CLUE: Safe Model-Based RL HVAC ControL Using Epistemic Uncertainty Estimation

Zhiyu An
University of California, Merced
zan7@ucmerced.edu

Xianzhong Ding
University of California, Merced
xding5@ucmerced.edu

Arya Rathee*
University of California, Santa Cruz
arathee@ucsc.edu

Wan Du
University of California, Merced
wdu3@ucmerced.edu

Abstract

Model-Based Reinforcement Learning (MBRL) has been widely studied for Heating, Ventilation, and Air Conditioning (HVAC) control in buildings. One of the fundamental problems is the large amount of data required to train a neural network for building dynamics modeling. In this paper, we developed *CLUE*, a safe MBRL HVAC control approach that can achieve low human comfort violation with a dynamics model trained on a small dataset. We used Gaussian Process (GP) as the building dynamics model, which provides the uncertainty of each output. The uncertainty result is then integrated into a safe HVAC control algorithm. Although GP has been studied for HVAC control, this work provides a data-efficient GP modeling method. We designed a novel meta kernel learning technique that incorporates domain knowledge from historical data of multiple buildings to set the GP kernel hyperparameters. Our method can significantly reduce the amount of data required for GP hyperparameter setting. Furthermore, we incorporate the GP-based uncertainty into a Model Predictive Path Integral (MPPI) process to find a safe, energy-efficient action for each control cycle. We generate a large number of action trajectories by the GP building dynamics model, and find the optimal trajectory by a novel MPPI objective function that considers the uncertainty of every action in all trajectories. We then execute the first action of the optimal trajectory. Extensive experiments in a simulated five-zone building show that *CLUE* only needs seven days of training data to provide comparable energy saving as the state-of-the-art MBRL method, but with 12.07% less comfort violations.

CCS Concepts

• **Computing methodologies** → **Control methods**.

Keywords

Epistemic uncertainty estimation, Model-based reinforcement learning, HVAC control, Model predictive control

*This work was accomplished when Arya Rathee was an undergraduate student at UC Merced and conducted an internship in Dr. Wan Du's research group.

BuildSys '23, November 15–16, 2023, Istanbul, Turkey

ACM ISBN 979-8-4007-0230-3/23/11…$15.00
https://doi.org/10.1145/3600100.3623742

ACM Reference Format:
Zhiyu An, Xianzhong Ding, Arya Rathee, and Wan Du. 2023. *CLUE*: Safe Model-Based RL HVAC ControL Using Epistemic Uncertainty Estimation. In *The 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '23), November 15–16, 2023, Istanbul, Turkey.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3600100.3623742

1 Introduction

Model-Based Reinforcement Learning (MBRL) is a promising approach for optimizing Heating, Ventilation, and Air Conditioning (HVAC) control in comparison to traditional Model Predictive Control (MPC) and Model-Free RL (MFRL) methods [1]. MPC [2] requires an accurate building thermal dynamics model, which is hard to formulate analytically [3]. MFRL directly learns a control policy by interacting with the building, but it normally takes years to converge [4, 5]. MBRL trains a building dynamics model using historical data and uses the model to find the best control action [4, 6]. One fundamental problem of MBRL-based HVAC control is the large amount of training data required to converge. For example, MB^2C [4] models the building dynamics by an ensemble of deep neural networks and uses a Model Predictive Path Integral (MPPI) algorithm to find the best control action. The deep ensemble model requires 183 days of training data to produce accurate predictions for the controller.

Extensive experiments on a simulated five-zone office building demonstrate that even if years of training data is used, the model still has epistemic uncertainty due to the training data bias, i.e., the model's predictions are not accurate at some states that the building does not experience often. To address this problem, we propose an MBRL-based HVAC control approach that can tolerate the inaccuracy of the building dynamics model trained on a small dataset. Our approach is aware of the model's prediction uncertainty and takes conservative control action when the prediction is uncertain. To enable safe HVAC control, we needed to first quantify the epistemic uncertainty of building dynamics models.

Epistemic uncertainty estimation has been used to improve the control performance in robotic motion planning [7] and RL in general [8]. For problems with low dimensions and discrete state spaces, such as multi-armed bandits [9], the count-based method [10] is used. It estimates the model error of the prediction on input by counting the number of data points in the training data with this input. However, building dynamics involve high dimensional continuous variables, which makes count-based methods unsuitable. Additionally, deep ensemble (DE) [8, 11] uses several neural networks to collectively

make a prediction and measures the level of agreement between the predictions as an uncertainty estimation result. As introduced above, it requires a large amount of data to train the neural networks for HVAC control.

To close this gap, we propose *CLUE*, a novel MBRL system for safe HVAC control, which uses a Gaussian Process (GP) model for the uncertainty-aware modeling of building dynamics. A GP model takes the state of the building (zone temperature, outdoor temperature, occupancy, etc.) and an action (heating and cooling setpoints) as inputs. It outputs a prediction of the state in the next time step as a Gaussian distribution comprised of a prediction mean and a variance. The variance is larger when the prediction is less certain, usually due to the lack of data about the given input. We used this variance as an indicator of epistemic uncertainty. Previous research in building control has demonstrated that GP models, in terms of model error, have the potential to outperform neural networks, random forests, and support vector machines [12, 13]. However, to incorporate GP into a safe and data-efficient HVAC control process, *CLUE* has two novel components. 1) Although GP methods are non-parametric, they require a predefined kernel function, which has a set of hyperparameters to instantiate. We design a new training procedure of GP-based building dynamics modeling, called meta kernel learning, which significantly improves the efficiency of the GP hyperparameter setting. 2) We incorporate the GP-based uncertainty into an MPPI algorithm to find a safe, energy-efficient action.

Selecting appropriate kernel parameters for GP with limited data remains a challenge [14]. Traditional kernel selection either relies on human experts to hand-pick the kernel parameters or uses gradient descent to train them. These methods are either costly or require large amounts of data to train a suitable kernel. If the kernel parameters are unsuitable, GP may perform poorly [15]. To address this issue, we designed a *meta kernel learning* procedure. Our key observation is that while it is sometimes difficult to obtain a large amount of historical data for a target building, it is easy to obtain large and comprehensive datasets from other buildings. With meta kernel learning, the reference data from multiple buildings enables the building dynamics model to automatically learn an effective kernel initialization, which can significantly reduce the learning burden without impacting the model's accuracy. In our experiment, we found that our method outperformed all baselines in terms of modeling accuracy and uncertainty estimation accuracy.

To effectively translate uncertainty estimation to safe control actions, we developed a confidence-based control algorithm, which allows us to safely and effectively apply MBRL when the model is inaccurate. With the MPPI control algorithm, we first generated a large number of action trajectories by our building dynamics model. Trajectories are the sequences of actions for the coming future time steps. The MPPI finds the trajectory that provides the highest energy savings and lowest human comfort violation rate. The controller then executes the first action of that trajectory. To incorporate GP-based uncertainty estimation in the above MPPI process, we developed a two-stage uncertainty-aware HVAC control algorithm that selects an action with a high reward and confidence. First, we used a threshold to filter out the trajectories whose first action has high uncertainty. Even if these trajectories are selected, their first actions cannot be executed due to high uncertainty. To find the best confidence threshold for safe HVAC control, we needed to know the relationship between the confidence value provided by the GP model and the expected model error. We designed an algorithm that can optimize the uncertainty threshold offline by testing the dynamics model on historical data. Second, we selected the optimal trajectory with a new MPPI objective function that considers the uncertainty of every action in all the remaining trajectories. Finally, we designed a fallback mechanism that employs a relatively reliable default control policy to override the MBRL control actions when no safe action is selected by the above control process.

We evaluated *CLUE* with comprehensive simulations in a $463m^2$ five-zone office building in three cities with EnergyPlus [16]. We first tested the uncertainty estimation accuracy of the proposed GP-based building dynamics model. The results show that the building dynamics model in *CLUE* outperformed all baselines in terms of modeling and uncertainty estimation accuracy. We then employed *CLUE* to control the simulated building and compared the human comfort and energy consumption between our system and the state-of-the-art MBRL solution. *CLUE* on average reduced comfort violations by 12.07% compared to the state-of-the-art MBRL method, with similar energy-saving and excellent data efficiency, i.e., *CLUE* reduced the data requirement from hundreds of days to only seven days.

In summary, this paper provides the following contributions.

- We are the first to include epistemic uncertainty estimation in shooting-based MBRL methods for HVAC control.
- We develop *CLUE*, a safe MBRL HVAC control system that adopts two novel design components, meta kernel learning and confidence-based control.
- We conducted comprehensive experiments with EnergyPlus simulations.

2 MBRL for HVAC Control

HVAC control can be formulated as a Markov Decision Process (MDP), denoted as $\mathcal{M} : \{\mathcal{S}, \mathcal{A}, r, \mathcal{P}, \gamma\}$, consisting of the state space $\mathcal{S}$, the action space $\mathcal{A}$, the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, the dynamics function $\mathcal{P}(s'|s, a)$ and discount factor γ. At each time step t, the controller is in state $s_t \in \mathcal{S}$, executes some action $a_t \in \mathcal{A}$, receives reward $r_t = r(s_t, a_t)$, and transitions to the next state s_{t+1} according to the dynamics function $s_{t+1} \sim \mathcal{P}(s_t, a_t)$. At each time step, the goal is to choose the action that maximizes the discounted sum of future rewards, given by $\sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'})$, where $\gamma \in [0, 1]$ is the discount factor that prioritizes near-term rewards.

MBRL-based control has two parts: the dynamics model and the controller. The dynamics model is used to make predictions, which are then used by the controller to choose an action. Let $f_\theta(s_t, a_t)$ denote a learned discrete-time dynamics model parameterized by θ, given a set of historical data $\{(s_t, a_t, s_{t+1})\}_n$. The dynamics model takes the current state s_t and action a_t and outputs a prediction of the next state at s_{t+1}. The controller then solves the following optimization problem:

$$(a_t, \cdots, a_{t+H-1}) = \arg \max_{(a_t, \cdots, a_{t+H-1})} \sum_{t'=t}^{t+H-1} \gamma^{t'-t} r(s_{t'}, a_{t'}) \quad (1)$$

The controller picks the action sequence that maximizes the cumulative discounted rewards of the future H time steps. In practice, it is often desirable to solve this optimization at each time step, i.e., executing only the first action from the sequence and then re-planning at the next time step with the updated state information. While the

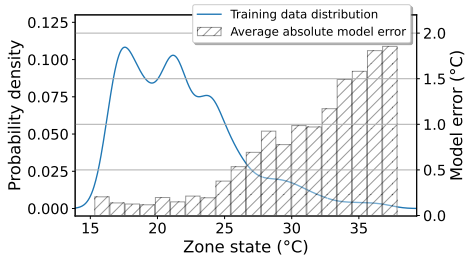

Figure 1: Model errors are >10× higher in data-sparse regions compared to data-dense regions.

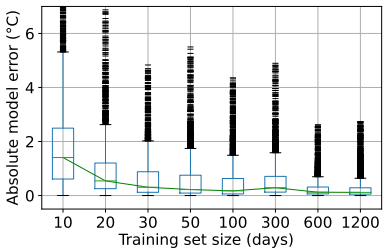

Figure 2: Model error distribution vs. training data set size.

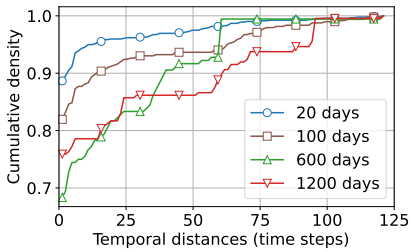

Figure 3: CDF of the distances between model errors for each training set size.

Disturbances	Outdoor Air Drybulb Temperature ($°C$)
	Outdoor Air Relative Humidity ($\%$)
	Site Wind Speed (m/s)
	Site Total Radiation Rate Per Area (W/m^2)
	Zone People Occupant Count ($No.$)
Zone State	Zone Air Temperature ($°C$)
Action	Zone Temperature Setpoint ($°C$)

Table 1: State and action variables.

random shooting method [17] is often used to solve the optimization problem in Eq.1, recent work has shown that MPPI is able to provide better optimization results [4]. To enable the above MBRL-based HVAC control process, we referred to MB^2C [4] for the following three MBRL components.

States. A state is a set of variables that are used as inputs and outputs of the building dynamics model. Two sets of states are defined: the disturbances and the zone state. The variables associated with these states are specified in Table 1. The disturbances comprises of variables that do not depend on the control action of the HVAC system, including weather conditions and occupancy. The zone state variable is the temperature of the controlled thermal zone, which depends on our control action and is used to calculate the building system reward.

Actions. The action is the temperature setpoint of the controlled thermal zone. In our experimental platform, the maximum and minimum temperature setpoints for the HVAC system are $30°C$ and $15°C$, respectively. Each controlled thermal zone is associated with a heating setpoint and a cooling setpoint, resulting in an action dimension of 2 for each zone.

Rewards. In our experimental platform, we adopted the same reward function as described in [18], represented by Eq. 2. The comfort zone is defined by two temperatures, $\overline{z}$ and $\underline{z}$, which represent the upper and lower bounds for the zone temperature, respectively. Here, $\overline{z}$ denotes the upper comfort limit, and $\underline{z}$ denotes the lower comfort limit. At each time step t, Z_t represents the zone temperature, and E_t represents the total energy consumption. To balance the relative importance of comfort and energy consumption, we used a weight variable $w_e \in [0, 1]$. This weight variable allows us to adjust the trade-off between comfort and energy consumption according to the specific requirements of the system.

$$r(s_t) = -w_e E_t - (1 - w_e)(|Z_t - \overline{z}|_+ + |Z_t - \underline{z}|_+) \quad (2)$$

In Eq. 2, we set $w_e = 0.1$ during occupied periods and $w_e = 1$ during unoccupied periods. E_t is approximated by the L-1 norm of the difference between the zone temperature set point (action) and the zone temperature at the current time step, i.e. the amount of heating/cooling provided by the HVAC actuator [6]. The lower and upper comfort limits $\underline{z}$ and $\overline{z}$ are $20°C$ and $23.5°C$, respectively, for the winter and $23°C$ and $26°C$ for the summer.

3 Motivation

To understand the prediction performance of existing state-of-the-art MBRL methods [4, 17], we performed a set of EnergyPlus simulations [16, 19] in a $463m^2$ building with five zones [18, 20]. In this simulation, we utilized an actual 2021 TMY3 weather profile from Pittsburgh, PA [18]. We employ a single climate for motivation experiments, while the actual analysis encompasses three climate zones. To investigate the prediction errors of building dynamics models, we experimented with the state-of-the-art deep ensemble method introduced in [4]. Specifically, we examined three key aspects: first, the relationship between model errors and the distribution of training data; second, the impact of varying training data sizes; and third, the temporal distribution of significant model errors.

Experiment results. Figure 1 depicts the relationship between the distribution of the training data and the prediction error of the building dynamics model. The model was trained on $120,000$ time steps (3.42 years) of data collected with the default Proportional–Integral–Derivative (PID) controller for 150 epochs and a learning rate of $1e-3$ to ensure convergence. After training, we employed the model to predict the subsequent 3000 time steps and recorded the prediction errors along with the corresponding zone state at the prediction's input, i.e., the starting zone state. To better analyze the results, we categorized the model errors based on the zone temperature in their inputs and calculated the average model errors for each bin.

In Figure 1, we found that the deep ensemble exhibited significantly higher errors when predicting in data-sparse regions of the state space. Notably, the deep ensemble model displayed greater accuracy when the zone state that fell within the temperature range of $15°C$ to $25°C$. Although the average model error is $0.29°C$, the model error consistently exceeded $1°C$ when the zone state exceeded $32°C$. The highest model error recorded was 9.36× higher than the average error and a staggering 15325× higher than the smallest model error.

To understand the substantial prediction errors observed in the deep ensemble models, we employed Kernel Density Estimation (KDE) to analyze the density distributions of the collected training data comprising 10,000 transitions. The blue curve presents the typical distribution patterns of the training data for the dynamics model

in MBRL-based HVAC control. It exhibits two clear modes around $17.5^{\circ}C$ and $21^{\circ}C$, representing the predominant temperatures during night and day. The majority of the transition data clusters around one or two central modes. The intrinsic bias in the thermal data adversely affects the prediction accuracy of data-driven dynamics models.

Two Questions Arise: The first question pertains to whether training models on more historical data can mitigate disastrous model errors. Unfortunately, the answer is no. We trained a deep ensemble model on different data sizes, ranging from 10 to 1200 days, and assessed its absolute error on the subsequent 30 days. The result is shown in Figure 2. We considered a model error larger than 2°C as unacceptable, since in the building sensor domain, a 2°C deviation from the true temperature is considered a sensor fault [21]. Despite training with nearly 4 years of data, model errors larger than 2°C persisted, i.e. 2.1% of all predictions deviated more than 2°C from the ground truth. The epistemic uncertainty from intrinsic bias in building thermal data significantly impacts accurate predictions, resulting in sub-optimal actions.

The second question revolves around whether we can ignore model errors if they occur independently over time, allowing the thermal system to tolerate short periods of controller glitches. Unfortunately, the answer is also no. To answer this question, we employed the deep ensemble model to make predictions with each of the 3000 time steps (one year) of data in the previous experiments, recorded the model errors, and flagged the model errors that exceed $2^{\circ}C$. Then, we went through all model errors in chronological order and recorded the number of time steps until we encounter every other flagged model error, i.e. the temporal distances between adjacent flags. Finally, we plotted the cumulative density function of the distances. To show how the trend changes with the size of the training data set, we conducted the above experiment four times using the same deep ensemble model trained on datasets of different sizes, ranging from 20 to 1200 days. The results are shown in Figure 3. Our analysis reveals that a significant majority (ranging from 68% to 89%) of the distances are 0, which means that the sizeable model errors are often consecutive. In other words, the majority of high model errors occur in clusters. If these errors are ignored, the thermal system potentially faces hours of controller dysfunction.

Our Key Idea. Based on the above observations, our main goal is to mitigate the adverse effects of high model error caused by distribution bias. Rather than attempting to refine the model or the training process to reduce the error, which we have shown might be unfeasible, we designed a procedure that identifies and alerts the controller when the model error *will* be high in the current time step. In essence, our system *predicts* the uncertainty about the model's *prediction* using its training data and the current input. Once a high error state-action pair is flagged, we either discard the prediction and opt for more confident alternatives, or in cases where no prediction is highly confident, allow the building's default controller to override the control action, compensating for the lack of data in that region of the state space.

4 The Design of *CLUE*

In this section, we describe the design of *CLUE*, including problem formulation, modeling system dynamics with the Gaussian process, and confidence-based MPPI controller.

4.1 Overview

Figure 4 depicts the overview of *CLUE*. At a high level, *CLUE* comprises two major components: a building system dynamics model and an MPPI-based controller. Our building dynamics model is a Gaussian process (GP) model which takes the current state of the building HVAC system as an input, and outputs the next state of the building HVAC system with a confidence interval. The prediction and the confidence interval are then used by the confidence-based controller to choose the best action.

The workflow of *CLUE* is summarized as follows. Initially, the system performs meta kernel learning (Section 4.3) to compute the GP prior without any previous knowledge about the target building $\mathcal{B}$. Meta-kernel learning uses reference data, i.e. any data collected from other buildings or via simulations, to learn an initialization of the GP kernel. Once the historical data of the target building is available, the system fits a GP using data solely from the target building while incorporating the learned GP prior. The result is the final building dynamics model. *CLUE* then optimizes the uncertainty value threshold ϵ, which is stored in the MPPI algorithm.

Upon deployment, the system initiates the confidence-based control procedure. At each control step, our system generates MBRL prediction trajectories using the GP model. Any trajectory where the uncertainty of the first time step exceeds the threshold ϵ is discarded. If all trajectories are discarded, the system sends the default controller action to the actuator. Otherwise, the remaining trajectories are utilized by an MPPI to compute the optimal action sequence $a(\cdot)$. The first action from this sequence is then sent to the actuator.

Lastly, the system awaits a single time step interval (e.g., 15 minutes in our case), observes a new state, and appends the corresponding tuple to the historical dataset $\mathcal{D}_{\mathcal{B}}$ before starting a new planning cycle.

4.2 Modeling Building Dynamics with GP

We model the building dynamics using GP, and use the variance of each prediction as the uncertainty value, i.e. higher variance indicates higher uncertainty. We concatenate the environment state (outdoor air temperature, humidity, occupancy, etc.) and the zone states (zone air temperature) together to form $s_t \in \mathcal{S}$, then concatenate s_t with the action $a_t \in \mathcal{A}$ (heating and cooling setpoints) to get the input variable. We will denote the input variable as $x \in \mathcal{X} = \mathcal{S} \times \mathcal{A}$ for simplicity. The output variable is the zone state at the next time step s_{t+1}. To make the notations consistent, we denote target output variable for x_i to be y_i. We further define our historical data set as $\mathcal{D} = (X, Y) = \{(x_i, y_i)\}_{i=1}^{n}$.

Modeling with GP involves a two-stage process. It starts from a pool of candidate functions (GP prior) and computes beliefs conditioned on training data (GP posterior). We will introduce both stages in the following.

First, we choose a function that, intuitively, quantifies the similarity between two inputs, i.e. $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. This function is called the GP kernel. The idea is that buildings in similar states that are taking similar actions should transition to a similar state. Thus, if $k(x_1, x_3) > k(x_2, x_3)$, then y_1 should be more indicative about y_3 than y_2 when extrapolating on x_3. Notably, each variable in our inputs contributes differently to their similarity to other inputs, depending on their level of influence according to the unknown dynamics function. The kernel function addresses this difference using

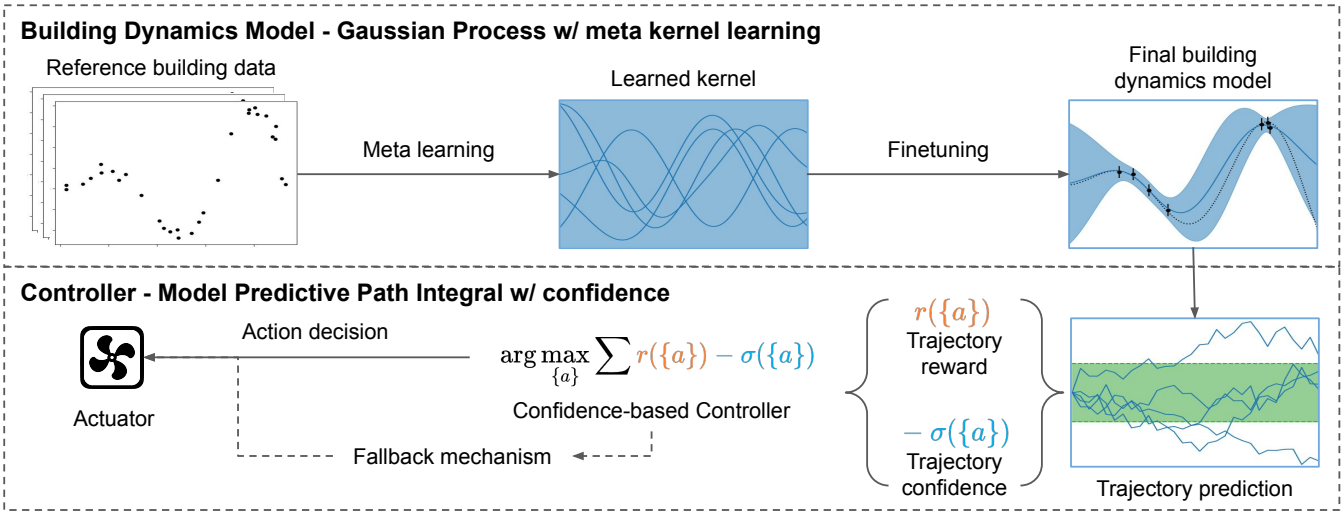

Figure 4: Overview of the proposed *CLUE* system.

adjustable parameters, dubbed hyperparameters. *CLUE* adopts the Radial Basis Function (RBF) kernel represented by Eq. 3. The RBF kernel is known for its expressive nature and has previously been used to model building thermal dynamics [12]. The hyperparameters are $\theta : \{\theta_{\text{scale}}, \Theta\}$, where θ_{scale} represents a scalar value, and Θ is an 8×8 matrix. Overall, the kernel parameters involve a total of 65 real numbers. The selection of these parameters is crucial as their values significantly impact both the modeling accuracy and the uncertainty estimation accuracy.

The second stage involves fitting data and making predictions. *CLUE* uses the exact regression technique, where we first compute the covariance matrix $K := k(X, X)$, and then calculate the GP posterior by Eq. 4 and Eq. 5, where I is the identity matrix and σ_n^2 is the noise variance (default to be 0) to account for aleatoric uncertainty. From this posterior, the prediction of y_* at x_* follows a Gaussian distribution as per Eq. 6.

$$k(x, x') = \theta_{\text{scale}} \exp\left(-\frac{1}{2}(x - x')^\top \Theta^{-2}(x - x')\right) \tag{3}$$

$$m_{\text{post}}(x) = m(x) + k(x, X)(K + \sigma_n^2 I)^{-1}(y - m(X)) \tag{4}$$

$$k_{\text{post}}(x, x') = k(x, x') - k(x, X)(K + \sigma_n^2 I)^{-1} k(X, x') \tag{5}$$

$$\mathcal{GP}(x_*) = \mathcal{N}(m_{\text{post}}(x_*), k_{\text{post}}(x_*, x_*)) \tag{6}$$

The prediction mean $m_{\text{post}}(x_*)$ represents the most likely model outcome, while the variance $k_{\text{post}}(x_*, x_*)$ indicates the level of uncertainty associated with the model outcome given data $\mathcal{D}$ and input x_*. The variance of prediction y_* is higher if x_* is in the data-scarce region of the input space $\mathcal{X}$ than if x_* is in the data-dense region. Therefore, we use variance as the indicator of epistemic uncertainty.

4.3 Meta Kernel Learning

To effectively model the building dynamics with GP, we optimize the initialization of the kernel parameters using data from a diverse set of reference buildings. To achieve this goal, we develop a new meta kernel learning method that combines two techniques, i.e., meta learning [22] and kernel learning [15].

Given a set of training data $\mathcal{D} : \{X, Y\}$, kernel learning [15] automatically optimizes the kernel parameters using gradient descent. It defines the loss function as the difference between the GP model's prediction and the ground truth, $\mathcal{L}(\mathcal{GP}_{\theta_k}) = MSE(\mathcal{GP}_{\theta_k}(X), Y)$. It calculates the gradient of the loss function with respect to the kernel parameters, $\nabla_{\theta_k}\mathcal{L}(\mathcal{GP}_{\theta_k})$, and updates the kernel parameters with the gradient. This process is repeated for a number of iterations to obtain a set of kernel parameters with a low model error.

Meta learning [22] trains a parameterized agent on multiple diverse tasks, enabling the agent to learn a suitable initialization of parameters from previously trained tasks. As a result, the agent can swiftly adapt to new tasks. Formally, instead of optimizing the model parameters according to $\theta_k^* = \arg\min_{\theta_k} \mathcal{L}(\mathcal{GP}_{\theta_k})$, meta learning optimizes the parameters according to $\theta_k^* = \arg\min_{\theta_k} \sum \mathcal{L}_{\mathcal{T}_i}(\mathcal{GP}_{\theta_k})$, where $\mathcal{T}_i$ is a task. It minimizes the sum of model error across a range of different tasks by accumulating the gradients across tasks, which we will introduce in the following.

In our meta kernel learning, we optimize the hyperparameters of a GP kernel, denoted as θ_k, using kernel learning. Rather than referring to MDPs as tasks, we consider modeling a dataset from a building over a specific period as a task. For instance, minimizing the modeling loss on the data from building A in Pittsburgh between May and August can be regarded as a task. The loss of the model on a task is defined in Eq. 7, which represents the model error of GP when using a kernel with hyperparameters θ_k on the mentioned dataset.

$$\mathcal{L}_{\mathcal{T}_i}(\mathcal{GP}_{\theta_k}) = MSE(\mathcal{GP}_{\theta_k}(X_{\mathcal{T}_i}), Y_{\mathcal{T}_i}) \tag{7}$$

$$\theta_k^* = \arg\min_{\theta} \sum_{i=1}^{n} \mathcal{L}_{\mathcal{T}_i}(\mathcal{GP}_{\theta_k})|_{\mathcal{T}_i \sim p(\mathcal{T})} \tag{8}$$

The objective of meta kernel learning is to minimize the sum of model errors across all tasks. We denote the set of tasks as $p(\mathcal{T})$. The objective function is defined in Eq. 8.

Algorithm 1: Meta Kernel Learning

Input: $p(\mathcal{T})$: distribution over all building data
Parameter: $\{\alpha, \beta\}$ step size hyperparameters
Randomly initialize θ_k
while *not converged* **do**
  Sample batch of building data $\{\mathcal{T}_1, \cdots, \mathcal{T}_i\} \sim p(\mathcal{T})$
  for *all* $\mathcal{T}_i$ **do**
    $\mathcal{GP} \leftarrow \text{GP Fit}(X_{\mathcal{T}_i}, Y_{\mathcal{T}_i})$
    $\theta_k' \leftarrow \theta_k - \alpha \nabla_{\theta_k} \mathcal{L}_{\mathcal{T}_i}(\mathcal{GP}_{\theta_k})$
  $\theta_k \leftarrow \theta_k - \beta \nabla_\theta \sum \mathcal{L}_{\mathcal{T}_i}(\mathcal{GP}_{\theta_k'})$

We present our design of the meta kernel learning procedure in algorithm 1. The kernel parameter is optimized using gradients accumulated across many tasks, and therefore improving the kernel while avoiding overfit to a single task. For the step size hyperparameters, we chose $\alpha = \beta = 1e-3$. This selection of step size leads to a relatively slower training process. However, in our experiments, we specifically chose a small step size to ensure convergence. The result of meta kernel learning is a set of kernel parameters that serve as the initialization of the kernel for the GP building dynamics model to be finetuned on the target building data. Once meta kernel learning is applied, we can obtain a building-specific kernel by fine-tuning the trained kernel using a minimal amount of data from the target building through traditional kernel learning.

4.4 Confidence-based Control

CLUE uses online planning with MPPI to select actions. Given the building state s_t at time t, the prediction horizon H, and an action sequence $a_{t:t+H} = \{a_t, ..., a_{t+H}\}$, our GP-based building dynamics model $\mathcal{GP}(x_*)$ makes predictions $s_{t:t+H}$. At each time step t, the MPPI controller applies the first action a_t of the optimized action sequence. Thus, it is critical that the first prediction of the trajectories has low model error. The uncertainty estimation does not directly translate to the expected model error. We translate the model error threshold to an uncertainty value threshold by testing the model on historical data. Then, *CLUE* identifies and filters out prediction trajectories whose uncertainty value exceeds the threshold.

4.4.1 Uncertainty Threshold Translation. Let the uncertainty value provided by the GP dynamics model at input x be $\sigma(x)$, model error threshold be e^* (measured in absolute error in Celsius degrees), and flagging threshold be $\epsilon \in \mathbb{R}^+$. If $\sigma(x) > \epsilon$, the state-action pair is flagged to have a high model error. Given an e^* specified by the engineers, the ideal translator maximizes the number of model errors>e^* flagged while minimizing the flags that have low model errors, i.e., the translator maximizes true positives and true negatives.

To find the appropriate threshold ϵ, we use an offline procedure that optimizes ϵ from historical data. Given a set of historical data of the target building $\mathcal{D} : \{X, Y\}$, we employ the current model $\mathcal{GP}$ to make a prediction on every input $x_i \in X$ in $\mathcal{D}$ and generate prediction results $(\vec{\mu}, \vec{\sigma}) = \mathcal{GP}(X)$, where $\vec{\mu}$ is the predicted building state and $\vec{\sigma}$ is the uncertainty value. We can then calculate $\vec{e} = |Y - \vec{\mu}|$, which is the absolute model error inferred by testing the current model with the historical data. For all predictions $\{(\mu, \sigma)\}_n$, we solve the following optimization problem:

$$\begin{aligned} &\text{minimize: } \text{count}(|y_i - \mu_i| < e^*) - \text{count}(|y_i - \mu_i| > e^*) \\ &\text{s.t. } \sigma_i > \epsilon \end{aligned} \tag{9}$$

By maximizing true positives and true negatives, *CLUE* finds the appropriate ϵ to be used for future predictions using the current model *GP*. This method works with any model error threshold $e^* \in \mathbb{R}^+$ specified by the engineers. *CLUE* applies this process offline to get the optimal flagging threshold, which is used in the confidence-aware MPPI control process.

4.4.2 Confidence-Aware MPPI. At each planning cycle, the MPPI controller computes a number of trajectories. *CLUE* checks the uncertainty value of the first time step of each trajectory and flags the trajectories that are expected to exceed the given model error threshold using the flagging threshold computed by the uncertainty threshold translator. Then, *CLUE* discards all trajectories that are flagged. This excludes high uncertainty thresholds from being used for action selection.

After filtering out trajectories according to their uncertainties, the confidence-based MPPI selects an optimal trajectory from the remaining trajectories. Although all remaining trajectories have low uncertainty at the first time step, the uncertainty values at the future time steps should also be considered, since they are related to the uncertainty of the predicted future rewards of the trajectories. *CLUE* addresses this by incorporating uncertainty into the optimization objective function, Eq. 10, where λ is a factor used to balance the magnitudes of uncertainty and reward. In other words, Eq. 10 determines the action sequence of the trajectory that maximizes the sum of discounted rewards while minimizing the sum of discounted uncertainty values. The design rationale behind Eq. 10 is that since the controller computes a roll-out trajectory in a bootstrap manner, i.e., for every time step in the prediction horizon, the impact of the model accuracy diminishes. In this design, the uncertainty value is discounted at the discount rate γ to take account of the relative importance of each uncertainty value. This modified objective function enables the controller to simultaneously optimize for both high reward and low uncertainty.

$$a(\cdot)^* = \arg\max_{a(\cdot)} \sum_{t=1}^{H} \gamma^t (r(x_t) - \lambda\sigma(x_t)) \tag{10}$$

$$a_{\text{new}} = a_{\text{prev}} + \frac{\sum_{k=1}^{K} \delta a_k \exp(\frac{1}{\eta} \sum_{t=1}^{H} \gamma^t (r(x_t) - \lambda\sigma(x_t)))}{\sum_{k=1}^{K} \exp(\frac{1}{\eta} \sum_{t=1}^{H} \gamma^t (r(x_t) - \lambda\sigma(x_t)))} \tag{11}$$

Implementing the designed optimization objective function into MPPI controllers is a straightforward and widely applicable process. To illustrate, we provided an example using the MPPI controller employed in recent research on HVAC control [4]. MPPI approximates the optimal action sequence by assessing the expected rewards of random trajectories and calculating the weighted sum of their action sequences using exponential weighting with respect to the cumulative discounted rewards generated by the action sequences [23]. A generic implementation of MPPI for HVAC control would use Eq. 2 for reward, such that $R = \sum_{t=1}^{H} \gamma^t r(s_t)$, where H is the prediction horizon length. In order to implement our proposed method, we modify the reward function to match the objective function in Eq. 10. The result is Eq. 11, where a is an action sequence, K is

the number of trajectories, δ_{a_k} is the action perturbation, and η is a hyperparameter. Eq. 11 shows how the weighted exponential sum of action sequences is now calculated with respect to the discounted sum of the system rewards and their confidence. In a similar manner, other MBRL controllers can implement our design by replacing the reward evaluation function.

4.4.3 Fallback Mechanism. When all trajectories exhibit high uncertainty, *CLUE* falls back to using the default controller as a safe and dependable alternative. Current HVAC systems' default controller is rule-based control, which mostly provides conservative and safe actions.

5 Evaluation

We conducted two sets of experiments to evaluate *CLUE*. First, we tested the modeling accuracy of GP with meta kernel learning and the uncertainty prediction accuracy of the fallback mechanism. Second, we deployed *CLUE* in the simulated environments and compared its performance with the state-of-the-art solutions.

5.1 Experiment Setting

5.1.1 Platform Setup. We used EnergyPlus [16] for high-fidelity building simulation, PyTorch [24] as the deep learning library, GPyTorch [25] for GPU-accelerated GP kernel learning, and Sinergym [18] as the virtual testbed that facilitates interaction with EnergyPlus in Python. Sinergym [18] sends the action chosen by *CLUE* to the EnergyPlus simulation session and sends the state observations back to *CLUE*. All software used for our experiment is open source.

5.1.2 Implementation details. Throughout our experiments, we used consistent experiment hyperparameters. We used epochs=150, learning_rate=1e-3, and weight_decay=1e-5 for DE models, iteration=800 and learning_rate=1e-2 for GP kernel learning, and iteration=200 and learning_rate=1e-2 for GP kernel finetuning. We used MSE as the loss criterion and Adam as the optimizer for all training. For meta kernel learning, the GP model trains on 35, 040 time steps (1 year) of data from each reference building. For the MPPI controller, we used the optimal hyperparameter configuration tested in [4], i.e. sample_number=1000 and horizon=20. For confidence-based MPPI, we used $\lambda = 1e-2$.

5.1.3 Environment selection. We conducted our simulation with a $463m^2$ building with five zones [18] in three climate-distinct cities from January 1st to January 31st and July 1st to July 31st. To test *CLUE*'s generalizability, we carefully chose three cities in the United States—Pittsburgh, Tucson, and New York—as our experiment environments. Each city represents a distinct climate type: Pittsburgh has a continental climate (ASHRAE 4A), Tucson has a hot desert climate (ASHRAE 2B), and New York has a humid continental climate (ASHRAE 4A) [26]. Actual 2021 TMY3 weather data for those cities are used [18]. This deliberate diversity in climate conditions allows our simulations to yield comprehensive insights applicable across various regions and climates.

5.1.4 Performance Metrics. We use two different sets of performance metrics to evaluate the uncertainty estimation accuracy and building control efficiency respectively.

1) Metrics for the uncertainty estimation:

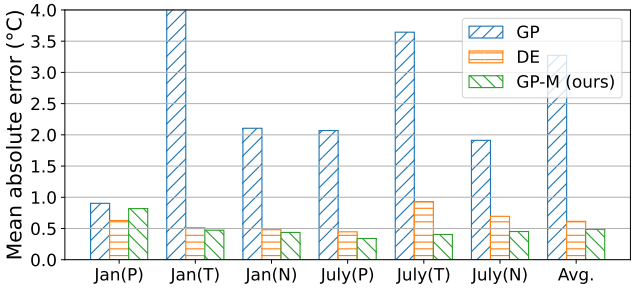

Figure 5: Model accuracy results.

- Accuracy: the sum of true positive (TP) and true negative (TN) results divided by the total number of results.
- Precision: the number of TP results divided by the number of all positive results.
- Recall: the number of TP results divided by the number of all samples that should have been identified as positive.

Note that we did not use the F_1 score metric due to its equal weighting of precision and recall [27], which does not align with our control system's objective. Instead, we directly logged the precision and recall values for informative data representation.

2) Metrics for building control efficiency:

- Cumulative reward: the weighted sum of the building system rewards calculated according to Eq. 2.
- Violation rate: the ratio of time steps where zone temperature violates comfort constraints to the total number of time steps.
- Energy consumption: total energy consumption in kWh.

5.1.5 Baselines. In terms of uncertainty estimation accuracy, we compared the performance of our method with the following baselines:

- Deep Ensemble [4]: an ensemble of five neural networks. Since uncertainty estimation is not adopted in [4], we use the uncertainty measurement method from [11], where the level of uncertainty is measured as the variance of a prediction Gaussian distribution: $\sigma_*^2(x) = M^{-1}\sum_m \mu_{\theta_m}^2(x) - \mu_*^2(x)$, where $M = 5$ is the number of models, $\mu_{\theta_m}(x)$ is the model m's prediction, and $\mu_*(x) = M^{-1}\sum_m \mu_{\theta_m}(x)$ is the mean of the predictions of all models.
- GP: GP method with kernel learning and identical uncertainty estimation as our approach, representing the generic approach for building thermal dynamics regression.

In terms of building control efficiency, we compared the performance of our proposed system with the following baselines:

- Rule-based: the default controller of the environment.
- DE-MBRL [4]: MBRL with deep ensemble and MPPI.
- *CLUE* w/o CB: *CLUE* without confidence-based control.

5.2 Modeling and Uncertainty Estimation

5.2.1 Modeling Accuracy. We compared our solution (GP with meta kernel learning, presented as GP-M in the figure) with the other two baseline methods, i.e., deep ensemble (DE) and GP. The results are shown in Figure 5. The DE model is trained on 2, 000 time steps (20.83 days) of data from the target building. The GP model is trained and fitted on the same data as above. The GP-M

model first employed meta kernel learning, then finetuned and fitted to the same data as the DE model. We found that, in terms of the average of all environments, the mean absolute model error of GP-M is 20.7% less than DE and 85.1% less than GP. GP-M outperformed GP in all environments and outperformed DE in five out of six environments. The result shows that meta kernel learning was able to effectively learn a suitable initialization for the kernel parameters, which significantly improved GP's modeling performance.

5.2.2 Uncertainty Estimation Accuracy. In the following experiment, we tested and compared the uncertainty estimation accuracy of GP-M and the baselines. We added our fallback mechanism (Section 4.4.3) to all three models and measured their abilities to accurately flag model errors larger than $1°C$. We used $1°C$ as a stricter threshold compared to the sensor fault ($2°C$)[21]. Then, we measured the accuracy, precision, and recall of them in the subsequent 30 days of the training set. Furthermore, to test the stability of each method, we trained new models and ran the same experiment five times. We then calculated the mean and the standard deviations of each result. The results are summarized in Table 2, where the best performances for each metric in each environment are in bold.

We found that our method consistently performed better than both baselines in terms of overall accuracy. Compared to GP, our method had a higher recall and lower precision, i.e. GP-M correctly flagged predominant portions of the large model errors, but also flagged some small model errors. This shows that GP-M is more conservative than GP, i.e. our method rather incorrectly flags a minor model error than missing a potential model error over the threshold.

In terms of stability, both vanilla GP and GP-M showed negligible instability ($< 0.5\%$ in all metrics). DE showed high instability across experiment runs, with a standard deviation of as high as 12% for recall. We attribute the high instability of the Deep Ensemble method to the randomness of the parameter initialization and the stochasticity of the training process of the neural network models. In comparison, GP-based methods are significantly more resistant to the said randomness.

5.2.3 Model Convergence. We conducted experiments to test the time step of convergence for *CLUE* and its DE-MBRL counterpart in terms of cumulative reward. The result is shown in Figure 6. In this experiment, DE-MBRL is trained offline using different amounts of data from the target building. Then the model freezes, and is used to control the 5-zone building in Pittsburgh in January. *CLUE* first applied meta kernel learning, then finetuned it using different amounts of data from the target building. The GP model is then fitted to 700 time steps of the target building data (the same data size as in [13]) to ensure effective modeling and computation efficiency. For the model error threshold e^* settings, we used exhaustive search to find the optimal settings between $0.5°C$ to $3°C$.

DE-MBRL took 50 days of offline training data to surpass the default controller and another 250 days of offline training data to reach its peak performance. *CLUE*'s performance converged at 7 days of data, where further training did not improve its control performance. With this insight in mind, we use 7 days of data for *CLUE* for the remainder of the experiments.

In the following experiments, the DE-MBRL was trained on $120,000$ (3.42 years) time steps of data gathered from the target building using the default PID controller. *CLUE* employed the same training procedure as above. The model error threshold is set to $0.5°C$. Then, we use *CLUE* and the baseline methods to control the building's HVAC system and observed the thermal comfort and energy-saving performances, as shown in the following subsections.

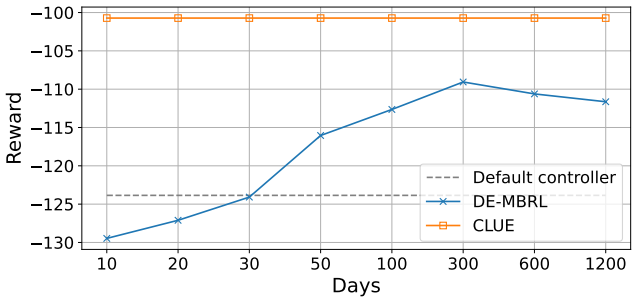

Figure 6: Data efficiency results.

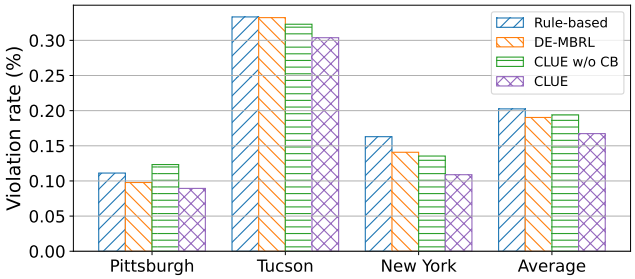

Figure 7: Comfort violation rate results.

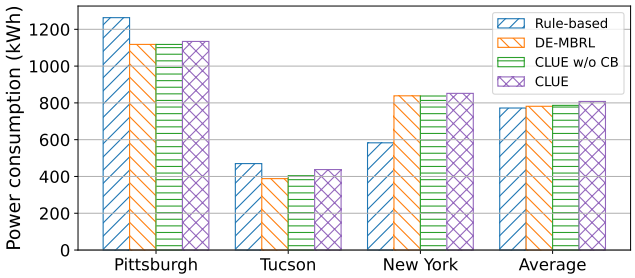

Figure 8: Energy consumption results.

5.3 Building Control

In terms of building control, we tested *CLUE* along with the baselines in the simulated environments of a $463m^2$ building with five zones [18] in three climate-distinct cities.

5.3.1 Thermal Comfort. We compared the violation rates of *CLUE* and the baselines in Figure 7. We found that *CLUE* consistently outperforms the baseline methods in terms of violation rates. Although the GP-M dynamics model used in *CLUE* has higher model error compared with DE as shown in Figure 5, *CLUE* still achieved a lower violation rate compared with its DE-MBRL counterpart, while *CLUE* w/o CB performed worse than DE-MBRL, as expected. This shows that *CLUE* is able to produce high-quality control actions even with an inaccurate dynamics model, which results in its excellent data efficiency.

Location	Time	Deep Ensemble [11]			GP			GP-M (ours)		
		Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Pittsburgh, PA	January	.796±.00	.521±.01	.740±.01	.877±.00	**.803±.00**	**.958±.00**	**.884±.00**	.768±.00	.677±.00
	July	.831±.01	**.851±.09**	.160±.10	.840±.00	.809±.00	.763±.00	**.961±.00**	.056±.00	**.999±.00**
Tucson, AZ	January	.736±.01	.439±.08	.693±.12	.847±.00	**.697±.00**	**.844±.00**	**.932±.00**	.341±.00	.694±.00
	July	.650±.00	.489±.00	.827±.00	.844±.00	**.854±.00**	.860±.00	**.947±.00**	.036±.00	**.999±.00**
New York, NY	January	.830±.00	.403±.02	.816±.00	.855±.00	**.883±.00**	.728±.00	**.965±.00**	.299±.00	**.900±.00**
	July	.679±.00	.373±.01	.812±.01	.797±.00	**.934±.00**	.718±.00	**.953±.00**	.205±.00	**.947±.00**

Table 2: Uncertainty estimation experiment results.

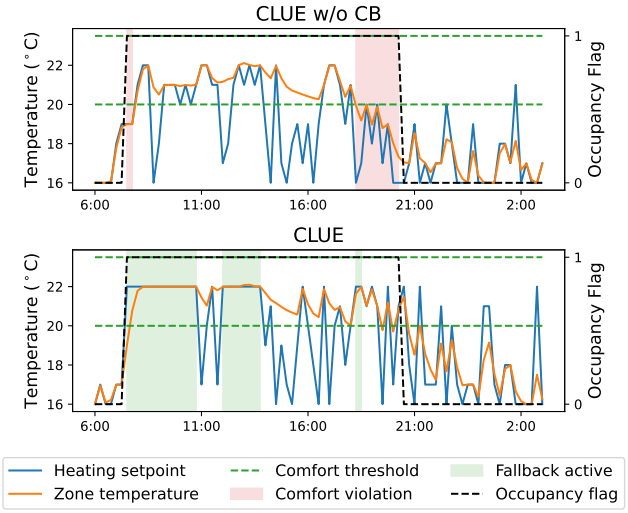

Figure 9: Analysis of the control performance gain of *CLUE*.

5.3.2 Energy Consumption. We compared the energy consumption of *CLUE* and the baselines in Figure 8. Overall, we found that *CLUE* consumes slightly more energy compared with its non-confidence-based counterparts. This is attributed to the operation of fallback mechanism, which consumes the same amount of energy as the rule-based controller during the time that the fallback is active. However, the energy consumption level of *CLUE* is still in-line with other MBRL methods. Notably, the low violation rate of *CLUE* makes it offer more time steps of comfort per unit of energy compared with its DE counterpart. This can make *CLUE* more preferable for the applications that prioritize comfort.

5.3.3 Effect of confidence-based control. To observe the effectiveness of the fallback mechanism, we compared *CLUE* with a version without the fallback mechanism, i.e., *CLUE* w/o CB, in Figure 7 and Figure 8. We found that while *CLUE* w/o CB performs better than DE-MBRL in two out of three times depending on if the model error for *CLUE* w/o CB is lower, *CLUE* was consistently superior to all baselines. This shows that the fallback mechanism ensures a high level of control performance even with an inaccurate model since our method can accurately predict its own model error.

5.3.4 Analysis of the control performance gain of CLUE. To investigate the reasons behind the building control performance gain of *CLUE* compared with the baseline method, we plotted the data from one day of our simulation experiment for *CLUE* and *CLUE* w/o CB in Figure 9. The simulation scenario was in January, so only the heating temperature setpoint was displayed. Out of the 85 time steps shown in the figure, 26 time steps had the fallback mechanism activated, i.e. the control action was overridden by the default controller about 30% of the times. During a one-day period, an ideal controller is expected to keep the zone temperature within the comfort range during the occupied times and let the zone naturally cool down to the environment temperature during the unoccupied times to save energy. There are usually two origins of performance gain for MBRL approaches [6]. First, the MBRL controllers learn to pre-heat the room to desired temperatures before the room is occupied to get a lower violation rate. Second, the MBRL approaches cool and re-heat the room repeatedly, keeping the temperature within the comfort range while saving energy during the time periods when the heating is turned off.

We found that the *CLUE* w/o CB made a mistake mid-way between 16 : 00 to 21 : 00. It falsely believed that the zone temperature would take longer to cool down and would turn the heating off prematurely. This usually happens when the dynamics model overestimates the room's thermal capacity. The same mistake happened to the DE-MBRL controller. After observing that the temperature has plummeted, it underestimated the amount of heating to reheat the zone and eventually caused 135 minutes of comfort violation. Our method, on the other hand, detected high model uncertainty and overrode 2 time steps of control action with the default controller between 16 : 00 to 21 : 00. This successfully kept the zone temperature within the comfort range and also within the data distribution, allowing the controller to correctly predict the amount of heating needed for the rest of the occupied times. As a result, the fallback mechanism prevented 135 minutes of comfort violation.

6 Related Work

Model-Free RL for HVAC. MFRL systems train a deep policy network for HVAC control using RL learning methods such as deep Q-learning [28] or actor-critic method [29]. They do not rely on a system dynamics model, but instead, a policy network that takes the current state and predicts the optimal action through interactions with the environment. Zhang et al. [29] implemented and deployed a DRL-based control method for radiant heating systems in a real-life office building. Ding et al. [28] proposed to use a deep RL model to control all building's subsystems, including HVAC, lighting, blind and window systems, using a tailored reward function. However, MFRL methods are data-hungry.

Model-Based RL for HVAC. Collecting large scale real-world data is often difficult [30]. To improve sample efficiency, researchers have

adopted MBRL for HVAC control [4, 17]. Zhang et al. [17] introduced an MBRL approach using a neural network to learn system dynamics and a random shooting method MPC for building control based on trajectory predictions. Chen et al. [6] employed offline training using expert demonstrations to reduce the data requirements. Ding et al. [4] expanded MBRL's control capabilities from a single zone to multi-zone buildings using a neural network as the dynamics model and a more efficient MPPI controller, achieving convergence with 183 days of training data. However, these building models lack uncertainty awareness and require hundreds of days of training data to reach satisfactory performance.

Safe RL. Safe RL has emerged as a critical research area to ensure the practical deployment of RL agents in real-world environments without causing harm [31]. One prominent method proposed by [32] is Constrained Policy Optimization, which adds safety constraints to the RL problem, restricting the agent from taking actions that violate safety requirements. However, selecting appropriate constraint formulations that accurately capture safety requirements remains challenging. Similarly, Model-Based Safe RL, as proposed by [33], employs learned environment models to simulate potential outcomes, enabling safer exploration. However, it may yield suboptimal real-world decisions due to potential model inaccuracies.

Safe RL for HVAC. Real-world exploration in a building can be hazardous when the outcomes of actions are unknown. Some studies [5, 29] propose using simulators to train RL agents and then deploying them in the building. However, safety is not guaranteed as mismatches between simulator data and real-world dynamics may make actions deemed "safe" in the simulator "unsafe" in reality. Another alternative involves utilizing batch RL methods [34, 35], which enable learning from historical data and enhancing the existing policy without requiring interactions with real buildings or simulators during training. However, the control performance of RL agents depends on data quantity and quality, making it challenging to ensure optimal outcomes.

7 Conclusion

This paper presents *CLUE*, a safe MBRL approach for HVAC control, excelling in high control performance despite training a dynamics model with limited data. *CLUE* leverages GP to quantify epistemic uncertainty caused by data scarcity. A novel meta-kernel learning technique is developed to effectively set GP kernel hyperparameters. GP-based uncertainty is integrated into a confidence-aware HVAC control process. Extensive evaluations demonstrate a 12.07% reduction in comfort violations and comparable energy-saving performance with just a seven-day historical dataset.

Acknowledgments

This work was supported in part by the UC National Laboratory Fees Research Program grant #69763. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker *et al.*, "Model-based reinforcement learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 16, no. 1, pp. 1–118, 2023.

[2] D. A. Winkler, A. Yadav, C. Chitu, and A. E. Cerpa, "Office: Optimization framework for improved comfort & efficiency," in *19th ACM/IEEE IPSN*, 2020.

[3] C. Agbi, Z. Song, and B. Krogh, "Parameter identifiability for multi-zone building models," in *2012 IEEE 51st IEEE CDC*. IEEE, 2012, pp. 6951–6956.

[4] X. Ding, W. Du, and A. E. Cerpa, "Mb2c: Model-based deep reinforcement learning for multi-zone building control," in *ACM BuildSys*, 2020, pp. 50–59.

[5] Z. Zhang and K. P. Lam, "Practical implementation and evaluation of deep reinforcement learning control for a radiant heating system," in *ACM BuildSys*, 2018.

[6] B. Chen, Z. Cai, and M. Bergés, "Gnu-rl: A precocial reinforcement learning solution for building hvac control using a differentiable mpc policy," in *ACM BuildSys*, 2019, pp. 316–325.

[7] W.-J. Baek, C. Ledermann, and T. Kröger, "Uncertainty estimation for safe human-robot collaboration using conservation measures," in *Proceedings of IAS-17*, 2023.

[8] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *NeurIPS*, 2018.

[9] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *European conference on machine learning*. Springer, 2005, pp. 437–448.

[10] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. Xi Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel, "# exploration: A study of count-based exploration for deep reinforcement learning," *NeurIPS*, vol. 30, 2017.

[11] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *NeurIPS*, vol. 30, 2017.

[12] F. Massa Gray and M. Schmidt, "Thermal building modelling using gaussian processes," *Energy and Buildings*, vol. 119, pp. 119–128, 2016.

[13] L. Goliatt, P. Capriles, and G. R. Duarte, "Modeling heating and cooling loads in buildings using gaussian processes," in *2018 IEEE CEC*. IEEE, 2018, pp. 1–6.

[14] T. X. Nghiem and C. N. Jones, "Data-driven demand response modeling and control of buildings with gaussian processes," in *ACC*. IEEE, 2017.

[15] D. Duvenaud, "Automatic model construction with gaussian processes," Ph.D. dissertation, University of Cambridge, 2014.

[16] DoE, "Energyplus input output reference," *US Department of Energy*, 2010.

[17] C. Zhang, S. R. Kuppannagari, R. Kannan, and V. K. Prasanna, "Building hvac scheduling using reinforcement learning via neural network based model approximation," in *ACM BuildSys*, 2019, pp. 287–296.

[18] J. Jiménez-Raboso, A. Campoy-Nieves, A. Manjavacas-Lucas, J. Gómez-Romero, and M. Molina-Solana, "Sinergym: a building simulation and control framework for training reinforcement learning agents," in *ACM BuildSys*, 2021, pp. 319–323.

[19] H. Rajabi, Z. Hu, X. Ding, S. Pan, W. Du, and A. Cerpa, "Modes: Multi-sensor occupancy data-driven estimation system for smart buildings," in *ACM e-Energy*, 2022.

[20] H. Rajabi, X. Ding, W. Du, and A. Cerpa, "Todos: Thermal sensor data-driven occupancy estimation system for smart buildings," in *ACM BuildSys*, 2023.

[21] D. Kumar, X. Ding, W. Du, and A. Cerpa, "Building sensor fault detection and diagnostic system," in *ACM BuildSys*, 2021, pp. 357–360.

[22] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*. PMLR, 2017, pp. 1126–1135.

[23] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE ICRA*, 2016.

[24] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *NeurIPS*, vol. 32, 2019.

[25] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," *NeurIPS*, vol. 31, 2018.

[26] A. STANDARD, "Ansi/ashrae addendum a to ansi/ashrae standard 169-2020," *ASHRAE Standing Standard Project Committee*, 2020.

[27] D. Hand and P. Christen, "A note on using the f-measure for evaluating record linkage algorithms," *Statistics and Computing*, vol. 28, pp. 539–547, 2018.

[28] X. Ding, W. Du, and A. Cerpa, "Octopus: Deep reinforcement learning for holistic smart building control," in *ACM BuildSys*, 2019, pp. 326–335.

[29] Z. Zhang, A. Chong, Y. Pan, C. Zhang, S. Lu, and K. P. Lam, "A deep reinforcement learning approach to using whole building energy model for hvac optimal control," in *2018 Building Performance Analysis Conference and SimBuild*, vol. 3, 2018, pp. 22–23.

[30] K. Yang, Y. Chen, X. Chen, and W. Du, "Link quality modeling for lora networks in orchards," in *22nd ACM/IEEE IPSN*, 2023, pp. 27–39.

[31] X. Ding and W. Du, "Drlic: Deep reinforcement learning for irrigation control," in *21st ACM/IEEE IPSN*. IEEE, 2022, pp. 41–53.

[32] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *ICML*. PMLR, 2017, pp. 22–31.

[33] A. K. Jayant and S. Bhatnagar, "Model-based safe deep reinforcement learning via a constrained proximal policy optimization algorithm," *NeurIPS*, 2022.

[34] C. Zhang, S. R. Kuppannagari, and V. K. Prasanna, "Safe building hvac control via batch reinforcement learning," *IEEE Transactions on Sustainable Computing*, 2022.

[35] H.-Y. Liu, B. Balaji, S. Gao, R. Gupta, and D. Hong, "Safe hvac control via batch reinforcement learning," in *ACM/IEEE ICCPS*. IEEE, 2022, pp. 181–192.