmark controller set the fan settings to [80%, 50% and 40%].

Finally, the number of candidates to be evaluated in real-time before applying the control laws needs to be determined. Given our objective of achieving quick response times, it is crucial to strike a balance between evaluation accuracy and computational efficiency. To address this, we employ two different strategies, as illustrated in Figure A2. In the first strategy, a single individual is evaluated. For the specialist populations, the best individual is selected, while for the general populations, a single individual is randomly chosen. This approach ensures a focused evaluation while minimizing computational overhead. In the second strategy, ten individuals are selected for real-time evaluation. Similar to the first strategy, individuals are randomly chosen from the general populations. However, for the specialist populations, the entire population is included in the evaluation. This expanded evaluation allows for a more comprehensive assessment of the control laws. Regardless of the chosen strategy, the randomly generated fan settings remain constant throughout the entire duration of the control experiment. This ensures consistency and eliminates any potential bias introduced by varying fan settings.

Appendix D. Single Individual Tests

Appendix D.1. High Load Test Case

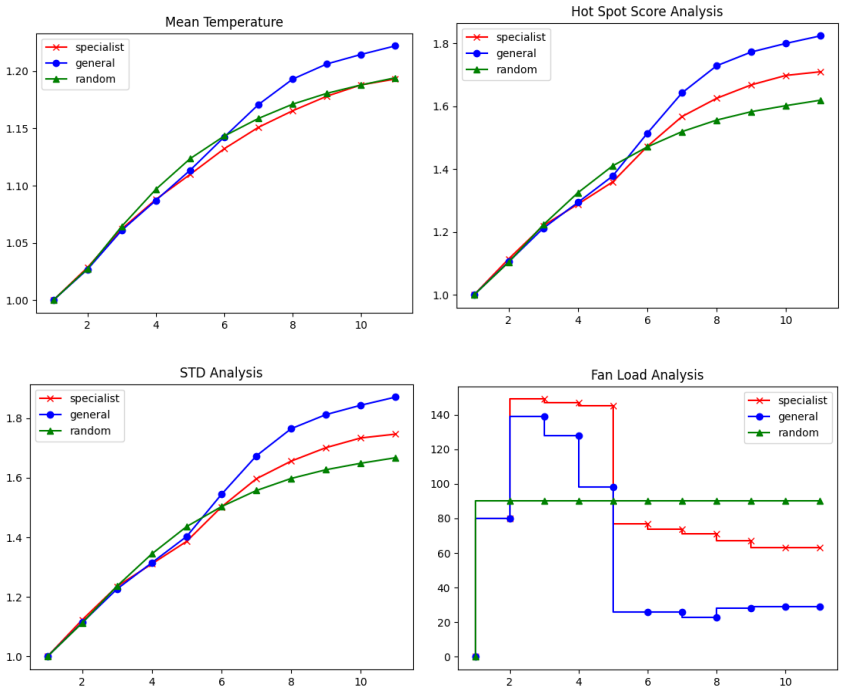


Figure A4. Performance metrics of MPC with 1 individual at high thermal loads. The x-axis denotes the time, while the y-axis shows the metric.

Appendix D.2. Medium Load Test Case

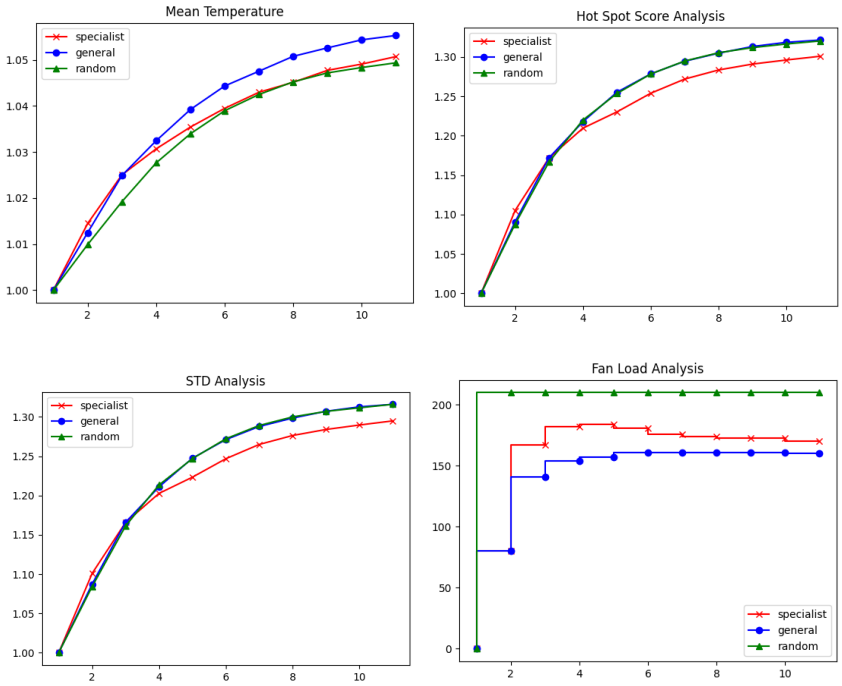


Figure A5. Performance metrics of MPC with 1 individual at medium thermal loads. The x-axis denotes the time, while the y-axis shows the metric.

Appendix D.3. Low Load Test Case

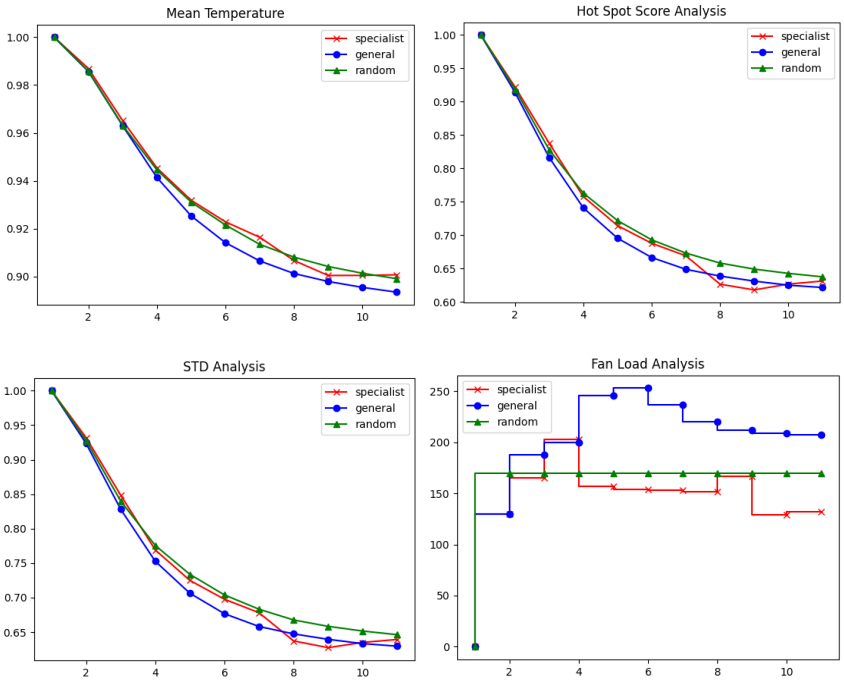


Figure A6. Performance metrics of MPC with 1 individual at low thermal loads. The x-axis denotes the time, while the y-axis shows the metric.

References

1. Marusak, P.M. Numerically Efficient Fuzzy MPC Algorithm with Advanced Generation of Prediction—Application to a Chemical Reactor. *Algorithms* **2020**, *13*, 143. [CrossRef]
2. Nebeluk, R.; Ławryńczuk, M. Tuning of Multivariable Model Predictive Control for Industrial Tasks. *Algorithms* **2021**, *14*, 10. [CrossRef]
3. Domański, P.D. Performance Assessment of Predictive Control—A Survey. *Algorithms* **2020**, *13*, 97. . a13040097. [CrossRef]
4. Wright, L.; Davidson, S. How to tell the difference between a model and a digital twin. *Adv. Model. Simul. Eng. Sci.* **2020**, *7*, 13. [CrossRef]
5. Sun, C.; Shi, V.G. PhysiNet: A combination of physics-based model and neural network model for digital twins. *Int. J. Intell. Syst.* **2022**, *37*, 5443–5456. [CrossRef]
6. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.Y.; Wong, W.K.; Woo, W.C. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In Proceedings of the Advances in Neural Information Processing Systems 28 (NIPS 2015), Montreal, QC, Canada, 7–12 December 2015; pp. 1–9 .
7. Liu, C.; Atkeson, C.G. Standing balance control using a trajectory library. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 10–15 October 2009; pp. 3031–3036. [CrossRef]
8. Koller, T.; Berkenkamp, F.; Turchetta, M.; Krause, A. Learning-Based Model Predictive Control for Safe Exploration. In Proceedings of the 2018 IEEE Conference on Decision and Control (CDC), Miami Beach, FL, USA, 17–19 December 2018; pp. 6059–6066. [CrossRef]
9. Tavakoli, M.; Shokridehaki, F.; Marzband, M.; Godina, R.; Pouresmaeil, E. A two stage hierarchical control approach for the optimal energy management in commercial building microgrids based on local wind power and PEVs. *Sustain. Cities Soc.* **2018**, *41*, 332–340. [CrossRef]
10. Maddalena, E.T.; Müller, S.A.; dos Santos, R.M.; Salzmann, C.; Jones, C.N. Experimental data-driven model predictive control of a hospital HVAC system during regular use. *Energy Build.* **2022**, *271*, 112316. [CrossRef]
11. McKinnon, C.D.; Schoellig, A.P. Learn Fast, Forget Slow: Safe Predictive Learning Control for Systems with Unknown and Changing Dynamics Performing Repetitive Tasks. *IEEE Robot. Autom. Lett.* **2019**, *4*, 2180–2187. [CrossRef]
12. Wang, H.; Chen, Y.; Kang, J.; Ding, Z.; Zhu, H. An XGBoost-Based predictive control strategy for HVAC systems in providing day-ahead demand response. *Build. Environ.* **2023**, *238*, 110350. [CrossRef]
13. Ay, M.; Stemmler, S.; Schwenzer, M.; Abel, D.; Bergs, T. Model Predictive Control in Milling based on Support Vector Machines. *IFAC-PapersOnLine* **2019**, *52*, 1797–1802.
14. Piche, S.; Sayyar-Rodsari, B.; Johnson, D.; Gerules, M. Nonlinear model predictive control using neural networks. *IEEE Control. Syst. Mag.* **2000**, *20*, 53–62. [CrossRef]
15. Mu, J.; Rees, D. Approximate model predictive control for gas turbine engines. In Proceedings of the 2004 American Control Conference, Boston, MA, USA, 30 June–2 July 2004; Volume 6, pp. 5704–5709. [CrossRef]
16. Afram, A.; Janabi-Sharifi, F.; Fung, A.S.; Raahemifar, K. Artificial neural network (ANN) based model predictive control (MPC) and optimization of HVAC systems: A state of the art review and case study of a residential HVAC system. *Energy Build.* **2017**, *141*, 96–113. [CrossRef]
17. Li, S.; Jiang, P.; Han, K. RBF Neural Network based Model Predictive Control Algorithm and its Application to a CSTR Process. In Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; pp. 2948–2952. [CrossRef]
18. Maddalena, E.; Moraes, C.D.S.; Waltrich, G.; Jones, C. A Neural Network Architecture to Learn Explicit MPC Controllers from Data. *IFAC-PapersOnLine* **2020**, *53*, 11362–11367.
19. Nubert, J.; Köhler, J.; Berenz, V.; Allgöwer, F.; Trimpe, S. Safe and Fast Tracking on a Robot Manipulator: Robust MPC and Neural Network Control. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3050–3057. [CrossRef]
20. Shin, Y.; Smith, R.; Hwang, S. Development of model predictive control system using an artificial neural network: A case study with a distillation column. *J. Clean. Prod.* **2020**, *277*, 124124. . [CrossRef]
21. Núñez, F.; Langarica, S.; Díaz, P.; Torres, M.; Salas, J.C. Neural Network-Based Model Predictive Control of a Paste Thickener Over an Industrial Internet Platform. *IEEE Trans. Ind. Inform.* **2020**, *16*, 2859–2867. [CrossRef]
22. Pan, Y.; Wang, J. Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. *IEEE Trans. Ind. Electron.* **2012**, *59*, 3089–3101. [CrossRef]
23. Pon Kumar, S.S.; Tulsyan, A.; Gopaluni, B.; Loewen, P. A Deep Learning Architecture for Predictive Control. *IFAC-PapersOnLine* **2018**, *51*, 512–517. [CrossRef]
24. Shahnazari, H.; Mhaskar, P.; House, J.M.; Salsbury, T.I. Modeling and fault diagnosis design for HVAC systems using recurrent neural networks. *Comput. Chem. Eng.* **2019**, *126*, 189–203. .

[CrossRef]
25. Wu, Z.; Tran, A.; Rincon, D.; Christofides, P.D. Machine learning-based predictive control of nonlinear processes. Part I: Theory. *AIChE J.* **2019**, *65*, e16729. [CrossRef]
26. Wu, Z.; Rincon, D.; Christofides, P.D. Real-Time Adaptive Machine-Learning-Based Predictive Control of Nonlinear Processes. *Ind. Eng. Chem. Res.* **2020**, *59*, 2275–2290. [CrossRef]
27. Huang, K.; Wei, K.; Li, F.; Yang, C.; Gui, W. LSTM-MPC: A Deep Learning Based Predictive Control Method for Multimode Process Control. *IEEE Trans. Ind. Electron.* **2022**, *70*, 11544–11554. [CrossRef]
28. Zarzycki, K.; Ławryńczuk, M. Advanced predictive control for GRU and LSTM networks. *Inf. Sci.* **2022**, *616*, 229–254. [CrossRef]
29. Zheng, Y.; Zhao, T.; Wang, X.; Wu, Z. Online learning-based predictive control of crystallization processes under batch-to-batch parametric drift. *AIChE J.* **2022**, *68*, e17815. [CrossRef]
30. Cho, M.; Ban, J.; Seo, M.; Kim, S.W. Neural network MPC for heating section of annealing furnace. *Expert Syst. Appl.* **2023**, *223*, 119869. [CrossRef]
31. Jung, M.; da Costa Mendes, P.R.; Önnheim, M.; Gustavsson, E. Model Predictive Control when utilizing LSTM as dynamic models. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106226. . [CrossRef]
32. Meng, J.; Li, C.; Tao, J.; Li, Y.; Tong, Y.; Wang, Y.; Zhang, L.; Dong, Y.; Du, J. RNN-LSTM-Based Model Predictive Control for a Corn-to-Sugar Process. *Processes* **2023**, *11*, 1080. [CrossRef]
33. Achirei, S.D.; Mocanu, R.; Popovici, A.T.; Dosoftei, C.C. Model-Predictive Control for Omnidirectional Mobile Robots in Logistic Environments Based on Object Detection Using CNNs. *Sensors* **2023**, *23*, 4992. [CrossRef] [PubMed]
34. Sands, T. Comparison and Interpretation Methods for Predictive Control of Mechanics. *Algorithms* **2019**, *12*, 232. [CrossRef]
35. Rosolia, U.; Zhang, X.; Borrelli, F. Data-Driven Predictive Control for Autonomous Systems. *Annu. Rev. Control. Robot. Auton. Syst.* **2018**, *1*, 259–286. [CrossRef]
36. Rawlings, J.B.; Maravelias, C.T. Bringing new technologies and approaches to the operation and control of chemical process systems. *AIChE J.* **2019**, *65*, e16615. [CrossRef]
37. Schwenzer, M.; Ay, M.; Bergs, T.; Abel, D. Review on model predictive control: An engineering perspective. *Int. J. Adv. Manuf. Technol.* **2021**, *117*, 1327–1349. [CrossRef]
38. Schweidtmann, A.M.; Esche, E.; Fischer, A.; Kloft, M.; Repke, J.U.; Sager, S.; Mitsos, A. Machine Learning in Chemical Engineering: A Perspective. *Chemie-Ingenieur-Technik* **2021**, *93*, 2029–2039. [CrossRef]
39. De Myttenaere, A.; Golden, B.; Le Grand, B.; Rossi, F. Mean absolute percentage error for regression models. *Neurocomputing* **2016**, *192*, 38–48. [CrossRef]
40. Nazmul Siddique, H. *Intelligent Control: A Hybrid Approach Based on Fuzzy Logic, Neural Networks and Genetic Algorithms*; Springer: Cham, Switzerland, 2013.
41. Ahvanooey, M.T.; Li, Q.; Wu, M.; Wang, S. A Survey of Genetic Programming and Its Applications. *KSII Trans. Internet Inf. Syst.* **2019**, *13*, 1765–1794.
42. Zheng, C.; Eskandari, M.; Li, M.; Sun, Z. GA-Reinforced Deep Neural Network for Net Electric Load Forecasting in Microgrids with Renewable Energy Resources for Scheduling Battery Energy Storage Systems. *Algorithms* **2022**, *15*, 338. [CrossRef]
43. Koza, J.R.; Keane, M.A.; Yu, J.; Bennett, F.H.; Mydlowec, W. Automatic creation of human-competitive programs and controllers by means of genetic programming. *Genet. Program. Evolvable Mach.* **2000**, *1*, 121–164. [CrossRef]
44. Grosman, B.; Lewin, D.R. Automated nonlinear model predictive control using genetic programming. *Comput. Chem. Eng.* **2002**, *26*, 631–640. [CrossRef]
45. Vyas, R.; Goel, P.; Tambe, S.S. Genetic programming applications in chemical sciences and engineering. In *Handbook of Genetic Programming Applications*; Springer: Cham, Switzerland, 2015, pp. 99–140.
46. Lotter, W.; Kreiman, G.; Cox, D. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv* **2016**, arXiv:1605.08104.
47. Hosseini, M.; Maida, A.S.; Hosseini, M.; Raju, G. Inception-inspired lstm for next-frame video prediction. *arXiv* **2019**, arXiv:1909.05622.
48. Plaster, B.; Kumar, G. Data-Driven Predictive Modeling of Neuronal Dynamics Using Long Short-Term Memory. *Algorithms* **2019**, *12*, 203. [CrossRef]
49. Desai, P.; Sujatha, C.; Chakraborty, S.; Ansuman, S.; Bhandari, S.; Kardiguddi, S. Next frame prediction using ConvLSTM. *J. Phys. Conf. Ser.* **2022**, *2161*, 012024. [CrossRef]
50. Hong, S.; Kim, S.; Joh, M.; Song, S.K. Psique: Next sequence prediction of satellite images using a convolutional sequence-to-sequence network. *arXiv* **2017**, arXiv:1711.10644.
51. Tang, M.; Liu, Y.; Durlofsky, L.J. A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *J. Comput. Phys.* **2020**, *413*, 109456.

[CrossRef]
52. Kakka, P.R. Sequence to sequence AE-ConvLSTM network for modelling the dynamics of PDE systems. *arXiv* **2022**, arXiv:2208.07315.
53. Mukherjee, S.; Ghosh, S.; Ghosh, S.; Kumar, P.; Roy, P.P. Predicting video-frames using encoder-convlstm combination. In Proceedings of the ICASSP 2019—2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 2027–2031.