MB^{2}C: Model-Based Deep Reinforcement Learning for Multi-zone Building Control

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

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

Alberto Cerpa
University of California, Merced
acerpa@ucmerced.edu

ABSTRACT

Reinforcement learning has been widely studied for controlling Heating, Ventilation, and Air conditioning (HVAC) systems. Most of the existing works are focused on Model-Free Reinforcement Learning (MFRL), which learns an agent by extensively trial-and-error interaction with a real building. However, one of the fundamental problems with MFRL is the very large amount of training data required to converge to acceptable performance. Although simulation models have been used to generate sufficient training data to accelerate the training process, MFRL needs a high-fidelity building model for simulation, which is also hard to calibrate. As a result, Model-Based Reinforcement Learning (MBRL) has been used for HVAC control. While MBRL schemes can achieve excellent sample efficiency (i.e. less training data), they often lag behind model-free approaches in terms of asymptotic control performance (i.e. high energy savings while meeting occupants' thermal comfort).

In this paper, we conduct a set of experiments to analyze the limitations of current MBRL-based HVAC control methods, in terms of model uncertainty and controller effectiveness. Using the lessons learned, we develop MB^{2}C, a novel MBRL-based HVAC control system that can achieve high control performance with excellent sample efficiency. MB^{2}C learns the building dynamics by employing an ensemble of environment-conditioned neural networks. It then applies a new control method, Model Predictive Path Integral (MPPI), for HVAC control. It produces candidate action sequences by using an importance sampling weighted algorithm that scales better to high state and action dimensions of multi-zone buildings. We evaluate MB^{2}C using EnergyPlus simulations in a five-zone office building. The results show that MB^{2}C can achieve 8.23% more energy savings compared to the state-of-the-art MBRL solution while maintaining similar thermal comfort. MB^{2}C can reduce the training data set by an order of magnitude (10.52×) while achieving comparable performance to MFRL approaches.

CCS CONCEPTS

• **Computing methodologies** → Control methods.

KEYWORDS

HVAC Control, Model-based Deep Reinforcement Learning, Model Predictive Control

ACM Reference Format:
Xianzhong Ding, Wan Du, and Alberto Cerpa. 2020. MB^{2}C: Model-Based Deep Reinforcement Learning for Multi-zone Building Control . In *The 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '20), November 18–20, 2020, Virtual Event, Japan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3408308.3427986

BuildSys '20, November 18–20, 2020, Virtual Event, Japan

ACM ISBN 978-1-4503-8061-4/20/11…$15.00
https://doi.org/10.1145/3408308.3427986

1 INTRODUCTION

Buildings account for 40% of energy usage in the US and 50% of that energy goes to Heating, Ventilation, and Air Conditioning (HVAC) [1]. Rule-based Control (RBC) is widely used to set actuators (e.g., heating or cooling temperature, and fan speed) in HVAC systems [2]. One of the main advantages is that they are easy to understand. However, RBC "rules" are usually set some if-then rules using many times static thresholds based on the rule-of-thumb rules and the experience of engineers and facility managers. They have two fundamental problems: first, they do not scale well with the problem size, as the buildings become larger and more complex, rules must be added; second, they do not handle incomplete or incorrect information very well, an occurrence common in buildings in practice; and finally, they do not necessarily provide a guarantee of optimal control.

Model Predictive Control (MPC) has been widely studied to address these drawbacks by finding optimal control actions based on an analytical building model [3, 4]. Normally, an optimization problem is formulated with the building model and some constraints, and analytic gradient computation is used to optimize over actions and building states simultaneously. However, this often requires convexification of the cost function and first or second-order approximations of building dynamics [5] in order to solve the optimization problem fast and to scale well. As a result, the models used in current solutions are simplified to deal with the parameter-fitting data requirement and computational complexity [3, 4].

Reinforcement Learning (RL) has been widely studied for HVAC control [6–9]. Current solutions mainly adopt Model-Free Reinforcement Learning (MFRL), which learns an optimal HVAC control policy by trial-and-error interactions with a real building. However, MFRL requires a large amount of interactions to converge, e.g., in our experiments, it requires 500,000 timesteps (5200 days) to achieve a high control performance. Although a simulated building model can be used to accelerate the training process, it needs a high-fidelity model, which is hard to calibrate [6, 7]. Recently, Model-Based Reinforcement Learning (MBRL) has been tested for HVAC control to achieve high data efficiency [10]. The HVAC system

dynamics is first learned using a neural network based on historical HVAC data. Based on the learned building dynamics model, an MPC controller tries to find the optimal control action by using a Random Shooting (RS) method [10]. For controlling a single-zone HVAC system, an MBRL-based approach saves approximately 10× training time of the MFRL approach, while achieving comparable performance [10]. However, most of the commercial buildings are multi-zone buildings [11]. In addition to the above scheme not being suitable for multi-zone HVAC systems, MBRL often lags behind the MFRL schemes in terms of control performance (high energy saving while meeting the thermal comfort of occupants).

To overcome these limitations, this paper presents MB^{2}C, a novel MBRL-based HVAC control approach that can achieve both the data/sample efficiency of MBRL and the control performance of MFRL. The design goal of MB^{2}C is to meet the thermal comfort requirements of the occupants while saving as much energy as possible. The energy consumed by a building HVAC system and the thermal comfort of occupants are determined by a set of factors, including current state of all zones, the outdoor weather and the control actions we are about to take (e.g. temperature setpoints). In a multi-zone building, the control actions can be represented as a vector A_s, which is a combination of control actions for all thermal zones. MB^{2}C finds the best A_s from all possible action combinations A_{all} for each control cycle. The best A_s maintains the thermal comfort in its acceptable range for the entire control interval with the lowest energy consumption. MB^{2}C is mainly composed of two parts: (a) a building dynamics model, and (b) an HVAC control algorithm.

Our building dynamics model employs an ensemble of environment-conditioned neural networks. We use a neural network model that takes the current state of the building and the action to perform as input, and outputs a prediction of the next state of the building. To capture model uncertainty, we design a novel weighted ensemble learning algorithm that aggregates the results of multiple building dynamics models by dynamically adjusting the weight of each model according to their accuracy. We also adopt an environment-conditioned neural network architecture by separating the action-depended state items (e.g., zone temperature) and the environment-related state items (e.g., outside temperature), since the latter cannot be actuated by control actions.

Based on a learned building dynamics model, a flexible way to solve the control optimization problem is a shooting method that samples stochastic action trajectories for a number of incoming time-steps [12]. An action trajectory is a set of actions for incoming H time-steps. Every time, H time-steps are evaluated, but only the first action will be executed at the next time-step. For example, RS has been used in the latest MBRL-based HVAC control solution [10], which entails sampling candidate actions from a uniform distribution. However, RS is insufficient to find the best action trajectory, because randomly-shot action trajectories may not include it. We adopt Model Predictive Path Integral (MPPI) control method, which has shown promising performance in robotics control [13]. MPPI derives an optimal control action as the first action of a noise-weighted average over sampled control action trajectories by changing the initial control input and variance of the sampling distribution. We customize MPPI control for building HVAC control under the MBRL-based framework with the best parameter setting.

We implement MB^{2}C in Tensorflow, an open-source machine learning library in Python, with a 3-layer neural network as the building dynamics model and an MPPI-based control algorithm. We study the performance of MB^{2}C and compare it with benchmark methods by controlling a building of five thermal zones. We conduct a variety of simulations in EnergyPlus for evaluation. Extensive simulations reveal that MB^{2}C outperforms the latest model-based DRL method by 8.23% in total energy consumption of the building, without scarifying thermal comfort. Compared with the model-free DRL approach, we reduce the training convergence time by 10.52×, more than an order of magnitude improvement.

2 RELATED WORK

Model Predictive Control for HVAC. MPC solves an optimal control problem iteratively over a receding time horizon. [3] proposed an MPC approach for HVAC control, which minimizes energy use while satisfying occupant comfort constraints. A very recent MPC work, OFFICE [4], proposed a novel MPC framework that optimally manages the trade-off between energy cost and quality of comfort to the building users, by including input data from where users are (and will be), what users want, how zones react to changes in a data-driven manner, and current and forecast weather data. However, MPC control works well for low-order system dynamics, and its control variables must be carefully set for different buildings.

Model-free DRL for HVAC control. Reinforcement Learning has been applied to many areas [14–20]. In particular, MFRL techniques have demonstrated the potential optimal HVAC controls. In MFRL schemes, the agent learns the policy by extensively trial-and-error interaction with the environment. [9] leveraged RL to calculate thermostat set-points to balance between occupant comfort and energy efficiency. [7] implemented and deployed a DRL-based control method for radiant heating systems in a real-life office building. A holistic building control accounting for HVAC, lighting, window opening and blind inclination was studied using branching dueling Q-network (BDQ) in [6]. However, practical application of RL was limited by its sample complexity, i.e. the long training time required to learn control strategies, especially for tasks associated with a large state-action space. Gnu-RL [21] adopted a differentiable MPC policy, which encodes domain knowledge on planning and system dynamics, making it both data-efficient and interpretive. However, they assumed that dynamics of a water-based radiant heating system can be locally linearized. The assumption worked for the problems they considered, but it may not extrapolate to more complex problems like ours.

Model-based DRL for HVAC control. To reduce sample complexity, researchers have adopted model-based deep reinforcement learning for HVAC control [10]. In this work, they proposed an MBRL approach that learns the system dynamics using a neural network. Then, they adopt MPC using the learned system dynamics to perform control with RS method. MBRL method works well when the action and state dimension is low, like single-zone building. They often cannot achieve the final performance as model-free method when they are applied to high state and action dimensions of multi-zone buildings.

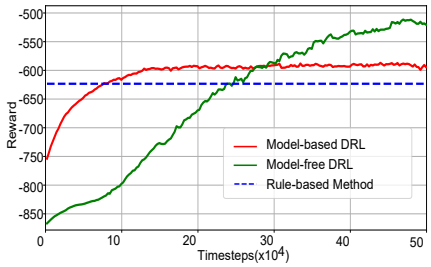

Figure 1: Convergence time and the achieved reward.

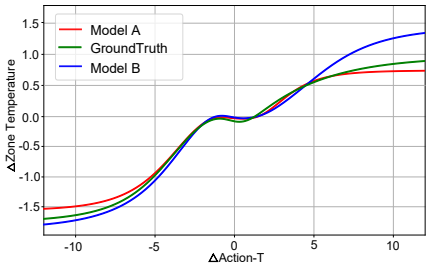

Figure 2: Uncertainty of the building dynamics model.

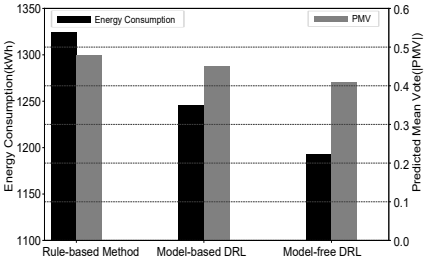

Figure 3: Random shooting in the model-based DRL method.

3 MOTIVATION

To understand the performance of a state-of-the-art MBRL method [10], we perform a set of simulations in EnergyPlus for a building with five zones. All system settings are the same as [10], except the state and action dimension is higher for the five-zone building, i.e. a multi-zone building instead of a single zone. We also implement a simple MFRL-based method, Proximal Policy Optimization (PPO) [22], for comparison in this preliminary evaluation. Thermal comfort is measured by PMV [23], which should be controlled within the range (-0.7~0.7). The simulations are conducted with weather data for the month of January. The building is 463 m^2 in Fresno CA. It has windows in all four facades and glass doors in south and north facades. The south-facing glass is shaded by overhangs. For our 5-zone building, the state dimension is 37, including indoor air temperature, humidity, PMV, energy consumption for each zone and related outdoor environmental parameters; and the action dimension is 10, including cooling and heating set points for each zone.

Experiment results. Figure 1 shows the energy-saving performance of model-based and model-free DRL control method with $50{\times}10^4$ time-steps of training data. The reward means the energy-saving performance under the reasonable thermal comfort that is defined in Section 4.2.4. We evaluate the accumulated reward every 2976 time-steps (one month). The performance of the rule-based method is a straight line, because its reward does not change as the weather data and building environment are deterministic.

From Figure 1, we can see that model-based DRL and PPO method need $7.5{\times}10^4$ and $23.75{\times}10^4$ time-steps to behave better performance than rule-based method. For converge time, the model-based method needs $11.5{\times}10^4$ and the PPO method needs $50{\times}10^4$ time-steps. The model-based method is 4.38× more data-efficient than PPO method. However, in the long run, the model-free method eventually outperforms the model-based method. It's easy to see that the model-free method is a trial and error method and the performance increases when using more training data. However, in this case, our model-based method cannot achieve the same performance as model-free method as the training data increases. The model-based method performs well when the action and state dimension is low (e.g., 9 in [10]). However, both the building dynamics model and the control method may not be efficient when the state and action dimension is high, like 47 in our 5-zone building.

Challenge 1 - Model Uncertainty. Neural network models may have epistemic uncertainty, due to the lack of sufficient data to uniquely model the underlying system [24–27]. In an MBRL-based HVAC control system, a building dynamics model predicts the next state of the building, given the current state (e.g. current zone temperature) and a control action (e.g. actuators' temperature set-points). Even a small bias of the building dynamics model may significantly impact the decision of the controller [25, 26]. We conduct an experiment to study this uncertainty of the existing building dynamics model. We use 8000 historical data points to train the model, and 2000 data points for testing.

Figure 2 shows the predictive zone temperature as a function of the action performed. The x-axis shows the temperature differential between the supply temperature (action) and the zone temperature at time t, and the y-axis shows the temperature differential between the zone temperature after and before actuation. The figure depicts the predicted temperatures of two neural network models and the ground truth. These two models have the same architecture and are trained with the same training data, but their training processes start with different initialization states. In the middle region of Figure 2, we have sufficient data, since most of the actions in the historical data do not change the state sharply. In this region, both models can accurately predict the next state. However, when the actions intend to change the state much, we do not have sufficient data for training, and the performance of the two models diverges.

Challenge 2 - Controller Effectiveness. RS generates N independent random action sequences $\{a_t, ...a_{t+H-1}\}$, where each sequence $A_i = \{a_0^i ... a_{H-1}^i\}$ for $i = 1...N$ is of length H action. Given a reward function $r(s,a)$ that defines the task, and given future state predictions $\hat{s}_{t+1} = s_t + f_\theta(\hat{s}_t, a_t)$ from the learned dynamics model f_θ, the optimal action sequence A_{i^*} is selected as the one with the highest predicted reward: $i^* = \arg\max_i R_i = \arg\max_i \sum_{t'=t}^{t+H-1} r(\hat{s}_{t'}, \hat{a}_{t'})$.

Figure 3 studies the energy consumption and thermal comfort of three HVAC control methods, including a rule-based method, a model-based method and a model-free method. To eliminate the impact of model uncertainty for the model-based method, we use the ground-truth states of the building as the results of the building dynamics model (i.e. perfect future state prediction). From the Figure 3, we can see that all three methods can meet the requirement of thermal comfort with same level of PMV value (0.48, 0.45, 0.41). The energy consumption of the model-based method is 4.70% higher than the model-free method. It is caused by RS control, because the building dynamics model used in the model-based method is perfect in this experiment.

Based on the previous observations, our main goal is to overcome the drawbacks of model uncertainty and controller effectiveness

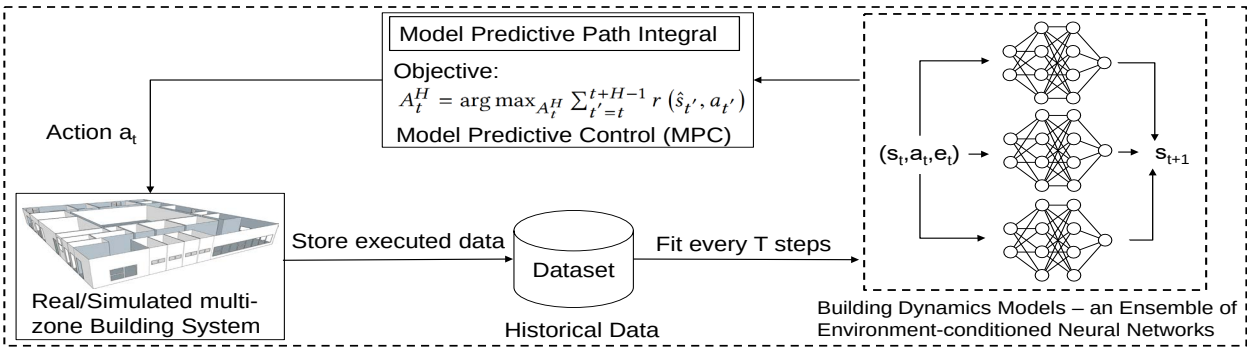

Figure 4: Overall of the Proposed Building Energy Control Framework

and find a method that is able to match the high performance of model-free methods while having the sample/data-efficiency of model-based methods.

4 DESIGN OF MB^{2}C

In this section, we describe the design of MB^{2}C, including model-based DRL for a multi-zone building control, the building dynamics model and its training details, online control action planning and in-situ update of the building dynamics model.

4.1 MB^{2}C Overview

Figure 4 shows the overview of MB^{2}C as a model-based DRL control approach [26] for multi-zone building HVAC systems. At a high level, MB^{2}C includes two key components, i.e., a building dynamics model and a Model Predictive Path Integral (MPPI) based controller. Our building dynamics model is built by an Ensemble of multiple Environment-conditioned Neural Networks (ENN). It takes the current state of the building HVAC system and a specific control action as input, and outputs the next state of the building HVAC system. Based on the historical data, we train the building dynamics model as a supervised learning process. With the trained building dynamics model, our MPPI-based controller can evaluate different control actions and find the best control action for next time step, which meets the thermal comfort requirement with minimal energy consumption.

When we deploy the system in a building, MB^{2}C executes the best control action by setting corresponding actuators every control cycle. At the same time, we accumulate building data traces, i.e., the next HVAC state determined by the current HVAC state and the executed control action. With the newly collected building traces, we can perform in-situ updating of the building dynamics model periodically (e.g., every week) with a sliding window of 2-months to improve its accuracy, as the seasonality of the data changes during the year. One iterative training process takes 25.32 minutes to finish using a laptop with Intel 4-core i7-6700 CPU and Nvidia GTX 960M GPU, and it can be performed in parallel when the current model is being used in the building; thus, the overhead of the iterative training process does not impact the usage of MB^{2}C in real buildings.

4.2 Model-Based Deep Reinforcement Learning for Multi-zone Building Control

We extend the current MBRL-based method to multi-zone building HVAC control, including the design of those key components.

4.2.1 Preliminaries for DRL. The goal of reinforcement learning is to learn a policy that maximizes the sum of future rewards. At each time step t, the controller is in state $s_t \in S$, executes some action $a_t \in A$, receives reward $r_t = r(s_t, a_t)$, and transitions to the next state s_{t+1} according to some unknown dynamics function $f : S \times A \rightarrow S$. The goal at each time step is to take 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 a discount factor that prioritizes near-term rewards. Note that performing this policy extraction requires knowing the underlying reward function $r(s_t, a_t)$ that we use for planning actions under the learned model.

In model-based reinforcement learning, a model of the dynamics is used to make predictions, which is used for action selection. Let $f_\theta(s_t, a_t)$ denote a learned discrete-time dynamics function, parameterized by θ, that takes the current state s_t and action a_t and outputs an estimate of the next state at time $t + \Delta t$. We can then choose actions by solving the following optimization problem:

$$(a_t, ...a_{t+H-1}) = \arg\max_{a_t,...a_{t+H-1}} \sum_{t'=t}^{t+H-1} \gamma^{t'-t} r\left(s_{t'}, a_{t'}\right) \tag{1}$$

In other words, we will pick the action sequence that maximizes the discounted sum of reward of future H time-steps. In practice, it is often desirable to solve this optimization at each time step, execute only the first action from the sequence, and then re-plan at the next time step with updated state information. Such a control scheme is often referred to as model predictive control (MPC), and is known to compensate well for errors in the model.

4.2.2 State Design. The state is what the building dynamics model takes as input for the next prediction step. In this study, we separate the state into 2 parts: (a) the building state (s_{ti}), which are the state variables that change with our control actions; and (b) the environment state (e_{ti}), which are the state variables that do not change with our control actions.

Building State (s_{ti}) The building state vector that changes over time t for the ith zone consists of the following items: indoor air temperature(°C), indoor air relative humidity (%), PMV, heating energy consumption (kWh) and cooling energy consumption (kWh).

Environment State (e_{ti}) The environment state vector that changes over time t for the ith consists of the following items: outdoor air temperature (°C), outdoor air relative humidity (%), diffuse solar radiation (W/m^2), direct solar radiation (W/m^2), solar incident angle (°), wind speed (m/s), wind direction and occupancy

flag (0 or 1). The occupancy flag is an indicator to detect whether there are people in the ith zone, and it is the only element in the vector that changes per zone.

Taking our 5-zone building as an example, the state dimension is 37 including the building, and environment state variables.

4.2.3 Action Design. The action vector (a_{ti}) shows the actuation variables used by the controller to control the building state (s_{ti}). The action state vector that changes over time t for the ith zone consists of the following items: cooling temperature set-point and the heating temperature set-point (both in °C). Given the current state (s_{ti} and e_{ti}) and action (a_{ti}), we want the controller to find the most suitable action combinations ($a_{(t+1)i}$) for all the zones to balance energy consumption and thermal comfort metrics. The action dimension is 10 in our five-zone building.

4.2.4 Reward Design. The reward function controls the optimization parameters that want to be maximized when the agent performs an action (a_{ti}) to transition from the building state s_{ti} to $s_{(t+1)i}$. Both thermal comfort and energy consumption should be incorporated. The reward function is defined as follows:

$$R = -\sum_{i=1}^{N} \left(\rho Norm(|PMV_i|) + Norm(E_i)\right), \tag{2}$$

where E is heating and cooling energy consumption for each zone, we use Fanger's formula for the Predictive Mean Vote (PMV) [23] to estimate comfortable temperature bounds for the "standard" occupant within the current seasonal conditions, as defined by ASHRAE standard 55 [28]. The maximum high/low end of the comfort range for Class C environments has PMV values of +/- 0.7. ρ is used to balance the relative importance between energy consumption and thermal comfort. We use ρ = 4 during occupied periods and 0.1 during unoccupied periods since the range of human comfort and energy consumption is different during occupied and unoccupied periods. The reward evaluates the actions to meet the requirement of thermal comfort of all the occupants in the building. N is the number of zones. In the following sections, we will remove the i index for each zone to simplify the notation.

4.3 Learning the Building Dynamics

We require a parameterization of the building dynamics model that can cope with high-dimensional state and action spaces, and the complex dynamics of a multi-zone building. Therefore, we represent the dynamics function $\hat{f}_\theta(s_t, a_t)$ as a multi-layer neural network, parameterized by θ. This function outputs the predicted change in state that occurs as a result of executing action a_t from state s_t, over the time step duration of Δt. Thus, the predicted next state is given by: $\hat{s}_{t+1} = s_t + \hat{f}_\theta(s_t, a_t)$. While choosing too small of a Δt leads to too small of a state difference to allow meaningful learning, increasing the Δt too much can also make the learning process more difficult because it increases the complexity of the underlying continuous-time dynamics.

4.3.1 Environment-conditioned Neural Network Architecture. We define a neural network model $\hat{f}_\theta(s_t, a_t)$ for the building dynamics. In order to make the model achieve both good predictive accuracies and tractable computational optimization, we propose a simple and

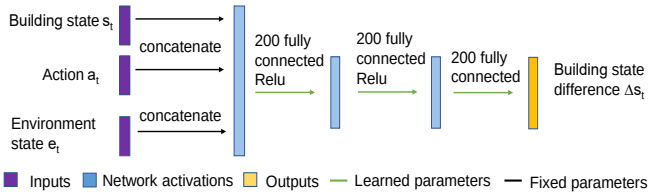

Figure 5: Environment-conditioned neural network for our Building Dynamics Model.

highly effective method for incorporating environment information. We formulate an environment-conditioned dynamics model $\hat{f}_\theta(s_t, a_t, e_t)$ that takes as input not only the current building state s_t and action a_t, but also the current environment state e_t. The model architecture is shown in Figure 5. The building state vector s_t, the action vector a_t and the environment state vector e_t are concatenated together and then are passed through two hidden layers and a final output layer. As opposed to a straightforward outputting of all the related states (building and environment), we produce a prediction of building state difference $\Delta\hat{s}_t$. This reduces the burden of the model to learn the changes in the environment that are not necessary. We provide the ground truth value for environment state, e.g., weather data and occupancy [21].

4.3.2 Weighted Ensemble Learning. As prior work [25, 26] has shown, capturing epistemic uncertainty in the network weights is important in model-based RL, especially with high capacity models that are liable to over-fit to the training set and extrapolate erroneously outside of it. To solve epistemic uncertainty, we propose a weighted ensemble learning algorithm, which approximates the posterior $p(\theta|D)$ with a set of M models, each with parameters θ_i. For deep models, it is sufficient to simply initialize each model θ_i with a different random initialization θ_i^0 and use different batches of data D_i at each training step.

We have M environmental-conditioned models. The input for all M models is the same and it includes the building and environment states and actions. To evaluate the performance of each model, we calculate the mean square error (MSE) of the past C timesteps (4 in our case) for each model compared to the ground truth for N states using Equation 3.

$$MSE = \sum_{i=1}^{C}\sum_{j=1}^{N} \phi^C \left| f_\theta\left(s_{i,j}, a_{i,j}\right) - \hat{f}_{true} \right|^2 \tag{3}$$

We introduce a temporal discount factor ϕ (0.9 in our case) that is used to evaluate how important past model error to the current model error. The temporal discount factor is a value between 0 and 1 since recent prediction cases are more important to the performance of current prediction. After we have the MSE for each model of past C timesteps, we first normalize the MSE to 0-1 scale. $Norm(x)$ is a normalization process, i.e., $Norm(x) = (x - x_{min})/(x_{max} - x_{min})$. Then we calculate the weight ratio W for all models by Equation 4.

$$W = \frac{1 - Norm(MSE_i)}{\sum_{i=1}^{M}(1 - Norm(MSE_i))} \tag{4}$$

The sum of all model's weight is 1. After that, we leverage Equation 5 to predict the next state.

$$s_{t+1} = \sum_{i=1}^{M} W_i f_{\theta_i}(s, a) \tag{5}$$

This allows our method to dynamically adjust the weights in aggregating the M models (M=5 in our case) during the prediction. As the states result in unequal prediction accuracy, our method is more robust against this variance.

4.4 Training the Building Dynamics Model

In this section, we illustrate how we pre-process training data, and train the proposed ENN model.

4.4.1 Data Collection. We collect the training dataset $D(s_t, a_t, s_{t+1})$ by executing the rule-based controller at each time step, and recording the resulting data $\tau = (s(0), a(0), s(1), a(1), ..., s(T-2), a(T-2), s(T-1)))$ of length T. We note that these data are very different from the data the controller will end up executing when planning with this learned dynamics model and a given reward function $r(s_t, a_t)$ (Section 4.5), showing the ability of model-based methods to learn from off-policy data.

4.4.2 Data Preprocessing. We slice the collected data $\{\tau\}$ into training data inputs (s_t, a_t) and corresponding output labels $s_{t+1} - s_t$. In building HVAC control, states can be temperature, humidity ratio, energy consumption, etc. These measurements have various ranges and the weights of the losses will be different if we feed the raw values directly to train the neural network model. Thus, we subtract the mean of the states/action and divide by the standard deviation $x' = \frac{x-\bar{x}}{\sigma(x)}$, where x stands for state or action.

4.4.3 Training the ENN Dynamics Model. ENN model consists of an ensemble of models. To make sure the models behave differently on the same dataset D, we randomly initialize model parameter $\theta_1, \theta_2, ..., \theta_M$ for all the dynamics models and use different batches of data D at each training step. We train the dynamics model $\hat{f}_\theta(s_t, a_t)$ using stochastic gradient descent [29] by minimizing the Mean Square Error (MSE) between predicted delta observation and ground truth delta observation as follows:

$$\varepsilon(\theta) = \frac{1}{D} \sum_{(s_t, a_t, s_{t+1}) \in D} \frac{1}{2} \| (s_{t+1} - s_t) - \hat{f}_\theta(s_t, a_t) \|^2 \quad (6)$$

We use 5-year weather data from Fresno, CA and Chicago, IL for the ENN model training and a completely different one-year for testing in this study. We provide the ENN model with ground truth information on future environment state, i.e. weather and occupancy [21]. In our implementation of ENN, we use the Adam optimizer [30] for gradient-based optimization with a learning rate of 10^{-3}. We train the ENN model with a batch size of 512 and a discount factor $\gamma = 0.99$. The number of epochs is 40. Each dynamics model consists of a neural network of two fully-connected hidden layers of size 200 with relu being nonlinear and a final fully-connected output layer. The weights and biases are initialized using the Xavier initialization process [31]. The number of samples for MPC controllers (RS, CEM, and MPPI) is 1000. The control cycle (timestep) is 15 minutes that have been used in classic control work [32]. We achieve convergence by 4.75×10^4 time-steps as explained in Section 5.3.1.

4.5 Online Control Action Planning

In our method, we use online planning with MPC to select actions via our model predictions. Given the building state s_t at time t, the prediction horizon H of the MPC controller, and an action sequence $a_{t:t+H} = \{a_t, ..., a_{t+H}\}$, the proposed ENN model $\hat{f}_\theta(s_t, a_t)$ produces a prediction over the resulting data $s_{t:t+H}$. At each time step t, the MPC controller applies the first action a_t of the sequence of optimized actions $A_t^H = \arg\max_{A_t^H} \sum_{t'=t}^{t+H-1} r\left(\hat{s}_{t'}, a_{t'}\right)$. We adopt the MPPI control method [13] to compute the optimal action sequence.

Model Predictive Path Integral (MPPI) Controller. MPPI control method has been applied to autonomously control a vehicle and get good performance. MPPI is an importance-sampling weighted algorithm and considers an update rule that more effectively integrates a larger number of samples into the distribution update. As derived by recent model-predictive path integral work [13], this general update rule takes the following form for time step t, from each of the K predicted trajectories:

$$a_t^{i+1} = a_t^i + \sum_{k=1}^{K} \omega(\varepsilon^k) \epsilon_t^k \quad (7)$$

Where ω is the importance-sampling weight for each trajectory and ϵ is the noise for exploration. The action for timesteps t of $(i+1)th$ trajectory is the sum of the action for timesteps t of ith trajectory and the noise-weighted average over sampled trajectories.

As shown in the algorithm 1, an initial control sequence is done either by initializing the input buffer with zeros or by using a secondary controller such as rule-based method and using its inputs as the initial control sequence. We first sample H noise from a normal distribution. Then, we compute K trajectories for H finite horizon with Brownian motion. For each trajectory generated, a cost is computed and stored in memory (line 2-7).

In model predictive control, optimization and execution take place simultaneously: a control sequence is computed, and then the first element of the sequence is executed. This process is repeated using the un-executed portion of the previous control sequence as the importance sampling trajectory for the next iteration. In order to ensure that at least one trajectory has non-zero mass (i.e., at least one trajectory has a lowest cost), we subtract the minimum cost of all the sampled trajectories from the cost function (line 9). Note that subtracting by a constant has no effect on the location of the minimum. In the second loop, we get the noise weighted average over K sampled trajectories (lines 10-11). The third loop computes an optimal input sequence using least cost of the trajectories for H finite horizons (lines 12-13). The top of the stack value is given to the actuators (line 14). After that, the whole input control sequence is left shifted by 1 (lines 15-16). To maintain the length of buffer, a_{init} is appended to the input control sequence (line 17). The states are then updated from the ENN model.

4.6 Putting It All Together

We summarize the working flow of MB^{2}C as follows. We first gather historical dataset D using a rule-based policy and randomly initialize model parameter $\theta_1, \theta_2, ..., \theta_M$ for ENN. Then we train the ENN model using this dataset by Equation 6. Finally, we deploy the learned ENN model and our MPPI controller in the real building for HVAC control.

For one control execution, we first obtain the current building state from sensors (e.g., zone temperature from a temperature

Algorithm 1: MPPI Controller

Input: ENN dynamics model $\hat{f}_\theta(s_t, a_t)$;
K: Number of samples, H: Length of horizon;
$(a_0, a_1, ...a_{H-1})$: Initial control sequence;
λ:Control hyper-parameter ;
Output: The control sequence $a_{t:t+H}$;

$s_0 \leftarrow GetStateEstimate()$;
for k = 0,1,...,K -1 **do**
  s $\leftarrow s_0$;
  Sample noise $\varepsilon^k = \{\epsilon_0^k, \epsilon_0^k, ...\epsilon_{H-1}^k\} \sim \mathbb{N}(\mu, \sigma)$;
  for t = 1,...,H **do**
    $s_t \leftarrow \hat{f}_\theta(s_{t-1}, a_{t-1} + \epsilon_{t-1}^k)$;
    $Cost(\varepsilon^k)$ += $-reward$ defined by equation 2 ;
$\beta \leftarrow min_k[Cost(\varepsilon^k)]$;
$\eta \leftarrow \sum_{k=0}^{K-1} exp(-\frac{1}{\lambda}(Cost(\varepsilon^k) - \beta))$;
for k = 0,1,...,K -1 **do**
  $\omega(\varepsilon^k) \leftarrow \frac{1}{\eta} exp(Cost(\varepsilon^k) - \beta)$;
for t = 0,1,...,H -1 **do**
  $a_t^* = a_t + \sum_{k=1}^{K} \omega(\varepsilon^k)\epsilon_t^k$;
SendToActuators(a_0);
for t = 0,1,...,H -1 **do**
  $a_{t-1} = a_t$;
$a_{t-1} = Initialize(a_{t-1})$;

sensor). After that, the best action sequence is sampled by MPPI controller with H horizon and the state is propagated by ENN model by solving the optimization problem defined in Equation 1. We execute the first action of the optimal action sequence in the building by setting corresponding actuators.

When MB^{2}C is running in the building, we can also collect building operation data, which is composed of control action execution records $D(s_t, a_t, s_{t+1})$, including current state, control action, and next state. We add the newly collected data into a sliding window for two months of data and train the ENN model from scratch again. We use a sliding window to adapt to the seasonality of the data, especially weather data. We randomly divide the training data set into a set of batch and update the weight through forward and backward propagation by feeding the data into the model. This process is called one epoch training after traversing all the batch of data. We will repeat this process for multiple epochs (40 in our current implementation) until the model converges. This is an iterative in-situ updating process to improve the accuracy of our building dynamic model.

5 EVALUATION

In this section, we conduct a variety of experiments in EnergyPlus to evaluate the performance of MB^{2}C and three baselines by a set of performance metrics.

5.1 Platform Setup

Building Example and its Dynamics Model in EnergyPlus In this work, we evaluate the performance of MB^{2}C in a building of 463 m^2 at Fresno, California. It is a single floor rectangular building of 5 thermal zones- 4 exterior zones, 1 interior zone. There are windows on all 4 facades. The HVAC system is single duct terminal reheat, which is composed by an Air Handler Unit (AHU) and Variable Air Volume (VAV) boxes. The AHU includes a fan, heating and cooling coils that can change the air's temperature. The VAV boxes take this pre-conditioned air from the main duct, heat it if necessary, and control the airflow provided to each zone.

Since we cannot conduct control experiments in the real building, we leverage a building model in EnergyPlus version 8.6 and conduct simulations with Typical Meteorological Year 3 (TMY3) weather data. In our implementation, the AHU set-point is set by default EnergyPlus control logic, and we only control the heating and cooling set-point in the VAV box.

EnergyPlus has been widely used to evaluate the HVAC control algorithm [6, 7, 10, 21]. There are four reasons why we choose EnergyPlus. First, we do not have one real building that allows us to conduct experiments. MB^{2}C could be deployed in a real building after we finish the ENN model training. Second, it is convenient to generate enough historical training data of rule-based method to train the ENN model. Third, in order to compare with a model-free DRL, we need a significant training data set to train these models since MFRL is not sample efficient. In our case, we need 5200 days (14+ years) of training data, which is unreasonable to obtain from real buildings. Finally, it is easy for us to evaluate the performance of different control algorithms under different locations, seasons and weather profiles.

MB^{2}C System Components As shown in Figure 4, MB^{2}C system includes two main parts: the building dynamics model ENN and the MPPI controller. We also need to store the newly collected building operation data for in-situ update of the building dynamics model. All these three components are all implemented in Tensorflow, which is an open-source machine learning library in Python. We use the building control virtual testbed (BCVTB) [33] for establishing a connection between EnergyPlus and MB^{2}C. We execute the control action by setting the temperature to a specific set point for each zone of our EnergyPlus building model during each control cycle.

5.2 Experiment Setting

We train ENN model based on the weather data from two different cities, Fresno, CA and Chicago, IL due to their distinct weather characteristics. The weather data for Fresno has intensive solar radiation and large variance in temperature, while Chicago is classified as hot-summer humid continental with four distinct seasons.

We compare MB^{2}C with the three baselines. We execute these four control methods to control the building HVAC system using the same weather data for simulation.

Rule-based Method: We implement a rule-based method according to our current campus building control policy for training data generation and comparison evaluation. We assign different zone temperature set-points. Each zone has a separate heating and cooling set-point. The heating set-point is set to 70 °F, and the

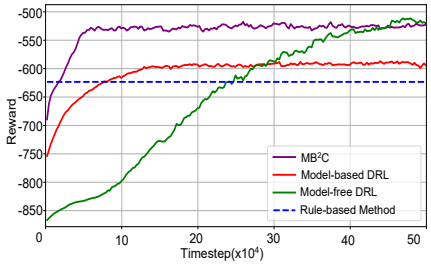

Figure 6: MB^{2}C Achieves both Data-Efficiency and High Performance.

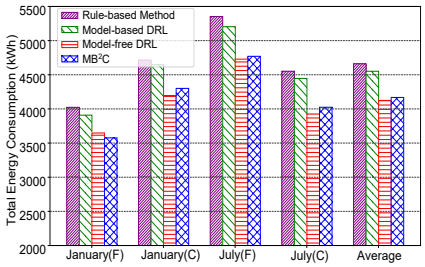

Figure 7: Energy Consumption of MB^{2}C and the Other Baselines.

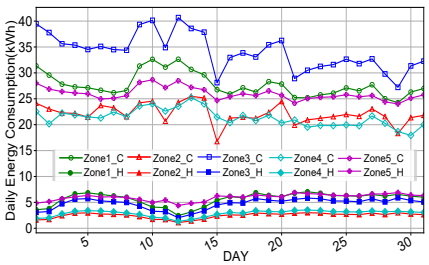

Figure 8: Daily Energy Consumption for Five Zones.

cooling set-point to 74 °F during the warm-up stage. The cooling set-point is limited between 72°F and 80°F, and the heating set-point is limited between 65°F and 72°F.

Model-free DRL: We implement Proximal Policy Optimization (PPO) [22] that is the default reinforcement learning algorithm at OpenAI because of its ease of use and good performance.

Model-based DRL with RS: For the conventional model-based method, we implement the deterministic neural network to model the building dynamics and RS method to choose the heating and cooling setpoints [10].

5.3 Experiment Results

We compare MB^{2}C with the above baselines by a set of performance metrics, including convergence analysis, energy efficiency and thermal comfort. We also study the performance of MB^{2}C, including its daily energy consumption for each zone, the performance gain of its key components, and its parameter setting.

5.3.1 Convergence Analysis. We first study the data efficiency of MB^{2}C and the other three baselines. For this study, we do not limit ourselves to a sliding window of two months for MB^{2}C, since the MFRL method requires copious amount of training data. Figure 6 shows that the accumulated reward of four control methods in each episode during a training process. One episode contains the data collected in one month, corresponding to 2976 time-step. We calculate the reward function every timestep. The reward in Figure 6 is the accumulated reward of one episode, i.e., the sum of the rewards of 2976 time-steps. From the results in Figure 6, we see that the episode reward increases and tends to be stable as the number of training episodes increases. When the episode reward does not change much, it means that we cannot do further to improve the learned control policy and thus the training process converges.

As indicated in Figure 6, MB^{2}C behaves better than rule-based method after the 1.75×10^4 time-steps. In this stage, the ENN model is first learned from off-line historical data. Then it can be deployed into real buildings and leverages the MPPI controller for exploration to further improve its performance. The model-based DRL and model-free DRL need 7.5×10^4 and 23.75×10^4 time-steps to behave better than rule-based method. MB^{2}C achieves 4.28× and 13.57× more data-efficient than model-based DRL and model-free DRL.

For convergence time, MB^{2}C converges faster than both model-based DRL and model-free DRL. MB^{2}C needs 4.75×10^4 and model-based DRL needs 11.5×10^4 time-steps. The model-free DRL needs 50×10^4 timesteps. MB^{2}C is 2.4× and 10.52× data-efficient than model-based DRL and model-free DRL with the same performance as model-free DRL.

5.3.2 Energy Efficiency. Figure 7 depicts the energy consumption results of four control methods. The results reveal that MB^{2}C saves 10.65% and 8.23% energy on average, compared with the rule-based method and model-based DRL. Compared with model-free DRL, MB^{2}C achieves comparable performance. MB^{2}C reduces the energy consumption of HVAC by modeling the complex building dynamics accurately and finding better heating and cooling setpoints.

We can also find that for different seasons and cities, the energy consumption is different. In Fresno, the building consumes 4770.04 kWh in July which is 33.39% more energy than that in January which consumes 3576.07 kWh. The reason is that in July, the outdoor air temperature range at Fresno is 15°C ~ 42°C. We have to keep cooling in daylight. However, in January, the outdoor air temperature range at Fresno -1°C ~ 18°C. This means that we can use outside air that is already in best range of thermal comfort to save energy.

In Chicago, the building consumes 4300.47 kWh in January that is 6.86% more energy than in July, because the weather is cold and the outdoor air temperature range in Chicago is -20°C ~ 15°C. In July, the outdoor air temperature range at Merced and Chicago is similar, 15°C ~ 42°C and 15°C ~ 40°C respectively. But the energy consumption in Fresno is 18.53% higher than the energy in Chicago. The reason is that the average day and night temperature difference for each day is larger than Chicago.

5.3.3 Thermal Comfort. Table 1 presents the average PMV value for all five zones in January and July under Fresno and Chicago weather data. All four control methods can maintain the PMV value in the desired range (-0.7~0.7) for most of the time. The average violation rate of model-based method is 1.97%, which is a little higher than the other three methods, because the controller tries random actions and some of the actions may lead to bad thermal comfort. MB^{2}C achieves a low average violation rate by leveraging more accurate ENN model and more effective MPPI controller.

5.3.4 Daily Energy Consumption for Five Zones. We analyze the daily energy consumption of MB^{2}C for five zones in July at Fresno. As shown in Figure 8, we record the heating energy and cooling energy for each zone per day. The top five hollow line symbols record the trend of cooling energy for five zones respectively. The bottom five solid line shows the trend of heating energy for five zones respectively. The energy spent by the third zone is higher

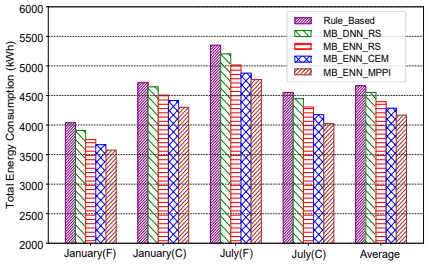

Figure 9: Energy Decomposition.

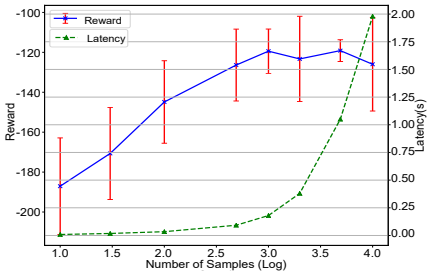

Figure 10: Samples of MPPI Controller.

Figure 11: Horizon of MPPI Controller.

Table 1: Thermal Comfort Statistical Results for Rule-based, Model-based, Model-free and MB^{2}C Schemes

Location	Comfort	Metric	Rule-based method		Model-based method		Model-free based method		MB^{2}C	
			January	July	January	July	January	July	January	July
Fresno	PMV	Mean	-0.36	-0.20	-0.32	-0.19	-0.11	-0.03	-0.04	0.13
		Std	0.26	0.36	0.31	0.34	0.15	0.18	0.11	0.14
		Violation rate	1.22%	1.51%	2.12%	1.71%	0	0.14%	0.40%	0.58%
Chicago	PMV	Mean	-0.17	-0.30	-0.26	-0.18	-0.25	0.07	-0.23	0.05
		Std	0.23	0.33	0.24	0.31	0.17	0.19	0.07	0.20
		Violation rate	1.20%	2.04%	1.9%	2.13%	0.95%	0	0.46%	1.23%

than the other zones, because the third zone is south-oriented and the sunlight hits into that zone most of the time.

We also see that both heating and cooling occurs in some days, because the day and night temperature difference is large. In the daylight, the average outdoor temperature is 38°C, and thus we need more energy for cooling. However, at night, the average outdoor temperature is 15°C, and thus we need some heating air to meet the minimum requirement of thermal comfort (in our simulations we assume an office-like environment with students working at night sometimes).

5.3.5 Performance Decomposition. We implement three versions of MB^{2}C with different control methods, i.e., RS (MB_ENN_RS), CEM (MB_ENN_CEM) and MPPI control method (MB_ENN_MPPI). We also compare with the rule-based method and the existing model-based DRL method (MB_DNN_RS).

For MB_ENN_CEM, we implement (Cross-entropy method) CEM [34] controller that begins as the RS method and does this sampling for multiple iterations $m \in \{0...M\}$ at each time step. The top J highest-scoring action sequences from each iteration are used to update and refine the mean and variance of the sampling distribution for the next iteration. After M iterations, the optimal heating and cooling actions are selected to be the resulting mean of the action distribution.

Figure 9 demonstrates the energy consumption of these four methods in two different months and at two different places (Fresno and Chicago). Compared with the rule-based method, MB_DNN_RS can only save 2.42% energy. When the building dynamics model in MB_DNN_RS changed to proposed model (MB_ENN_RS), 3.34% more energy can be saved, which illustrates the efficiency of proposed model. When we change the RS method to CEM method and MPPI method with the proposed model, 2.39% and 4.89 % more energy can be saved that illustrates efficiency of the MPPI controller.

5.3.6 Parameter Setting. MB^{2}C has two important parameters that may influence its performance.

The Number of Samples in the MPPI Algorithm. Figure 10 illustrates the performance of the MPPI controller as the number of sample trajectories is changed. We run MPPI controller with ground truth model to investigate the effect of different number of trajectories (10, 30, 100, 500, 1000, 2000, 5000, 10000). We ran 10 times to calculate the mean and standard reward for each number of trajectories. From Figure 10, we can see that the reward increases quickly as we increase the number of trajectories before 1000 trajectories (power of 3 in the figure). Then it increases slowly after 1000 trajectories, indicating that it is enough for the MPPI algorithm to converge. We also calculate the latency for making one action selection under different number of trajectories. we can see that the latency increases exponentially when trajectories increase. Thus we choose 1000 as the number of trajectories by considering the best reward and lower latency trade-off.

The Length of Horizon in the MPC Process. The horizon refers to the number of steps to look ahead in the MPC process. We investigate the effect of different length of Horizon H in Algorithm 1 to the performance of MPPI Controller. From Figure 11, we can see that the reward increases as the length of H increases and achieves the highest reward when H is 20. Then the reward decreases when we continue increasing the length of H. The reason is that small horizon results in more greedy actions that may not consider future dynamics. Large horizon produces worse actions since the prediction errors aggregate as the horizon becomes larger. We choose 20 for the horizon, which balances the prediction errors and action performance with short latency.

6 DISCUSSION

Building Model Calibration. Currently, we are leveraging the existing five-zone building model in EnergyPlus to evaluate all the existing control methods. We have not done the calibration for this building model since we have no historical operation data of that building. The buildings implemented in EnergyPlus are based on first principles thermodynamical models, so we expect

this model to be similar in performance to a real building. Moreover, it is reasonable to compare all the control methods based on the same building model implemented in EnergyPlus as ground truth. So, for the evaluation done in the paper, we believe this is a fair comparison to test the relative performance of different schemes for "a particular building". If the proposed MB^{2}C was to be deployed in a real building, we would first need to learn the dynamics model from the existing historical data from a real building. Then we deploy the model in the real building for control. If we were to do simulations to test MB^{2}C before real deployment, we need to develop a calibrated EnergyPlus model that matches the target building [6, 7].

Occupancy and Weather Model. In MB^{2}C, we provide the ground truth value of weather and occupancy for ENN dynamics model. MB^{2}C might be a bit more optimistic since we assume perfect prediction for the weather and occupancy. The errors in prediction may impact controller performance. However, we believe the performance will not significantly deviate from actual results considering model prediction errors. First is that the existing occupancy and weather prediction model [3, 4, 35, 36] show very small prediction error. Second is that MPPI controller outputs the optimal trajectory over the planning horizon. MPPI only takes the first optimal action and re-plans at the next time step based on new observations. This efficiently avoids compounding model error over time.

7 CONCLUSIONS

This paper proposes MB^{2}C, a novel model-based DRL HVAC control system for multi-zone buildings. We develop a new building dynamics model as an ensemble of multiple environment-conditioned neural network models. We also adopt a model predictive path integral control method to perform HVAC control. We compare the performance of MB^{2}C with the rule-based, and state-of-the-art model-based and model-free DRL schemes. The results show that MB^{2}C can achieve 10.65%, 8.23% energy savings on the former and comparable performance with the later, while maintaining (and sometimes even improving) thermal comfort of occupants. Perhaps more importantly, we can achieve this by significantly reducing the training set required by an order of magnitude (10.52× less).

8 ACKNOWLEDGMENTS

We would like to thank anonymous reviewers and our shepherd for their constructive comments and helpful suggestions. This material is based upon work partially supported by the National Science Foundation under grants #CCF-2008837, and a 2020 Seed Fund award from Tecnológico de Monterrey & CITRIS and the Banatao Institute at the University of California.

REFERENCES

[1] Ltd. DR International.2012. 2011 building energy data book. https://openei.org/doe-opendata/dataset/buildings-energy-data-book.
[2] Jyri Salpakari and Peter Lund. Optimal and rule-based control strategies for energy flexibility in buildings with pv. *Applied Energy*, 161:425–436, 2016.
[3] Alex Beltran and Alberto E Cerpa. Optimal hvac building control with occupancy prediction. In *ACM BuildSys*, 2014.
[4] Daniel A Winkler, Ashish Yadav, Claudia Chitu, and Alberto E Cerpa. Office: Optimization framework for improved comfort & efficiency. In *ACM/IEEE IPSN*, 2020.
[5] Narendra N Kota, John M House, Jasbir S Arora, and Theodore F Smith. Optimal control of hvac systems using ddp and nlp techniques. *Optimal Control Applications and Methods*, 17(1):71–78, 1996.
[6] Xianzhong Ding, Wan Du, and Alberto Cerpa. Octopus: Deep reinforcement learning for holistic smart building control. In *ACM BuildSys*, 2019.
[7] Zhiang Zhang and Khee Poh Lam. Practical implementation and evaluation of deep reinforcement learning control for a radiant heating system. In *ACM BuildSys*, 2018.
[8] Zoltan Nagy, June Y Park, and J Vazquez-Canteli. Reinforcement learning for intelligent environments: A tutorial. *Handbook of Sustainable and Resilient Infrastructure*, 2018.
[9] June Young Park and Zoltan Nagy. Hvaclearn: A reinforcement learning based occupant-centric control for thermostat set-points. In *ACM e-Energy*, 2020.
[10] Chi Zhang, Sanmukh R Kuppannagari, Rajgopal Kannan, and Viktor K Prasanna. Building hvac scheduling using reinforcement learning via neural network based model approximation. In *ACM BuildSys*, 2019.
[11] Siddharth Goyal and Prabir Barooah. A method for model-reduction of non-linear thermal dynamics of multi-zone buildings. *Energy and Buildings*, 2012.
[12] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *IEEE ICRA*, 2018.
[13] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *IEEE ICRA*, 2017.
[14] Tong Wu and Jorge Ortiz. Towards adaptive anomaly detection in buildings with deep reinforcement learning. In *ACM BuildSys*, 2019.
[15] Bharathan Balaji, Sunil Mallya, et al. Deepracer: Autonomous racing platform for experimentation with sim2real reinforcement learning. In *IEEE ICRA*, 2020.
[16] Francesco Fraternali, Bharathan Balaji, Yuvraj Agarwal, and Rajesh K Gupta. Aces: Automatic configuration of energy harvesting sensors with reinforcement learning. *ACM TOSN*, 2020.
[17] Zhihao Shen, Wan Du, Xi Zhao, and Jianhua Zou. Dmm: fast map matching for cellular data. In *ACM MobiCom*, 2020.
[18] Zhihao Shen, Kang Yang, Wan Du, Xi Zhao, and Jianhua Zou. Deepapp: A deep reinforcement learning framework for mobile application usage prediction. In *ACM SenSys*, 2019.
[19] Zhi Cao, Honggang Zhang, Yu Cao, and Benyuan Liu. A deep reinforcement learning approach to multi-component job scheduling in edge computing. In *IEEE MSN*, 2019.
[20] Miaomiao Liu, Xianzhong Ding, and Wan Du. Continuous, real-time object detection on mobiledevices without offloading. In *IEEE ICDCS*, 2020.
[21] Bingqing Chen, Zicheng Cai, and Mario Bergés. Gnu-rl: A precocial reinforcement learning solution for building hvac control using a differentiable mpc policy. In *ACM BuildSys*, 2019.
[22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
[23] Poul O Fanger et al. Thermal comfort. analysis and applications in environmental engineering. *Thermal comfort. Analysis and applications in environmental engineering.*, 1970.
[24] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017.
[25] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.
[26] 2019 Sergey Levine. Model-based reinforcement learning. http://rail.eecs.berkeley.edu/deeprlcourse/.
[27] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *CoRL*, 2020.
[28] A. Standard. Standard 55-2004-thermal environmental conditions for human occupancy. *ASHRAE Inc*, 2004.
[29] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.
[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
[31] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.
[32] Mesut Avci, Murat Erkoc, Amir Rahmani, and Shihab Asfour. Model predictive hvac load control in buildings using real-time electricity pricing. *Energy and Buildings*, 2013.
[33] Michael Wetter. Co-simulation of building energy and control systems with the building controls virtual test bed. *Journal of Building Performance Simulation*, 2011.
[34] Zdravko I Botev, Dirk P Kroese, Reuven Y Rubinstein, and Pierre L'Ecuyer. The cross-entropy method for optimization. In *Handbook of statistics*. Elsevier, 2013.
[35] Claudia Chiţu, Grigore Stamatescu, and Alberto Cerpa. Building occupancy estimation using supervised learning techniques. In *IEEE ICSTCC*, 2019.
[36] Grigore Stamatescu, Alex Beltran, and Alberto Cerpa. Data-driven comfort models for user-centric predictive control in smart buildings. In *ACM BuildSys*, 2016.