frontiers in Built Environment

ORIGINAL RESEARCH
published: 13 November 2020
doi: 10.3389/fbuil.2020.562239

Gnu-RL: A Practical and Scalable Reinforcement Learning Solution for Building HVAC Control Using a Differentiable MPC Policy

Bingqing Chen[1]*, *Zicheng Cai*[2] *and Mario Bergés*[1]

[1] *Department of Civil and Environmental Engineering, Carnegie Mellon University, Pittsburgh, PA, United States,* [2] *Walker Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX, United States*

We propose Gnu-RL: a novel approach that enables real-world deployment of reinforcement learning (RL) for building control and requires no prior information other than historical data from existing Heating Ventilation and Air Conditioning (HVAC) controllers. In contrast with existing RL agents, which are opaque to expert interrogation and need ample training to achieve reasonable control performance, Gnu-RL is much more attractive for real-world applications. Furthermore, Gnu-RL avoids the need to develop and calibrate simulation environments for each building or system under control, thus making it highly scalable. To achieve this, we bootstrap the Gnu-RL agent with domain knowledge and expert demonstration. Specifically, Gnu-RL adopts a recently-developed Differentiable Model Predictive Control (MPC) policy, which encodes domain knowledge on planning and system dynamics, making it both sample-efficient and interpretable. Prior to any interaction with the environment, a Gnu-RL agent is pre-trained on historical data using imitation learning, enabling it to match the behavior of the existing controller. Once it is put in charge of controlling the environment, the agent continues to improve its policy end-to-end, using a policy gradient algorithm. We evaluate Gnu-RL in both simulation studies and a real-world testbed. Firstly, we validated the Gnu-RL agent is indeed practical and scalable by applying it to three commercial reference buildings in simulation, and demonstrated the superiority of the Differentiable MPC policy compared to a generic neural network. In another simulation experiment, our approach saved 6.6% of the total energy compared to the best published RL result for the same environment, while maintaining a higher level of occupant comfort. Finally, Gnu-RL was deployed to control the airflow to a real-world conference room with a Variable-Air-Volume (VAV) system during a 3-weeks period. Our results show that Gnu-RL reduced cooling demand by 16.7% compared to the existing controller and tracked the temperature set-point better.

Keywords: reinforcement learning, model predictive control, imitation learning, building control, heating, ventilation, air conditioning (HVAC)

OPEN ACCESS

Edited by:
Farrokh Jazizadeh, Virginia Tech, United States

Reviewed by:
Qinglong Meng, Chang'an University, China
Jorge Ortiz, Rutgers, The State University of New Jersey, United States

***Correspondence:**
Bingqing Chen
bingqinc@andrew.cmu.edu

Specialty section:
This article was submitted to Structural Sensing, a section of the journal Frontiers in Built Environment

Received: *14 May 2020*
Accepted: *17 September 2020*
Published: *13 November 2020*

Citation:
Chen B, Cai Z and Bergés M (2020) Gnu-RL: A Practical and Scalable Reinforcement Learning Solution for Building HVAC Control Using a Differentiable MPC Policy. Front. Built Environ. 6:562239. doi: 10.3389/fbuil.2020.562239

1. INTRODUCTION

Advanced control strategies for operating heating, ventilation, and air conditioning (HVAC) systems have the potential to significantly improve the energy efficiency of our building stock and enhance comfort (Bengea et al., 2012; Li and Wen, 2014; Goetzler et al., 2017). However, the majority of HVAC systems today are still operated by simple rule-based and feedback controls, such as *on-off* control or proportional-integral-derivative (PID) control. These prescriptive and reactive control strategies do not take into consideration predictive information on disturbances, such as weather and occupancy (Killian and Kozek, 2016), making their energy performance sub-optimal.

Optimal control strategies, such as model predictive control (MPC) address these drawbacks by iteratively optimizing an objective function over a receding time horizon. However, despite many successful applications of MPC (e.g., Prívara et al., 2011; Aswani et al., 2012; Maasoumy et al., 2014), its widespread adoption has been limited by the need of accurate models (Privara et al., 2013; Killian and Kozek, 2016). This is especially challenging because buildings are heterogeneous (Lü et al., 2015), e.g., they have different layouts, HVAC configurations, and occupancy patterns. Thus, custom models are required for each thermal zone or building under control, limiting the scalability of MPC. Another drawback is that MPC treats modeling and planning as two separate tasks. Furthermore, the quality of the model is evaluated by criteria, such as prediction error, which may or may not lead to good control performance[1].

Reinforcement learning (RL), on the other hand, is a learning-based approach that adapts to different environments by learning a control policy through direct interaction with the environment (Sutton and Barto, 2018). RL has achieved remarkable success in mastering many complex tasks, such as playing Go (Moyer, 2016) and StarCraft (Vinyals et al., 2017). It has also been shown to be a feasible approach for optimal control of HVAC systems (Liu and Henze, 2006; Dalamagkidis et al., 2007). However, the flexibility of the RL framework comes at the cost of increased sample complexity (Kakade, 2003). Taking recent published results as examples, an RL agent may need 5 million interaction steps (equivalent to 47.5 years in simulation) to achieve the same performance as a feedback controller on an HVAC system (Zhang and Lam, 2018). Even after pretraining on expert demonstration, an RL agent may still require an additional year of training in simulation to achieve similar performance to a baseline controller (Jia et al., 2019). The long training time may explain the limited number of real-world applications of RL for HVAC control. The studies that did validate their RL agents in real-world testbeds (Liu and Henze, 2006; Zhang and Lam, 2018) depended on high-fidelity models to train their agents in simulation first. Aside from the common problems with sim-to-real transfer (Peng et al., 2018), such approach requires a high-fidelity model for each building and thus shifts the focus back to modeling (or model calibration) (Yang et al., 2015).

To overcome these limitations, we propose Gnu-RL[2]: an RL agent that exhibits a high level of independence and maturity at the onset of its learning phase (i.e., a precocial agent) and requires no prior information other than historical data from existing controllers. As shown in **Figure 1**, the Gnu-RL agent leverages a recently-developed Differentiable MPC policy (Amos et al., 2018) that encodes domain knowledge on system dynamics and control. We do away with the need for high-fidelity models by initializing the Gnu-RL agent on historical data from the existing controller. By imitating the existing controller, the Gnu-RL agent behaves similarly to it prior to any interaction with the environment. Once deployed, it continues to improve its policy end-to-end using a policy gradient algorithm.

Firstly, we validated our approach is both practical and scalable by applying it to three different buildings, the `warehouse`, the `small office`, and the `medium office` from the Department of Energy (DOE) commercial reference buildings (Deru et al., 2011). We also demonstrated the superiority of the Differentiable MPC policy compared to a generic neural network. Then, we evaluated the performance of Gnu-RL in detail in both a simulation study (an EnergyPlus model of a 600 m^2 office building from Zhang and Lam, 2018) and a real-world deployment (a 20 m^2 conference room). First, Gnu-RL was directly deployed to control the simulation environment after offline pretraining on state-action pairs generated by a P-controller. In this setting, our algorithm saved 6.6% of the total energy compared to the best-performing RL agent in Zhang and Lam (2018), while maintaining a higher level of occupant comfort. Next, we deployed Gnu-RL in a real conference room for a 3-weeks period. The agent was pre-trained on historical data from the Building Automation System (BAS). Here, Gnu-RL saved 16.7% of cooling demand compared to the existing control strategy based on a fixed schedule, while achieving better set-point tracking.

This paper is an extension of our prior work (Chen et al., 2019). In this paper, we provided further validation of our approach on three new buildings (section 5), with larger state-action space compared to the environments used in our prior work.

The rest of the paper is organized as follows. Section 2 reviews related work on MPC and RL for HVAC control. Section 3 provides technical background on the techniques used by Gnu-RL, and section 4 formally introduces Gnu-RL. We then describe experiments and results both in simulation studies (sections 5 and 6) and in a real-world testbed (section 7), and present conclusions and directions for future work in section 8.

2. RELATED WORK

Because Gnu-RL updates a Differentiable MPC policy with a RL algorithm, it can be seen as a hybrid of MPC and RL. Thus,

[1] We elaborate on this issue in sections 6.3 and 6.4.

[2] The name Gnu comes from drawing an analogy between RL agents and animals. Gnus are among the most successful herbivores in the African savanna, and part of their success can be attributed to the precociality of their youngsters who are able to outrun predators within a day after their birth. It is also worth mentioning, that the name has no connection to the GNU operating system.

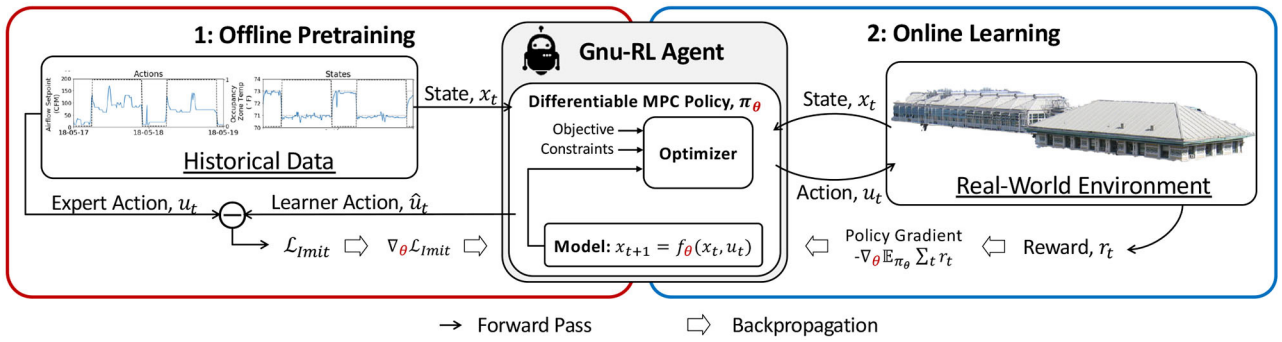

FIGURE 1 | Framework. The Gnu-RL agent utilizes a Differentiable MPC policy, which encodes domain knowledge on planning and system dynamics. In the offline pretraining phase, the agent is initialized by imitating historical data from the existing controller. In the online learning phase, the agent continues to improve its policy end-to-end using a policy gradient algorithm.

in this section we review both MPC and RL approaches to HVAC control.

2.1. Model Predictive Control for HVAC

Model predictive control is a planning-based method that solves an optimal control problem iteratively over a receding time horizon. Some of the advantages of MPC are that it takes into consideration future disturbances and that it can handle multiple constraints and objectives, e.g., energy and comfort (Killian and Kozek, 2016).

However, it can be argued that the main roadblock preventing widespread adoption of MPC is its reliance on a model (Privara et al., 2013; Killian and Kozek, 2016). By some estimates, modeling can account for up to 75% of the time and resources required for implementing MPC in practice (Rockett and Hathway, 2017). Because buildings are highly heterogeneous (Lü et al., 2015), a custom model is required for each thermal zone or building under control.

Privara et al. (2013) identified two paradigms for modeling building dynamics: physics-based and statistics-based. Physics-based models, e.g., EnergyPlus, utilize physical knowledge and material properties of a building to create detailed representation of the building dynamics. A major shortcoming is that such models are not control-oriented (Atam and Helsen, 2016). Nonetheless, it is not impossible to use such models for control. For instance, Zhao et al. (2013) used exhaustive search optimization to derive control policy for an EnergyPlus model. Furthermore, physics-based model requires significant modeling effort, because they have a large number of free parameters to be specified by engineers (e.g., 2,500 parameters for a medium-sized building; Karaguzel and Lam, 2011), and information required for determining these parameters are scattered in different design documents (Gu et al., 2014). Statistical models assume a parametric model form, which may or may not have physical underpinnings, and identifies model parameters directly from data. While this approach is potentially scalable, a practical problem is that the experimental conditions required for accurate identification of building systems fall outside of normal building operations (Agbi et al., 2012). Alternatively, excitation signals from actuators may be used to help identify model parameters (Prívara et al., 2011; Agbi et al., 2012; Aswani et al., 2012), which requires careful design of experiments (Cai et al., 2016) and may disturb normal operation. Even then, it is still difficult to identify some parameters with supposedly rich excitation signals (Agbi et al., 2012).

To add to the challenge, there are many sources of stochasticity and uncertainty in building dynamics (Maasoumy et al., 2014). Learning-based MPC (Aswani et al., 2012) accounts for the stochasticity from internal thermal gains by modeling the building dynamics in a semi-parametric form, where the zone temperature evolves following a linear model and the internal thermal gain is learned from data as a non-parametric term. Learning-based MPC also decouples robustness and performance by using a statistical model to optimize performance and a deterministic model to impose comfort constraints. Adaptive MPC attempts to address the uncertainty by updating model parameters online with new observations. Here the objective is to estimate model parameters that minimize the difference between prediction and observation over time. Examples of this line of work include the use of Extended/Unscented Kalman Filter (Fux et al., 2014; Maasoumy et al., 2014) to simultaneously estimate states and model parameters. More recently, Yuan et al. (2020) developed an adaptive MPC controller that learns a probabilistic occupancy distribution based on fine-grained occupancy counts collected from real buildings.

2.2. Reinforcement Learning for HVAC

The early works by Liu and Henze (2006) and Dalamagkidis et al. (2007) demonstrated the potential of using RL for optimal HVAC controls. 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 (Liu and Henze, 2006). In **Figure 2**, we summarized the training time reported in Liu and Henze (2006), Dalamagkidis et al. (2007), Yang et al. (2015), Li et al. (2017), Wei et al. (2017), Zhang and Lam (2018), Gao et al. (2019), and Jia et al. (2019). We acknowledge the limitations of the summary. Firstly, the RL agents were evaluated in different

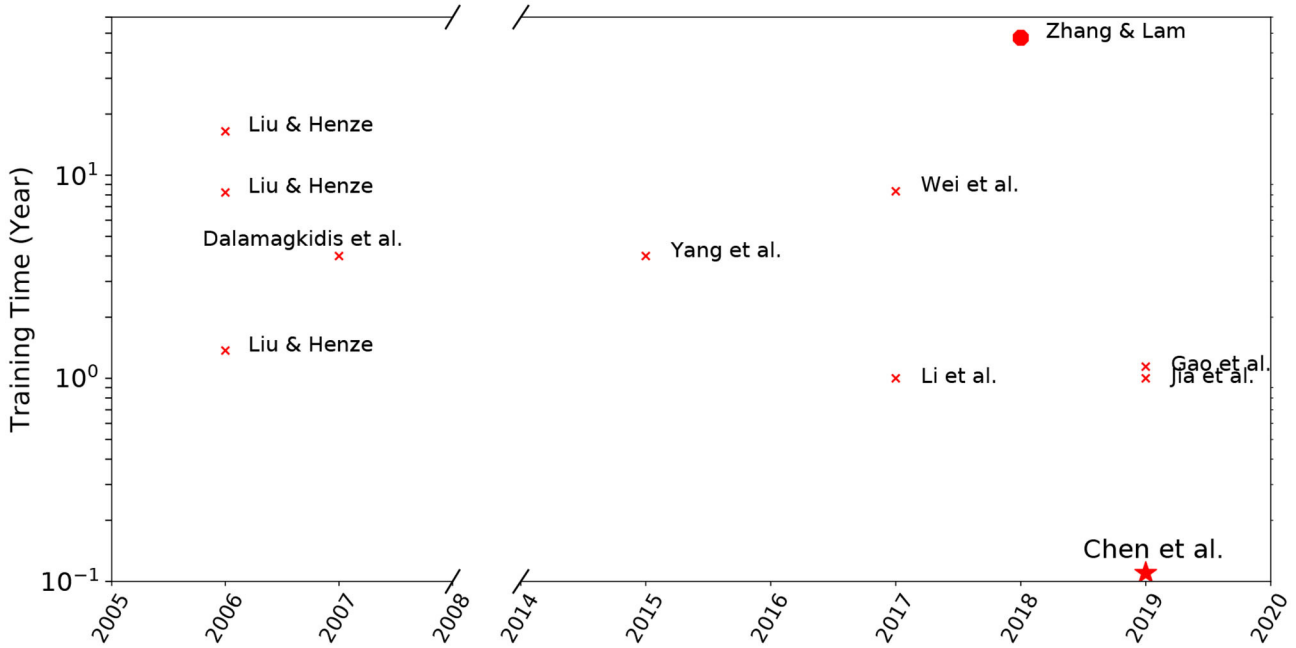

FIGURE 2 | Summary of training time reported in the literature; By incorporating domain knowledge and expert demonstration, our proposed Gnu-RL agent drastically reduced training time compared to existing works.

environments of varying complexity, making direct comparison impossible. We only considered works that evaluated their agents in physics-based emulators, including EnergyPlus, TRNSYS, and MATLAB Simulink. Another limitation is that most of these papers reported the total training time. A more meaningful evaluation metric may be the amount of time required to reach the same performance as a baseline controller. Despite these limitations, the observation here is that the amount of time used for training is typically in the order of years. In our prior work (Chen et al., 2019), we proposed Gnu-RL, a precocial agent that is capable of controlling HVAC system well at "birth." We also want to draw the readers' attention to Zhang and Lam (2018), to which we benchmarked our approach in section 6.

The long training time may explain why applications beyond simulation are numbered, despite the numerous publications on this topic. While Liu and Henze (2006) and Zhang and Lam (2018) validated their RL agents in real-world testbeds, they both assumed the existence of high-fidelity models for training their agents in simulation first. Such approach depends on a high-fidelity model for each building and thus shifts the focus back to modeling (Yang et al., 2015). Costanzo et al. (2016) also reported a real-world deployment, but provided no quantitative evaluation. Google announced a 40% energy consumption reduction in their data centers (Evans and Gao, 2016), but to our knowledge no technical publication of this achievement is available yet.

To reduce sample complexity, researchers have experimented with different approaches. Li et al. (2017) and Jia et al. (2019) initialized their agents with historical data from the existing controller. Li et al. (2017) populated the replay memory with historical data. Similar to our approach, Jia et al. (2019) cloned the behavior of the existing controller based on historical data. Furthermore, Jia et al. (2019) proposed to use the expert policy, i.e., the policy from the existing controller, as a baseline to reduce variance. Costanzo et al. (2016) augmented the replay memory with additional state-action pairs using a neural network. Chen et al. (2018) and Nagy et al. (2018) incorporated a model of system dynamics into RL. Nagy et al. (2018) trained a neural network as a model for state transitions and incorporated multi-step planning into RL. The experiments in Nagy et al. (2018) showed that model-based RL was generally superior to model-free RL in terms of sample efficiency, energy performance, and occupants' comfort. In comparison, Chen et al. (2018) used a special input-convex neural network (Amos et al., 2017) to model the system dynamics. With such a model choice, multi-step planning over the complex neural network dynamics is a convex problem and could be solved optimally with gradient descent. Chen et al. (2018) validated their work in the `large office` from DOE commercial reference buildings, controlling the temperature setpoints of 16 thermal zones. In fact, to our knowledge, this is the largest validation on this topic in terms of the state-action space. While their approach showed significant energy savings, no evaluation on thermal comfort was provided.

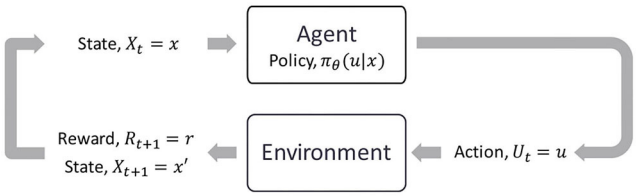

FIGURE 3 | The agent-environment interaction in a Markov Decision Process.

3. BACKGROUND

We now present background technical concepts used by Gnu-RL.

3.1. Reinforcement Learning

RL learns an optimal control policy through direct interaction with the environment. The optimal control problem may be formulated as a Markov Decision Process[3] (MDP), as shown in **Figure 3**. At each time step t, the agent selects an action $U_t = u$ given the current state $X_t = x$ based on its policy π_θ (Equation 1a). In modern RL, the policy is commonly approximated by a neural network parameterized by θ (Mnih et al., 2013). Using a neural network as a function approximator allows one to handle large state-action space and generalize from past experiences (Sutton and Barto, 2018). When the agent takes the action u, there is a state transition $X_{t+1} = x'$ based the environment dynamics f (Equation 1b) and the agent receives a reward $R_{t+1} = r$. In our problem formulation, the reward is the negative of a cost function (Equation 1c). The objective of RL is to find a policy π_θ that maximizes the expected total reward or, equivalently, minimizes the expected total cost.

$$u \sim \pi_\theta(u|x) = \mathbb{P}(U_t|X_t = x;\ \theta) \tag{1a}$$

$$x' \sim f(x,u) = \mathbb{P}(X_{t+1}|X_t = x, U_t = u) \tag{1b}$$

$$r = -C(x,u) \tag{1c}$$

RL methods can be categorized as model-free and model-based based on whether a model of the environment is used. There are three common approaches to model-free RL, i.e., value-based methods, policy gradient methods, and actor-critic methods. Valued-based methods, e.g., Q-learning and its variants, learns the policy indirectly by learning a value function, e.g., $Q_\pi(x,u)$ (Equation 2a) or $V_\pi(x)$ (Equation 2b), and takes the action that maximizes the value function with exploration. γ is the discount factor. Alternatively, one may use the advantage function, $A_\pi(x,u)$ as given in Equation (2c), which could be interpreted as how much a given action improve upon the policy's average behavior. The value function may be updated via methods, such as Bellman backup (Sutton and Barto, 2018). Value-based methods have been applied to HVAC control in works, such as Liu and Henze (2007), Dalamagkidis et al. (2007), Yang et al. (2015), and Costanzo et al. (2016). A major shortcoming of Q-learning is that it is only applicable to problems with discrete action spaces. Thus, for problems with continuous action spaces, i.e., a large portion of HVAC control problems, each continuous action need to discretized into a number of discrete actions, which needlessly enlarge the action space.

$$Q_{\pi_\theta}(x,u) = \mathbb{E}_{\pi_\theta}\big[\sum_{k=0}^{\infty}\gamma^k r_{t+k}|x_t = x, u_t = u\big] \tag{2a}$$

$$V_{\pi_\theta}(x) = \mathbb{E}_{\pi_\theta}\big[\sum_{k=0}^{\infty}\gamma^k r_{t+k}|x_t = x\big] \tag{2b}$$

$$A_{\pi_\theta}(x,u) = Q_{\pi_\theta}(x,u) - V_{\pi_\theta}(x) \tag{2c}$$

Policy gradients methods directly search for an optimal policy π^*_θ, using stochastic estimates of policy gradients, and are applicable to problems with continuous action spaces. In this work, we used Proximal Policy Optimization (PPO) (Schulman et al., 2017) as our online learning algorithm, which is elaborated in section 3.2. Actor-critic methods, e.g., Advantage Actor-Critic (A2C), Asynchronous Advantage Actor-Critic (A3C) (Mnih et al., 2016), and Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015), are hybrids of the value-based and policy gradient approaches. Actor-critic methods use a policy network to select actions (the actor), and a value network to evaluate the action (the critic). Actor-critic methods have been used for HVAC control in Li et al. (2017), Zhang and Lam (2018), and Gao et al. (2019). A2C and A3C use a Q-network as critic and thus are only applicable to problems with discrete action spaces, i.e., the same drawback as Q-learning. DDPG, on the other hand, is applicable to problems with continuous action spaces. Although we classified PPO as a policy gradient method, one can and should incorporate a critic for variance reduction and more robust performance.

Alternatively, one can incorporate a model of the environment to improve the sample efficiency. Previously, it was believed that model-based RL could not perform as well as model-free RL asymptotically. But, recent work (Chua et al., 2018) has shown that model-based RL can match the asymptotic performance of model-free RL algorithms, while being significantly more sample efficient. A common idea for model-based RL is to simultaneously learn a model of the environment and plan ahead based on the learned model. The classical Dyna-Q (Sutton and Barto, 2018) is an example of such approach. Developing upon the idea, people have modeled the environment with Gaussian Process (Kamthe and Deisenroth, 2017), locally linear models (Watter et al., 2015), and neural networks (Chua et al., 2018).

3.2. Proximal Policy Optimization

As discussed earlier, policy gradient methods directly optimize the policy π_θ to maximize the expected total reward (Equation 3a). To do that, these methods compute an estimate of the policy gradient defined in Equation (3b) and optimize

[3]While Richard Bellman's notation is more commonly seen in RL literature (i.e., s-states, a-actions, r-rewards), we adopt Lev Pontryagin's notation (i.e., x-states, u-actions, c-costs) to be consistent with classical control literature.

the objective with stochastic gradient ascent (Equation 3c). Aside from the obvious benefit of being able to handle continuous action spaces, policy gradients methods have several advantages over value-based ones, as discussed in Sutton and Barto (2018). Firstly, policy gradient methods can learn both deterministic and stochastic policies, while there is no natural way to learn stochastic policy with value-based methods. Secondly, the policy may be a simpler function to approximate, and thus policy-based methods typically learn faster and yield a superior asymptotic policy. Finally, the choice of policy parameterization is a natural way to inject domain knowledge into RL.

$$\theta^* = \arg\max_\theta \mathbb{E}_{\pi_\theta}\Big[\sum_{k=0}^{\infty}\gamma^k r_{t+k}\Big] \tag{3a}$$

$$g \coloneqq \nabla_\theta \mathbb{E}_{\pi_\theta}\Big[\sum_{k=0}^{\infty}\gamma^k r_{t+k}\Big] \tag{3b}$$

$$\theta \leftarrow \theta + \alpha\hat{g} \tag{3c}$$

A variety of policy gradient algorithms have been proposed in the literature. Perhaps, REINFORCE (Sutton and Barto, 2018) is the most well-known one. However, it suffers from large performance variance and unstable policy updates (Schulman et al., 2017). PPO (Schulman et al., 2017) is the most recent work in a line of research that improved upon vanilla policy gradient methods, including Natural Policy Gradients (Kakade, 2002) and Trust Region Policy Optimization (TRPO) (Schulman et al., 2015a). The intuition behind these methods is that in each update the policy π_θ should not change too much. This idea is most clear with the objective of TRPO (Equation 4), which is to maximize an importance weighted advantage estimate, subject to a constraint on the size of the policy update.

$$\begin{aligned} \max_\theta \quad & \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(u_t|x_t)}{\pi_{\theta_{old}}(u_t|x_t)}\hat{A}_t\right] \\ \text{s.t.} \quad & \hat{\mathbb{E}}_t\left[\mathrm{KL}\left[\pi_\theta(\cdot|x_t), \pi_{\theta_{old}}(\cdot|x_t)\right]\right] \le \delta \end{aligned} \tag{4}$$

However, it is not straightforward to solve the optimization problem posed in Equation (4), due to the constraint. PPO simplified the problem using a surrogate objective, given in Equation (5a). $w_t(\theta)$ denotes the importance weighting of the policy after and before the update, i.e., $\frac{\pi_\theta(u_t|x_t)}{\pi_{\theta_{old}}(u_t|x_t)}$, and ϵ is a hyperparameter. For ease of notation in future discussion, we denote the negative of the objective as $\mathcal{L}_{PPO}$ (Equation 5b). PPO is known to be stable and robust to hyperparameters and network architectures (Schulman et al., 2017). It was also shown to outperform methods, such as A3C (Mnih et al., 2016) and TRPO (Schulman et al., 2015a).

$$\max_\theta \quad \hat{\mathbb{E}}_t\left[\min\left(w_t(\theta)\hat{A}_t, \mathrm{clip}(w_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \tag{5a}$$

$$\mathcal{L}_{PPO}(\theta) = -\hat{\mathbb{E}}_t\left[\min\left(w_t(\theta)\hat{A}_t, \mathrm{clip}(w_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \tag{5b}$$

There are a number of options one can use for the advantage estimate $\hat{A}_t$, including total rewards (Equation 6a), Q-function, advantage function, and k-step TD residual (Equation 6b). Schulman et al. (2015b) provides a thorough discussion on possible options and the bias-variance trade-off of these options.

$$\hat{A}_{t,R} = \sum_{k=0}^{\infty}\gamma^k r_{t+k} \tag{6a}$$

$$\hat{A}_{t,TD} = -V_\pi(x_t) + \sum_{l=0}^{k-1}\gamma^l r_{t+l} + \gamma^k V_\pi(x_{t+k}) \tag{6b}$$

3.3. Differentiable MPC

Since the success of Mnih et al. (2013), it is common to approximate the policy $\pi_\theta(u|x)$ with a neural network. However, a generic neural network does not encode any domain knowledge on planning or system dynamics, which is abundant in existing HVAC control literature. We hypothesized that encoding such knowledge in the policy would expedite the learning process.

In this paper, we took advantage of a newly developed Differentiable MPC policy (Amos et al., 2018) in place of a neural network. In the forward pass, the Differentiable MPC policy solves a box-constrained linear-quadratic regulator (LQR) problem given in Equation (7). Specifically, it finds the optimal trajectory, $\tau^*_{1:T} = \{x^*_t, u^*_t\}_{1:T}$, which minimizes the quadratic cost function over the planning horizon $1:T$ (Equation 7a) and satisfies a linear model of the system dynamics (Equation 7b). Furthermore, the Differentiable MPC implementation allows one to incorporate box constraints on the control action (Equation 7c). It is also possible to use the Differentiable MPC policy for non-quadratic cost and non-linear dynamics with local linear quadratic approximation.

$$\tau^*_{1:T} = \arg\min_{\tau_{1:T}} \sum_t \frac{1}{2}\tau_t^T C_t \tau_t + c_t^T \tau_t \tag{7a}$$

$$\text{s.t.} \quad x_1 = x_{\text{init}},\ x_{t+1} = F_t\tau_t + f_t \tag{7b}$$

$$\underline{u} \le u \le \bar{u} \tag{7c}$$

In the backward pass, the differentiable nature of the policy allows us to update the model parameters end-to-end. The learnable parameters are $\{C, c, F, f\}$. The derivatives of the loss with respect to the model parameters can be obtained by differentiating the Karush-Kuhn-Tucker (KKT) conditions of the problem given in Equation (7), using the techniques developed in Amos and Kolter (2017). While the LQR problem (Equation 7) is convex, optimizing an objective with respect to controller parameters is not (Amos et al., 2018). This is analogous to the dual-estimation problem formulation in Adaptive MPC, i.e., simultaneously estimating states and parameters. Even for a

linear system, the dual-estimation problem yields a non-convex problem (Fux et al., 2014).

4. APPROACH

The overall framework of our approach is summarized in **Figure 1**. To make our agent precocial, we took advantage of domain knowledge on HVAC control and expert demonstration from existing controllers. Specifically, a Gnu-RL agent utilizes a Differentiable MPC policy, which encodes domain knowledge on planning and system dynamics. The training includes two phases: offline pretraining and online learning. In the offline pretraining phase, the agent is initialized by imitating the historical state-action pairs from the existing controller. Using this approach, the Gnu-RL agent learns to behave similarly to the existing controller, without any interaction with the environment. Thus, the pretrained agent is precocial and may be deployed into a real-world environment directly with minimal disturbance to normal operation. In the online learning phase, the agent interacts with the environment and improves its policy using a policy gradient algorithm. While the agent already performs reasonably well at the onset of online learning phase, it continues to fine-tune its policy based on new observations.

We first formulate the HVAC control problem (section 4.1), and then elaborate on the procedures used to train the agent during the offline pretraining phase (section 4.2) and the online learning phase (section 4.3).

4.1. Problem Formulation

We adapt the problem formulation in the Differentiable MPC policy (i.e., Equation 7) to HVAC control. We use a linear model for the system dynamics, as shown in Equation (8a). Though building thermodynamics are non-linear in nature, we assume that it may be locally linearized for the state-action space and the temporal resolution that we are interested in Privara et al. (2013). We define the state x_t as the zone temperature and the action u_t depends on the specific problem. One should choose the action such that it is consistent with the linear assumption or linearize it to be so. Besides the state x_t and the control action u_t, we also consider uncontrollable disturbances d_t, such as weather and internal thermal gains. We define the number of states, actions, and disturbances as m, n, p, respectively. Thus, $x_t \in \mathbb{R}^m$, $u_t \in \mathbb{R}^n$, $d_t \in \mathbb{R}^p$, $A \in \mathbb{R}^{m\times m}$, $B_u \in \mathbb{R}^{m\times n}$, and $B_d \in \mathbb{R}^{m\times p}$. Equation (8a) can be written in the form of Equation (7b), as shown in Equation (8b). While the original formulation of Differentiable MPC policy learns f_t, we only learn B_d since the disturbances may be supplied by predictive models. Thus, the learnable parameters θ are $\{A, B_u, B_d\}$, which characterize the building thermodynamics. Compared to using a neural network policy, the number of free parameters is drastically reduced and the policy is interpretable to engineers. At each time step t, we provide the agent with predictive information on disturbance for the planning horizon, i.e., $d_{t:t+T-1}$.

$$x_{t+1} = Ax_t + B_u u_t + B_d d_t \tag{8a}$$

$$= \underbrace{\begin{bmatrix} A & B_u \end{bmatrix}}_{F} \underbrace{\begin{bmatrix} x_t \\ u_t \end{bmatrix}}_{\tau_t} + \underbrace{B_d d_t}_{f_t} \tag{8b}$$

Our objective (Equation 9) is to minimize energy consumption, while maintaining thermal comfort. We balance relative importance of thermal comfort and energy with hyperparameter η. One may choose different values of η for occupied and unoccupied periods. We use the quadratic different difference between actual zone temperatures and setpoints as a proxy for thermal comfort, and thus $O_t = \eta_t I_m$ and $p_t = -\eta_t x_{t,\text{setpoint}}$. The cost with respect to actions may be defined based on the specific problem; some options may be $R_t = I_n$ or $s_t = \vec{1}$. Similarly, Equation (9a) can be written in the form of Equation (7a), as shown in Equation (9b). The Differentiable MPC allows for a learnable cost function, but we assume the cost function to be specified by engineers.

$$C_t(x_t, u_t) = \frac{1}{2} x_t^T O_t x_t + p_t^T x_t + \frac{1}{2} u_t^T R_t u_t + s_t^T u_t \tag{9a}$$

$$= \frac{1}{2} \underbrace{\begin{bmatrix} x_t^T & u_t^T \end{bmatrix}}_{\tau_t^T} \underbrace{\begin{bmatrix} O_t & 0 \\ 0 & R_t \end{bmatrix}}_{C_t} \underbrace{\begin{bmatrix} x_t \\ u_t \end{bmatrix}}_{\tau_t} + \underbrace{\begin{bmatrix} p_t^T & s_t^T \end{bmatrix}}_{c_t^T} \underbrace{\begin{bmatrix} x_t \\ u_t \end{bmatrix}}_{\tau_t} \tag{9b}$$

There are some finer points that we need to highlight. The Differentiable MPC policy outputs the optimal trajectory over the planning horizon, i.e., $\tau^*_{t:t+T-1} = \{x^*_t, u^*_t\}_{t:t+T-1}$. However, we only take the first optimal action u^*_t and re-plan at the next time step based on new observations. This avoids compounding model error over time. We use the re-planning procedure for both offline pretraining and online learning. Moreover, since the HVAC control problems we are interested in have continuous action spaces, we use a Gaussian policy (Sutton and Barto, 2018) around the optimal action u^*_t (Equation 10). σ can be interpreted as the amount of exploration.

$$\hat{u}_t \sim \pi_\theta(u|x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(u-u^*_t)^2}{2\sigma^2}\right) \tag{10}$$

It should be noted that Amos et al. (2018) demonstrated the Differentiable MPC policy in the context of imitation learning, which is a supervised learning problem. Here we extended it to policy learning, which is generally considered as a harder problem.

4.2. Offline Pretraining

Imitation learning is a supervised approach for an agent to learn a policy. The premise is that it is easier for the expert to demonstrate the desired behavior, compared to asking the expert to encode or fine-tune a policy (Silver et al., 2010). In an HVAC control application, the existing controller can be considered as the expert and the historical state-action pairs logged in BAS as expert demonstrations. Specifically, the agent learns the mapping between states to actions, i.e., the policy $\pi_\theta(u|x)$, using expert demonstrations as ground truth.

In behavior cloning, one minimizes the mean squared error (MSE) loss between the expert actions and learner actions.

Since the Differentiable MPC policy also produces next-state predictions, we minimized the MSE loss between the states and actions from the expert and our agent simultaneously, as given by Equation (11). u_t and $\hat{u}_t$ are the actions from the expert and the learner, respectively. x_{t+1} and $\hat{x}_{t+1}$ are the actual next state vs. the next state predicted by the learner. The hyperparameter λ balances the relative importance of actions and next-state predictions. For instance, the engineer can choose a larger λ, if he has limited confidence on the actions of the existing controller. The procedures for offline pre-training are outlined in Algorithm 1. We can repeat the procedures in Algorithm 1 for a suitable number of epochs. For parameter selection, we make a train-test split over the expert demonstrations and select $\hat{\theta}$ with the smallest test loss.

$$\mathcal{L}_{\text{Imit}}(\theta) = \sum_t \lambda ||x_{t+1} - \hat{x}_{t+1}||_2^2 + ||u_t - \hat{u}_t||_2^2 \tag{11}$$

Algorithm 1: Offline Pre-training—Imitation Learning.

Input: A Differentiable MPC policy π_θ;
Expert demonstrations X, U;
Randomly initialize policy parameter $\theta = \{A, B_d, B_u\}$;
for $i = 0, \ldots,$ # *Episodes* **do**
  for $t = 0, \ldots,$ # *Steps* **do**
    $\hat{u}_t = \pi_\theta(x_t)$;
    $\hat{x}_{t+1} = f_\theta(x_t, u_t)$;
  end
  $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\text{Imit}}(\theta)$;
end
Output: A pre-trained policy, $\pi_{\tilde{\theta}}$

4.3. Online Learning

We adopt a policy gradient algorithm for this paper, because it integrates naturally with the Differentiable MPC policy. To elaborate, one can replace a neural network with a Differentiable MPC policy, and update model parameters, θ, using the same approach as laid out in Equation (3). The procedures for online learning with PPO are outlined in Algorithm 2.

As mentioned in section 3.2, there are a number of possible choices for advantage estimate $\hat{A}_t$. In our prior work (Chen et al., 2019), we used the total rewards (Equation 6a) for ease of implementation. But, this option also results in the largest variance. On the other hand, using advantage function results in the smallest variance, but the advantage function must be learned first (Schulman et al., 2015b), which is problematic given we want our agent to be precocial. A good compromise may be the baselined version of total rewards given in Equation (12), where π_0 refers to the policy from the existing controller and V_{π_0} is its value function. V_{π_0} can be learned offline based on historical data. Since V_{π_0} is not a function of θ, $\nabla_\theta V_{\pi_0} = 0$. Thus, offset a baseline from the total rewards reduces variance without introducing a bias. The intuition of this formulation is well-explained in Jia et al. (2019). Due to nature of HVAC control problem, the rewards fluctuate with weather and operation condition, e.g., occupied and unoccupied, regardless of the policy. To reduce variance from using raw rewards, one can use offset the rewards by those that would have been obtained by the existing controller. This reformulates the original objective of maximizing expected total reward to improving upon the policy of the existing controller.

$$\hat{A}_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} - V_{\pi_0}(x_t) \tag{12}$$

Algorithm 2: Online Learning—PPO (Modified from Schulman et al., 2017).

Input: A pretrained policy, $\pi_{\tilde{\theta}}$;
for $i = 0, \ldots,$ # *Episodes* **do**
  $\theta_{old} \leftarrow \theta$;
  for $t = 0, \ldots,$ # *Steps* **do**
    $\hat{u}_t = \pi_\theta(x_t)$;
    $x_{t+1}, r_{t+1} = \text{env.step}(\hat{u}_t)$;
  end
  Compute $\hat{A}_t$;
  With minibatch of size M;
    $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{PPO}(\theta)$;
end

5. EXPERIMENT 1: SIMULATION STUDY ON COMMERCIAL REFERENCE BUILDINGS

We validated that our proposed approach is indeed a practical and scalable solution for HVAC control. We also show our approach is generalizable across different buildings, by applying it to the `warehouse`, the `small office`, and the `medium office` from the DOE commercial reference buildings (Deru et al., 2011). We demonstrated that the Differentiable MPC policy is superior to a generic neural network policy, in that it is interpretable, more sample efficient, and has smaller performance variance. Specifically, we compared the Differentiable MPC policy to a long short-term memory (LSTM) network, with reference to Wang et al. (2017). Finally, we established two performance baselines for benchmarking the RL agents: an optimal LQR, i.e., the theoretical performance upper bound, and a PI controller, which is representative of controllers in existing buildings.

5.1. Simulation Environments

The simulation environments used in this experiment are based on the `warehouse`, the `small office`, and the `medium office` from DOE commercial reference buildings. We utilized OpenBuild (Gorecki et al., 2015), a toolbox for co-simulation and controller design, for fast prototyping of controllable environments based on the

TABLE 1 | Dimensions of the state space models of the `warehouse`, the `small office`, and the `medium office`; Only the zone temperatures, i.e., the states, are observable, while the temperature at the other thermal nodes are not.

	Warehouse	Small office	Medium office
# of thermal nodes (l)	99	144	358
# of states (m)	2	5	15
# of actions (n)	2	5	15
# of disturbances (p)	27	57	85

EnergyPlus models. OpenBuild abstracts away the complexity of HVAC system and allows control over the heat flux to zones of the building directly. Furthermore, OpenBuild creates a linear state space model (SSM) of the building thermodynamics, such that the optimal performance may be calculated analytically to benchmark the performance of our proposed approach.

Each linear SSM is created based on the RC modeling framework, where the building envelope is represented as a connected graph of thermal nodes. We denote the temperature at these thermal nodes as $z_t \in \mathbb{R}^l$, where l is the number of thermal nodes. The disturbance d_t to each thermal node includes weather and internal thermal gains from lighting, equipment, and occupancy. Both the RC model parameters and disturbances are calculated based on building and weather descriptions from the EnergyPlus files. We tabulated the dimensions of the SSM created by OpenBuild in **Table 1**.

We assume that only the zone temperatures x_t are observable. As one can see in **Table 1** the number of zones is much smaller than that of the thermal nodes, i.e., $m << l$. The actions u_t are the heat flux to each zone, and thus $m = n$ and B_u is a square matrix for each environment. For all three buildings, we conducted the experiment on Typical Meteorological Year 3 (TMY3) weather sequence (Wilcox and Marion, 2008) in Chicago. For this experiment, we defined the cost function as in Equation (13) and use a constant $\eta = 10$.

$$C_t(x_t, u_t) = \frac{\eta}{2}||x_t - x_{t,\text{setpoint}}||_2^2 + \frac{1}{2}||u_t||_2^2 \tag{13}$$

5.2. Implementation Details

All three environments used a 15-min simulation and control time step. Both the Gnu-RL agent and the LSTM agent plan ahead for 6 steps, i.e., a 1.5-h planning horizon. We considered each calendar day as an episode and each calendar year as an epoch. To making training easier, particularly for the LSTM policy, we incorporated an episodic reset mechanism, i.e., the temperature at all thermal nodes was reset to setpoint at the beginning of each day. This prevents the agent from being stuck at undesirable state-space for excess amount of time. We used min-max normalization to normalize all disturbance terms to 0–1. We also provided the agent with ground truth information on future disturbances.

RL for all our experiments[4] was implemented in PyTorch (Paszke et al., 2017). Following Amos et al. (2018), we used RMSprop (Tieleman and Hinton, 2012) as the optimizer for the Differentiable MPC policy and ADAM (Kingma and Ba, 2014) as the optimizer for the LSTM policy. The LSTM policy has 2 layers, each with 32 units and ReLU activation. To have a fair comparison, the LSTM policy has the same input and output as the Differentiable MPC policy, i.e., input $= \{x_t, d_{t:t+T-1}\}$ and output $= \{u_{t:t+T-1}\}$.

All experiments were repeated over five random seeds. For offline pretraining, we used a learning rate of 1×10^{-3}, except for the Gnu-RL agent in the `small office`, where we used a learning rate of 1×10^{-2} due to the particularly bad initialization. For the Differentiable MPC policies, we initialized A and B_u to be identity matrix and randomly initialized B_d with a uniform distribution over [0, 0.1]. The LSTM policies were initialized by PyTorch defaults. While we minimizes the imitation loss during offline pretraining, we evaluated the performance directly by letting the agents control the environment. Specifically, we froze the policy every 100 episodes and let the agent control the environment with a reduced amount of exploration, i.e., $\sigma = 0.1$. We report the mean and standard deviation of the episodic rewards over 30 randomly sampled episodes.

For online learning, we used a learning rate of 2.5×10^{-4} for the Differentiable MPC policy, and a learning rate of 5×10^{-4} for the LSTM policy. We evaluated the performance following the same procedure as in offline pretraining, i.e., we evaluate the policy every 100 episodes and report the mean and standard deviation of rewards over 30 episodes. We re-scaled the reward to be around 1, for better performance (Henderson et al., 2018). For hyperparameters, we used $\gamma = 0.8$, $\epsilon = 0.1$, and M = 48. For the Differentiable MPC policy, we used a σ that linearly decayed from 1 to 0.1. For the LSTM policy, we let the neural network learn σ simultaneously.

The results for both offline pretraining and online learning are summarized in **Figure 4**, where we compared the performance of the Differentiable MPC policy with that of the LSTM policy. For each experiment, we average the mean and the standard deviation of episodic rewards over the five runs and show the confidence interval of one standard deviation around the mean. At the same time, we also plotted the performance of individual runs with a thinner line weight. For ease of comparison, we normalize all rewards by that of the PI controller for the corresponding environment.

We also compared the performance of RL agents with two baselines: an optimal LQR and a PI controller.

5.2.1. Optimal LQR

Since OpenBuild linearized the system dynamics, one can derive the optimal performance analytically for each environment with LQR. We assume the LQR has ground truth parameters of the model, full observability over all the thermal nodes z_t, and perfect predictive information of disturbances. These assumptions are not realistic. But, this provides us with a theoretical upper bound for the control performance.

[4]The code is available at https://github.com/INFERLab/Gnu-RL

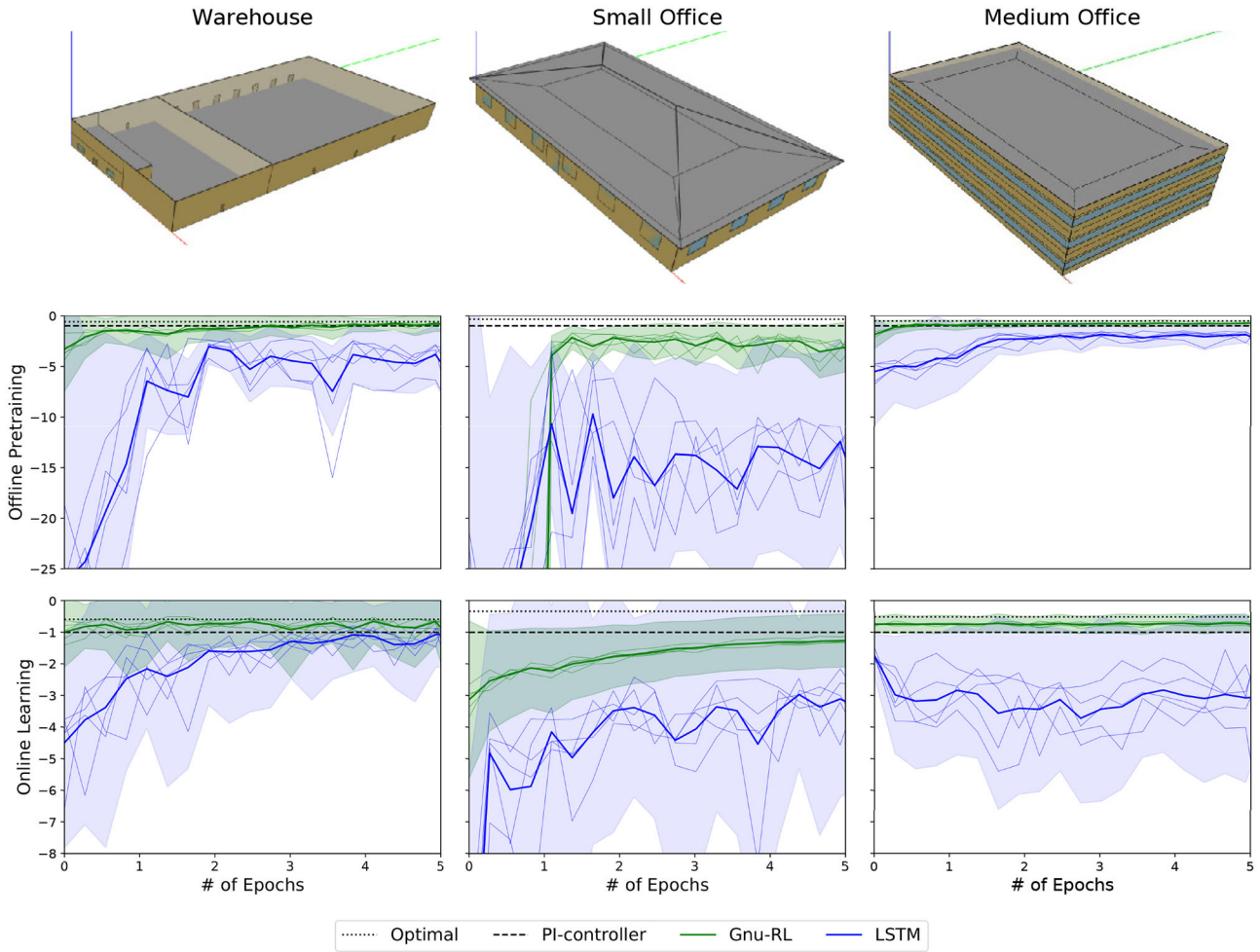

FIGURE 4 | Summary of the experiments on the `warehouse`, the `small office`, and the `medium office`; Each experiment is repeated over five random seeds. The performance of our proposed Gnu-RL agent is compared to a LSTM policy, during offline pretraining and online learning. We also include two baselines: an optimal LQR and a PI controller. All rewards are normalized by that of the PI controller for the corresponding environment.

5.2.2. PI Controller

For a more realistic performance baseline, we developed a PI controller for each zone with MATLAB PID tuner (Mathworks, 2020) based on ground truth model parameters. This is representative of controllers in existing buildings. Furthermore, we simulate state-action pairs with these PI controllers as expert demonstration.

5.3. Results: Offline Pretraining

In the offline pretraining phase, the agents were pretrained by imitating expert demonstration from the PI controllers. We trained all agents for 5 epochs, i.e., we go through 1-year worth of expert demonstration for five times. As shown in **Figure 4**, the performance had plateaued by then. By the end of offline pretraining, the Gnu-RL agents were performing similarly to the PI controllers. In fact, the Differentiable MPC policy outperformed the PI controller in the `warehouse` and the `medium office`. We hypothesize that the domain knowledge encoded in the Differentiable MPC policy enabled the agent to extrapolate beyond expert demonstration. While the Gnu-RL agent in the `small office` was not performing as well as the PI controller, it drastically improved upon its poor initial performance. Furthermore, given the same information, the Differentiable MPC policy achieved significantly better performance than its LSTM counterpart. This phenomenon was also observed in Amos et al. (2018). Due to its encoded knowledge, the Differentiable MPC policy was able to learn with lower sample complexity compared to a neural network. Finally, the Differentiable MPC policy has much smaller performance variance than the LSTM policy, which is desirable in practice. The same characteristics was also observed during online learning.

Note that the Differentiable MPC policies were initialized in the same way for the three environments, which worked well for the `warehouse` and the `medium office`, but not for the `small office`. This implies the initialization scheme should be based on the specific environment. Since the parameters could be trapped in local minimal, it is preferable to initialize as well as possible. Engineering estimates of the parameters, which

have well-defined physical meaning, may be a more appropriate initialization scheme.

5.4. Results: Online Training

In the online training phase, the agents continue to improve their policies through direct interaction with the environment. We trained all agents for 5 epochs, i.e., we go through the TMY3 weather sequence five times. As shown in **Figure 4**, the Gnu-RL agents in the `warehouse` and the `medium office` were already performing better than the PI controller at the onset of online training phase and thus they basically provided energy savings and/or comfort improvement for free. On the other hand, the LSTM policy consistently under-performed the Gnu-RL agents over the 5-years training period. In fact, the best-performing LSTM policy only approached the performance of the PI controller in the `warehouse` at the end of 5-years training period. Theoretically, a sufficiently expressive neural network policy would eventually outperform the model-based Differentiable MPC policy. But, that is not meaningful for practical applications.

In the `warehouse`, the Gnu-RL agent improved its performance approaching the optimal. But, we also see fluctuations in performance. This may be a result of learning rate being too large, when the performance was already close to optimal. The performance of the Gnu-RL agent in the `small office` suffered due to the poor initialization. Regardless, the agent improved its policy and approached the performance of the PI controller over time. This again highlights the importance of having a reasonably good initialization. Alternatively, other improvements may be available to expedite the learning. In the `medium office`, the performance curves for both the Gnu-RL agent and the LSTM policy were close to flat throughout the training period. We hypothesize the large state-action space of this environment makes convergence difficult (Liu and Henze, 2006).

6. EXPERIMENT 2: SIMULATION STUDY ON INTELLIGENT WORKSPACE

We also validated our approach in a simulation environment with detailed HVAC system. Specifically, we trained and evaluated our agent using the EnergyPlus model from Zhang and Lam (2018), which was modeled after the Intelligent Workspace (IW) on Carnegie Mellon University (CMU) campus. For offline pretraining, we used a baseline P-controller for expert demonstration and simulated the state-actions pairs under the TMY3 weather sequence, from Jan. 1st to Mar. 31st. We pretrained our agent on the simulated state-action pairs. For online learning, We deployed our agent in the simulation environment, using the weather sequence in 2017 from Jan. 1st to Mar. 31st. Since the simulation environment, the state-action space, and the weather sequence for training and testing are the same as those in Zhang and Lam (2018), our results are directly comparable. However, Zhang and Lam (2018) assumed the existence of a high-fidelity model for training, while we only assumed the existence of historical data from the existing controller.

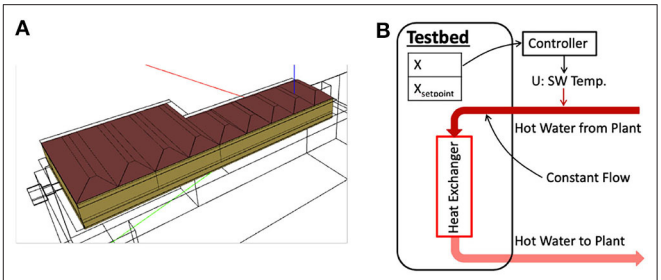

FIGURE 5 | Simulation testbed based on Intelligent Workspace. **(A)** Is a geometric view of the EnergyPlus model rendered by OpenStudio (Guglielmetti et al., 2011), and **(B)** Is a schematic of the water-based radiant heating system.

To understand how our approach compare to MPC, we compared imitation learning with system identification during the offline pretraining stage, and policy gradient methods with Adaptive MPC during the online learning stage. In the offline pretraining phase, we initialized our agent with imitation learning. In comparison, it is possible to initialize the agent with system identification using the same information. System identification is the class of methods that estimate model parameters of a dynamic system based on input and output signals (Ljung, 1999). Specifically, prediction error methods (PEM) look for parameters that minimize the difference between predicted states and observed states. For online learning, we compared our approach to Adaptive MPC. RL algorithms update model parameters end-to-end, with the objective of maximizing expected reward. On the other hand, it is also possible to update parameters online using Adaptive MPC, with the objective of minimizing prediction error (Fux et al., 2014; Maasoumy et al., 2014).

6.1. Simulation Testbed

The IW (**Figure 5**) is a 600 m^2 multi-functional space, including a classroom, a common area, and offices. We used the same EnergyPlus model used in Zhang and Lam (2018), which was calibrated against operational data. In this experiment, we controlled the water-based radiant heating system. **Figure 5B** shows a schematic of the system and the control logic. The hot water is supplied by a district heating plant. The supply water (SW) flow is kept constant and the supply water temperature is controlled by a P-controller to maintain zone temperature. We trained our agent to control the supply water temperature in place of the existing P-controller during the heating season. The allowable range of supply water temperature is 20–65°C. The variables considered for this problem are listed in **Table 2**. The cost function used for this experiment and the real-world experiment presented in section 7 in given in Equation (14).

$$C_t(x_t, u_t) = \frac{\eta_t}{2}||x_t - x_{t,\text{setpoint}}||_2^2 + \frac{1}{2}||u_t||_1 \qquad (14)$$

TABLE 2 | The state, action, and disturbance terms defined for the simulation study on Intelligent Workspace, a 600 m^2 multi-functional space.

X-state	Zone temperature (°C)
U-Control Action	SW temperature (°C)
D-Disturbance	Outdoor air temperature (°C)
	Outdoor air Relative Humidity (%)
	Diffuse solar radiation (W/m^2)
	Direct solar radiation (W/m^2)
	Occupancy flag
	Wind speed (m/s)

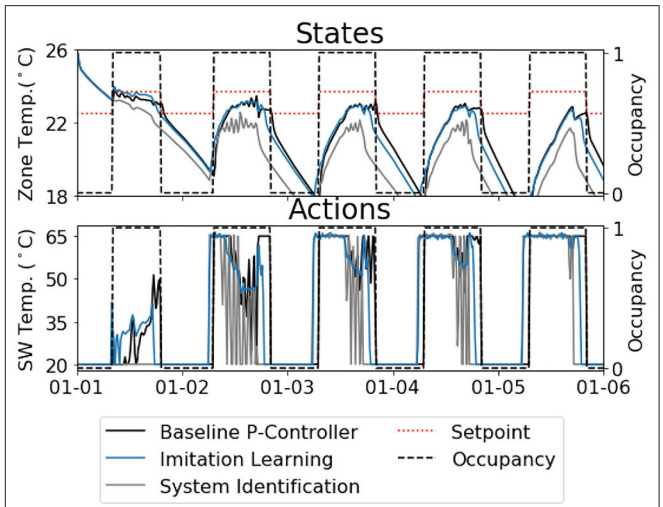

FIGURE 6 | Comparison of two initialization schemes: imitation learning vs. system identification (evaluated on TMY3 weather sequence); The agent initialized with imitation learning behaved similarly to the baseline P-controller, while the agent initialized with system identification consistently underestimated the heat requirement. Both agents were initialized with the same information, i.e., export demonstration from the baseline P-controller.

6.2. Implementation Details

The implementation details are the same as section 5.2, unless specified otherwise. We used the OpenAI Gym (Brockman et al., 2016) wrapper for EnergyPlus developed in Zhang and Lam (2018) to interface with the simulation environment. The EnergyPlus model has a 5-min simulation time step. Following Zhang and Lam (2018), each action was repeated for three times (a 15-min control time step). The agent plans ahead for 12 steps (a 3-h planning horizon). We shifted the 20–65°C range of supply water temperature setpoint to 0–45°C for the control action. We used $\eta = 3$ during occupied periods and 0.1 during unoccupied periods. For offline pre-training, we used a learning rate of 1×10^{-4} and a λ of 100. λ was adjusted so that loss from states and loss from actions were about the same magnitude. During online training, we used a learning rate of 5×10^{-4}.

6.3. Results: Offline Pretraining

6.3.1. Existing vs. Baseline P-Controller

The existing P-controller for supply water temperature operates 24/7, which is not the intended behavior for our agent and is not a fair comparison[5]. Instead, we modified the existing P-controller to be operational only during occupied periods, and call it the baseline P-controller. We simulated state-action pairs using the baseline P-controller under TMY3 weather sequence from Jan. 1st to Mar. 31st, as expert demonstrations.

We compared the performance of initializing model parameters, i.e., $\theta = \{A, B_u, B_d\}$, with imitation learning and system identification. We used PEM for system identification, as described in Privara et al. (2013). We assumed the same model (Equation 8a) and used the same time series for both initialization schemes. We evaluated the performance of the two initialization schemes by letting the pretrained agents control the simulation environment under the TMY3 weather sequence, with fixed parameters. **Figure 6** shows the behavior of the initialized agents over a 5-days period, neither of which had interacted with the environment before. The agent initialized with imitation learning behaved similarly to the baseline P-controller and tracked temperature setpoint well. The agent initialized with system identification, however, consistently underestimated the amount of heating required, despite its small prediction error (RMSE = 0.15°C).

The poor performance of the agent initialized with system identification is not surprising, as the experimental conditions required for accurate identification of building systems fall outside normal building operations (Agbi et al., 2012). In practice, excitation signals from actuators were often necessary to identify model parameters (Prívara et al., 2011; Agbi et al., 2012). However, such procedure requires careful design of experiments (Agbi et al., 2012) and may disturb normal operation. Instead, we successfully initialized the agent with imitation learning on observational data, which neither required experimentation nor disturbed occupants. The superior performance of imitation learning can be attributed to the fact that the agent imitated the expert on top of learning system dynamics. Learning how the expert would have acted under a given circumstance was directly relevant to the control task.

6.4. Results: Online Training

After pretraining, our agent controlled the environment using the actual weather sequence in 2017. The left hand side of **Figure 7** shows the behavior of Gnu-RL at the onset of training for a 4-days period. Gnu-RL already knew how to track temperature setpoint as well as the baseline P-controller, despite the fact that it had not interacted with the environment before. In comparison, a recent publication on the same environment (Zhang and Lam, 2018) took 47.5 years in simulation to achieve similar performance to the existing controller.

The results with comparison to Zhang and Lam (2018) are tabulated in **Table 3**. The heating demand and predicted percent dissatisfied (PPD) were calculated by EnergyPlus. Similar to Zhang and Lam (2018), we only considered PPD during occupied

[5]To illustrate the control strategy used by the existing P-controller, we included the data traces from the actual system from Jan. 1 to Jan. 4, 2017 in **Figure 7**.

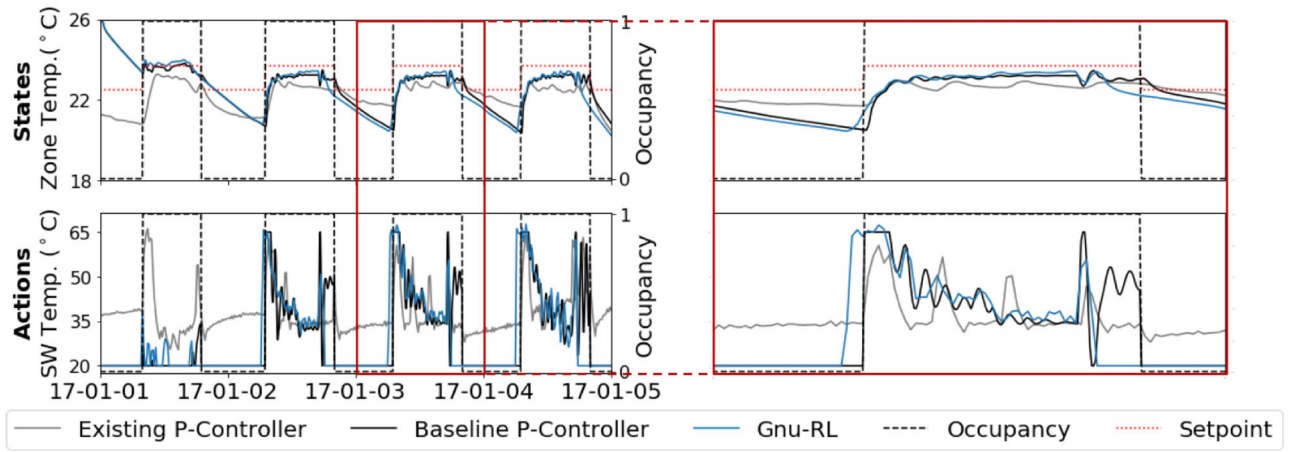

FIGURE 7 | Performance of the Gnu-RL agent at the onset of deployment; The Gnu-RL already tracked temperature setpoint as well as the baseline P-controller.

TABLE 3 | Comparison of performance during online learning phase.

	Heating demand (kW)	PPD mean (%)	PPD SD (%)
Existing P-Controller (Zhang and Lam, 2018)	43,709	9.45	5.59
Agent #6 (Zhang and Lam, 2018)	37,131	11.71	3.76
Baseline P-Controller	35,792	9.71	6.87
Gnu-RL	34,687	9.56	6.39
Gnu-RL + $\mathcal{L}_{PEM}$	24,901	18.77	12.48

periods. Our agent saved 6.6% energy and better maintained comfort compared to the best performing RL agent in Zhang and Lam (2018). To understand where the energy savings come from, we show a close-up view of the state-action pairs over a single day on the right hand side of **Figure 7**. While the baseline P-controller heats up the space following a fixed occupancy schedule, Gnu-RL preheats the space prior to occupancy and lets temperature float toward the end of occupancy. This explains the savings with respect to the baseline P-controller. It is worth noting that the preheating behavior was not present in the baseline P-controller. The knowledge embedded in the Differentiable MPC policy enabled our agent to extrapolate beyond expert demonstration.

Our approach finds model parameters that maximize expected reward using a policy gradient algorithm. Alternatively, Adaptive MPC updates model parameters online by minimizing prediction error. We compare the performance of two approaches with their respective objectives: minimizing prediction error (Equation 15) and maximizing expected reward (Equation 5). To minimize prediction error, we use the same procedures as in Algorithm 2, but use $\mathcal{L}_{PEM}$ in place of $\mathcal{L}_{PPO}$. Both agents are initialized with the same parameters from imitation learning.

$$\mathcal{L}_{\text{PEM}}(\theta) = \sum_{t} (x_{t+1} - \hat{x}_{t+1})^2 \tag{15}$$

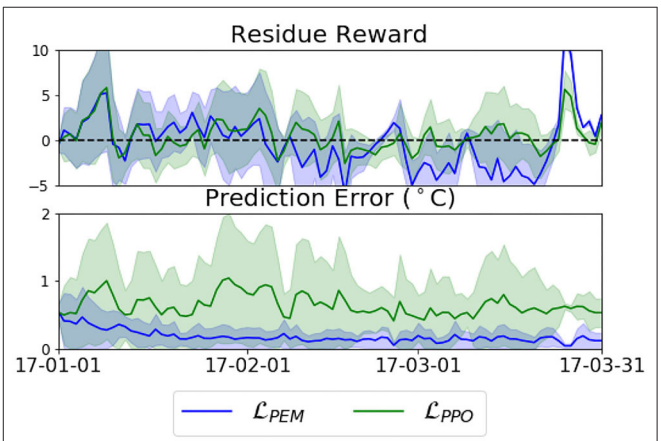

FIGURE 8 | Comparison of two approaches Policy Gradient ($\mathcal{L}_{PPO}$) vs. Adaptive MPC ($\mathcal{L}_{PEM}$); While optimizing $\mathcal{L}_{PEM}$ resulted in consistently smaller prediction error, optimizing $\mathcal{L}_{PPO}$ resulted in larger overall reward.

Figure 8 compares the performance of optimizing two different objectives over time. Because the rewards are also a function of the weather sequence, we show the difference between rewards from the agent and that from the baseline P-controller, which we call residual reward. While optimizing $\mathcal{L}_{PEM}$ resulted in consistently smaller prediction error, optimizing $\mathcal{L}_{PPO}$ resulted in larger overall reward. **Table 3** also shows that the PEM agent failed to maintain comfort, despite its small prediction error. One way to interpret this result is that minimizing prediction error is only a surrogate for learning a control policy (Amos et al., 2018). It is clear from the comparison that small prediction error does not necessarily translate to good control performance. We observed a similar result in section 6.3. Another common observation from both comparisons is that it is highly-effective to directly optimize the task objective, whether it is imitation or control.

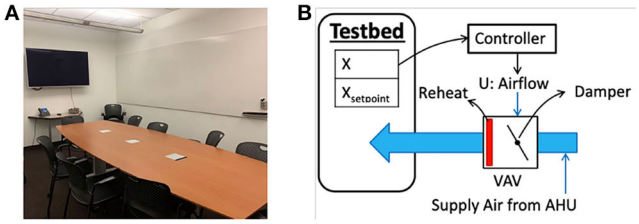

FIGURE 9 | Real-world testbed. **(A)** Is a photo of the conference room, and **(B)** Is a schematic of the VAV under control.

TABLE 4 | The state, action, and disturbance terms defined for controlling a real-world conference room.

X-state	Zone temperature (°F)
U-control action	Supply airflow (CFM)
D-disturbance	Outdoor air temperature (°F)
	Discharge air temperature (°F)
	Occupancy flag
	Occupancy count

7. EXPERIMENT 3: REAL-WORLD DEPLOYMENT

Given the promising results in our simulation studies, we repeated our experiment in a real-world conference room on campus, during Jun. 5th-Jun. 25st, 2019 to validate that our approach can make possible real-world deployment of RL for HVAC control with no prior information other than historical data. While the procedure here follows the same framework, there are additional challenges from a real-world deployment. Firstly, the existing controller in our testbed is not able to track temperature setpoint well. Thus, our agent needed to learn from sub-optimal expert moves. Secondly, real-world deployment demands a higher-level of robustness compared to simulation study. For instance, the agent's intended actions are not necessarily the same as the actions taken, e.g., there is a 1–2 min delay with the BAS interface. Finally, RL is sensitive to hyperparameters and other implementation details (Henderson et al., 2018). However, it is difficult to fine-tune these design choices in a real-world deployment. We resorted to using the implementation details that worked well for the simulation study (section 6.2) unless specified otherwise, although the implementation details may not be optimal for this specific problem.

7.1. Testbed

The conference room is a 20 m^2 single-zone space (**Figure 9**) controlled by a variable air volume (VAV) box. **Figure 9B** shows a schematic of the system and the control logic. In the cooling season, the VAV box discharges a variable volume of cool air into the room. The cool air is supplied by an air handling unit (AHU) at 55°F. In this experiment, we controlled the amount of airflow that was supplied to the room. We let the existing PID controller determine the damper position to meet our proposed airflow setpoint. The VAV box is also equipped with a hot-water-based reheat coil, which was kept closed throughout the experiment for energy efficiency. The variables used in the problem is listed in **Table 4**. In the existing control logic, the maximum allowable airflow is 200 CFM, and the minimum allowable values are 10 CFM for unoccupied periods and 35 CFM for occupied periods. We followed the same upper and lower bounds for our control action.

7.2. Implementation Details

The implementation details are the same as in section 6.2, unless specified otherwise. We used the same 15-min control time step. But, the thermal response of the real-world testbed is faster than the simulation one, due to the difference in system configuration. On hindsight, a smaller control time step may be more appropriate. We normalized both our disturbance terms and control actions to the range of 0–1. Due to the different scales of state-action space from the simulation experiment, we used $\eta = 2$ during occupied periods and 0.01 during unoccupied ones.

For offline pre-training, we used a learning rate of 1×10^{-3} and a $\lambda = 0.1$. For online learning, we used PI DataLink (OSIsoft, 2019) to access real time observations from BAS and we modified the existing control program to use the command we published. The parameters were updated every day at midnight. The conference room needs to be booked in advance, which gave us predictive information on occupancy. At the same time, the conference room is equipped with a depth-image-based occupancy sensor (Munir et al., 2017), which enabled our agent to adjust to unscheduled occupancy. To be conservative, the agent cools the space if there is either a scheduled meeting or occupancy based on the sensor. For predictive information on weather, we pulled weather forecast at the start of each day using Dark Sky API (Dark Sky, 2020). After we decided on the experimental testbed, we discovered that there was likely a leakage in the hot water supply for the reheat coil. As a result, the discharge air temperature is inversely proportional to the amount of airflow in the range of 56 to 65°F. Having a disturbance term that is a function of control action adds complexity to the environment. In practice, we assumed the discharge air temperature to be the current observed value for the entire planning horizon.

7.3. Results: Offline Pre-training

We used the time sequences from May 1 to August 31 in 2017 and 2018 for training and testing, respectively. We manually selected days where the controller tracks the temperature reasonably well: 20 days from 2017 for training and 13 days from 2018 for testing. Again, We pre-trained our agent using imitation learning. We repeated the procedures in Algorithm 1 over 20 epochs and picked the set of parameters with smallest test loss. The MSE were 0.1 and 0.028 for the state and normalized action, respectively.

7.4. Results: Online Training

Figure 10 shows how our agent's behavior evolved over the 3-weeks experiment period. Each snapshot shows the state-action pairs over a 1-day period. Initially (6/10), our agent

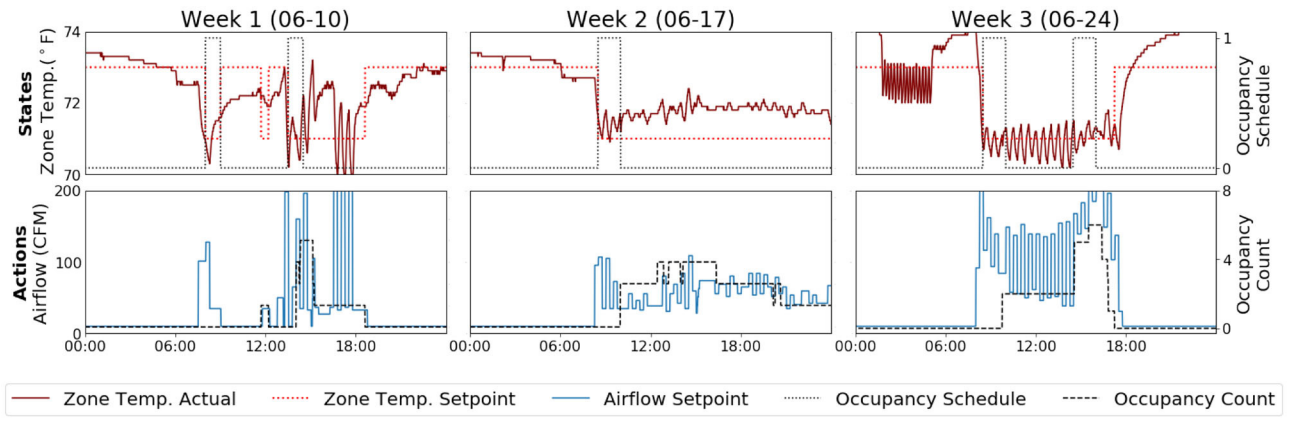

FIGURE 10 | Performance of the Gnu-RL agent during a 3-weeks real-world deployment; The Gnu-RL agent continuously improved its policy over time.

TABLE 5 | Summary of results for real-world deployment.

		Cooling demand	OAT		IAT
			Mean	SD	RMSE
		(kWh)	(°F)	(°F)	(°F)
Existing controller	Jun. 2017	169.4	69.6	6.9	2.4
	Jun. 2018	130.7	71.9	7.1	2.7
	Normalized	99.4	–	–	–
Gnu-RL		82.8	69.9	6.2	1.02

knew to pre-cool the space before scheduled occupancy, but it tended to overshoot. At Week 2 (6/17), the agent was no longer overshooting. But, it consistently underestimated the amount of cooling required to maintain temperature. By the end of the experiment (6/24), the agent was tracking setpoint reasonably well despite the varying number of occupants. Also shown in **Figure 10**, there were quite a few discrepancies between the meeting schedule and the real-time occupancy counts. Other than that, there were also counting errors from the occupancy sensor. For example, there is a positive cumulative counting error toward the end of 6/17.

The performance of our agent with comparison to the existing controller, which follows a fixed occupancy schedule from 6 a.m. to 10 p.m., is summarized in **Table 5**. The Gnu-RL agent saved 16.7% of total cooling demand compared to the existing control strategy, while tracking temperature setpoint significantly better. It should be noted that the cooling demand is not proportional to the energy consumption. The total cooling demand[6] of the existing controller is calculated using the historical data from Jun. 2017 and 2018 and is normalized for the duration of the experiment and outdoor air temperature (OAT), following the Weather Normalized Method suggested by Energy Star (Energy Star, 2018). Since it is difficult to calculate PPD for the real-world deployment, we used the RMSE between indoor air temperature (IAT) and setpoint as a proxy for comfort and evaluate it only during occupied periods.

[6]The cooling demand is calculated as $\dot{Q} = c\dot{m}\Delta T$, where Q is the cooling demand, c is the specific heat of air, m is the amount of airflow, and ΔT is the difference between mixed air temperature and supply air temperature from the AHU. Mixed air is the mixture of recirculation air and outdoor air.

8. DISCUSSION AND CONCLUSIONS

We proposed Gnu-RL, a precocial RL agent that is capable of controlling HVAC at "birth." To achieve this, we bootstrapped our agent with domain knowledge and expert demonstration. We demonstrated in both simulation studies and a real-world deployment that Gnu-RL had reasonably good initial performance and continued to improve over time. Firstly, we demonstrated that the Gnu-RL agent is scalable, by applying it to three different buildings. Furthermore, we showed that the Differentiable MPC policy is superior to a LSTM policy in that it is interpretable, more sample-efficient, and has smaller performance variance. In another simulation study, we benchmarked our approach to a recent publication, and our agent saved 6.6% energy compared to the best performing RL agent in Zhang and Lam (2018), while maintaining occupants' comfort better. We also compared our approach to alternatives, i.e., system identification and Adaptive MPC, and demonstrated that it is more effective to optimize task objectives end-to-end. In the real-world conference room, where Gnu-RL was deployed for a 3-weeks period, it saved 16.7% of cooling demand compared to the existing controller, while tracking the temperature setpoint better.

All the energy savings were achieved without the need for a high-fidelity model. Thus, to use our approach in practice, an engineer only needs to identify the state, action, and disturbance terms of interest and define the cost function. The only prior information we used was historical data from the existing controllers. While we discussed our approach in the context of HVAC, it is readily transferable to the control of other building systems. Furthermore, the requirement of historical data does not preclude the usage of this method on new buildings. Since, there are only a small number of free parameters and these parameters

have well-defined physical meaning, it is straightforward to initialize these parameters with engineering calculations.

In summary, our proposed approach, Gnu-RL, was shown to be a promising practical and scalable RL solution for HVAC control. However, there are many potential improvements to explore in future work, starting by relaxing some of the assumptions made here. For example, we assumed that future occupancy information was available, which is seldom the case. As future work, we will incorporate a probabilistic occupancy model into the RL framework. For reference, Levine (2018) presented the close connection between RL and probabilistic graphical models. Similarly, we assumed that building systems can be locally linearized. The assumption worked for the problems we considered, but it may or may not extrapolate to more complex problems.

Additionally, we identified a few directions for further research. Firstly, there is a need for standardized evaluation, including common simulation testbeds, baseline controllers, and evaluation procedures, such that researchers can compare their results on equal footings. As a step toward this direction, we conducted a simulation study in the same environment as in Zhang and Lam (2018), along with the same state-action space and weather sequence, making our results comparable. We also make our code publicly available at https://github.com/INFERLab/Gnu-RL. Regarding evaluation procedures, we refer readers to Henderson et al. (2018), which provided a thorough discussion on the challenges in reproducing RL results, along with recommendations.

Secondly, there is a need to develop offline evaluation procedures for pretrained agents (Dulac-Arnold et al., 2019). To elaborate, in our real-world deployment, we could only observe the imitation loss from our agent after offline pretraining. Yet, it was an indirect proxy for control performance. In fact, our agent tended to overshoot at initialization, contrary to our expectation based on the imitation loss. Thus, there is a need to evaluate the performance of pretrained agents without access to the environment. This is important in practice to reassure building owners/operators the expected performance of a novel controller before deploying it in a real building.

Thirdly, there are a number of engineering decisions one needs to make when applying our approaches, ranging from initialization of model parameters, the hyperparameter η that balances energy and comfort, and other hyperparameters used during online learning. Furthermore, one cannot expect to do hyperparameter selection for real buildings in the same way as in simulation. We made those decisions based on implementation details used in the literature and engineering judgments. While these decisions generally worked well in our experiments, they do not guarantee good results (recall the experiment on the `small office` in section 5). More experiments on different environments may lead to additional insights on how to make these decisions intelligently.

Finally, control problems with larger state-action spaces generally require longer learning/training time (Liu and Henze, 2006). This is observed in our experiment on the `medium office`. In existing buildings, HVAC controllers operates independently based on their local information. But, for scenarios where whole-building control is necessary, there may be a need for multi-agent RL, where the large original problem is subdivided into more tractable sub-problems.

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

BC conceived of the study, designed and implemented the experiments, and wrote the paper. ZC supplied other authors with background knowledge on model-based control and consulted on the experimental design. MB provided the guidance and oversaw the study. All authors read, edited, and contributed to the writing of final manuscript.

FUNDING

This material was based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Building Technologies Office Award Number DE-EE0007682.

ACKNOWLEDGMENTS

First and foremost, we would like to thank Brandon Amos, whose prior work was the primary inspiration for us. Secondly, we were indebted to Zhiang Zhang for sharing his model with us. Finally, we appreciate Michael Frenak's help in facilitating the real-world deployment. The content of this manuscript has been presented in part at the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (Chen et al., 2019).

REFERENCES

Agbi, C., Song, Z., and Krogh, B. (2012). "Parameter identifiability for multi-zone building models," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)* (Maui, HI: IEEE), 6951–6956. doi: 10.1109/CDC.2012.6425995

Amos, B., Jimenez, I., Sacks, J., Boots, B., and Kolter, J. Z. (2018). *Differentiable MPC for End-to-End Planning and Control* (Montreal, QC: Neural Information Processing Systems Foundation, Inc). pp. 8289–8300.

Amos, B., and Kolter, J. Z. (2017). "Optnet: differentiable optimization as a layer in neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70 (JMLR.org) (Sydney, NSW), 136–145.

Amos, B., Xu, L., and Kolter, J. Z. (2017). "Input convex neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70 (JMLR.org) (Sydney, NSW), 146–155.

Aswani, A., Master, N., Taneja, J., Culler, D., and Tomlin, C. (2012). Reducing transient and steady state electricity consumption in HVAC using learning-based model-predictive control. *Proc. IEEE* 100, 240–253. doi: 10.1109/JPROC.2011.2161242

Atam, E., and Helsen, L. (2016). Control-oriented thermal modeling of multizone buildings: methods and issues: intelligent control of a building system. *IEEE Control Syst. Mag.* 36, 86–111. doi: 10.1109/MCS.2016.2535913

Bengea, S., Kelman, A., Borrelli, F., Taylor, R., and Narayanan, S. (2012). "Model predictive control for mid-size commercial building HVAC: implementation, results and energy savings," in *Second International Conference on Building Energy and Environment* (Boulder, CO), 979–986.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., et al. (2016). OpenAI gym. *arXiv[Preprint].arXiv:1606.01540*.

Cai, J., Kim, D., Braun, J. E., and Hu, J. (2016). "Optimizing zone temperature setpoint excitation to minimize training data for data-driven dynamic building models," in *2016 American Control Conference (ACC)* (Boston, MA: IEEE), 1478–1484. doi: 10.1109/ACC.2016.7525125

Chen, B., Cai, Z., and Bergés, M. (2019). "Gnu-RL: a precocial reinforcement learning solution for building HVAC control using a differentiable mpc policy," in *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation* (New York, NY), 316–325. doi: 10.1145/3360322.3360849

Chen, Y., Shi, Y., and Zhang, B. (2018). Optimal control via neural networks: a convex approach. *arXiv[Preprint].arXiv:1805.11835*.

Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Advances in Neural Information Processing Systems* (Montreal, QC: Neural Information Processing Systems Foundation), 4754–4765.

Costanzo, G. T., Iacovella, S., Ruelens, F., Leurs, T., and Claessens, B. J. (2016). Experimental analysis of data-driven control for a building heating system. *Sustain. Energy Grids Netw.* 6, 81–90. doi: 10.1016/j.segan.2016.02.002

Dalamagkidis, K., Kolokotsa, D., Kalaitzakis, K., and Stavrakakis, G. S. (2007). Reinforcement learning for energy conservation and comfort in buildings. *Build. Environ.* 42, 2686–2698. doi: 10.1016/j.buildenv.2006.07.010

Dark Sky (2020). *Dark Sky API*. Available online at: https://darksky.net/dev (accessed June 19, 2019).

Deru, M., Field, K., Studer, D., Benne, K., Griffith, B., Torcellini, P., et al. (2011). *US Department of Energy Commercial Reference Building Models of the National Building Stock* (Golden, CO: National Renewable Energy Laboratory).

Dulac-Arnold, G., Mankowitz, D., and Hester, T. (2019). Challenges of real-world reinforcement learning. *arXiv[Prerpin].arXiv:1904.12901*.

Energy Star (2018). *Portfolio Manager Technical Reference: Climate and Weather*. Available online at: https://portfoliomanager.energystar.gov/pdf/reference/Climate%20and%20Weather.pdf (accessed June 19, 2019).

Evans, R., and Gao, J. (2016). *DeepMind AI Reduces Google Data Centre Cooling Bill by 40%*. Available online at: https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/ (accessed June 19, 2019).

Fux, S. F., Ashouri, A., Benz, M. J., and Guzzella, L. (2014). EKF based self-adaptive thermal model for a passive house. *Energy Build.* 68, 811–817. doi: 10.1016/j.enbuild.2012.06.016

Gao, G., Li, J., and Wen, Y. (2019). Energy-efficient thermal comfort control in smart buildings via deep reinforcement learning. *arXiv[Preprint].arXiv:1901.04693*.

Goetzler, W., Shandross, R., Young, J., Petritchenko, O., Ringo, D., and McClive, S. (2017). *Energy Savings Potential and RD&D Opportunities for Commercial Building HVAC Systems*. Technical report, Navigant Consulting, Burlington, MA, United States.

Gorecki, T. T., Qureshi, F. A., and Jones, C. N. (2015). "Openbuild: an integrated simulation environment for building control," in *2015 IEEE Conference on Control Applications (CCA)* (Sydney, NSW: IEEE), 1522–1527. doi: 10.1109/CCA.2015.7320826

Gu, B., Ergan, S., and Akinci, B. (2014). "Generating AS-IS building information models for facility management by leveraging heterogeneous existing information sources: a case study," in *Construction Research Congress 2014: Construction in a Global Network* (Altanta, GA), 1911–1920. doi: 10.1061/9780784413517.195

Guglielmetti, R., Macumber, D., and Long, N. (2011). *Openstudio: An Open Source Integrated Analysis Platform*. Technical report, National Renewable Energy Lab. (NREL), Golden, CO, United States.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). "Deep reinforcement learning that matters," in *Thirty-Second AAAI Conference on Artificial Intelligence* (New Orleans, LA).

Jia, R., Jin, M., Sun, K., Hong, T., and Spanos, C. (2019). Advanced building control via deep reinforcement learning. *Energy Proc.* 158, 6158–6163. doi: 10.1016/j.egypro.2019.01.494

Kakade, S. M. (2002). "A natural policy gradient," in *Advances in Neural Information Processing Systems* (Montreal, QC: Neural Information Processing Systems Foundation, Inc), 1531–1538.

Kakade, S. M. (2003). *On the sample complexity of reinforcement learning* (Ph.D. thesis), University of London, London.

Kamthe, S., and Deisenroth, M. P. (2017). Data-efficient reinforcement learning with probabilistic model predictive control. *arXiv[Preprint].arXiv:1706.06491*.

Karaguzel, O. T., and Lam, K. P. (2011). "Development of whole-building energy performance models as benchmarks for retrofit projects," in *Proceedings of the 2011 Winter Simulation Conference (WSC)* (Phoenix, AZ: IEEE), 838–849. doi: 10.1109/WSC.2011.6147810

Killian, M., and Kozek, M. (2016). Ten questions concerning model predictive control for energy efficient buildings. *Build. Environ.* 105, 403–412. doi: 10.1016/j.buildenv.2016.05.034

Kingma, D. P., and Ba, J. (2014). ADAM: a method for stochastic optimization. *arXiv[Preprint].arXiv:1412.6980*.

Levine, S. (2018). Reinforcement learning and control as probabilistic inference: tutorial and review. *arXiv[Preprint].arXiv:1805.00909*.

Li, X., and Wen, J. (2014). Review of building energy modeling for control and operation. *Renew. Sustain. Energy Rev.* 37, 517–537. doi: 10.1016/j.rser.2014.05.056

Li, Y., Wen, Y., Guan, K., and Tao, D. (2017). Transforming cooling optimization for green data center via deep reinforcement learning. *arXiv[Preprint].arXiv:1709.05077*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. *arXiv[Preprint].arXiv:1509.02971*.

Liu, S., and Henze, G. P. (2006). Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory: part 2, results and analysis. *Energy Build.* 38, 148–161. doi: 10.1016/j.enbuild.2005.06.002

Liu, S., and Henze, G. P. (2007). Evaluation of reinforcement learning for optimal control of building active and passive thermal storage inventory. *J. Sol. Energy Eng.* 129, 215–225. doi: 10.1115/1.2710491

Ljung, L. (1999). System identification. *Wiley Encycl. Electric. Electron. Eng.* 1–19. doi: 10.1002/047134608X.W1046

Lü, X., Lu, T., Kibert, C. J., and Viljanen, M. (2015). Modeling and forecasting energy consumption for heterogeneous buildings using a physical-statistical approach. *Appl. Energy* 144, 261–275. doi: 10.1016/j.apenergy.2014.12.019

Maasoumy, M., Razmara, M., Shahbakhti, M., and Vincentelli, A. S. (2014). Handling model uncertainty in model predictive control for energy efficient buildings. *Energy Build.* 77, 377–392. doi: 10.1016/j.enbuild.2014.03.057

Mathworks (2020). *Tune PID Controllers–MATLAB*. Available online at: https://www.mathworks.com/help/control/ref/pidtuner-app.html (accessed April 19, 2020).

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., et al. (2016). "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning* (New York, NY), 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing Atari with deep reinforcement learning. *arXiv[Preprint].arXiv:1312.5602*.

Moyer, C. (2016). How google's alphago beat a go world champion. *Atlantic* 28. Available online at: https://www.theatlantic.com/technology/archive/2016/03/the-invisible-opponent/475611/

Munir, S., Arora, R. S., Hesling, C., Li, J., Francis, J., Shelton, C., et al. (2017). "Real-time fine grained occupancy estimation using depth sensors on arm embedded platforms," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)* (Pittsburgh, PA: IEEE), 295–306. doi: 10.1109/RTAS.2017.8

Nagy, A., Kazmi, H., Cheaib, F., and Driesen, J. (2018). Deep reinforcement learning for optimal control of space heating. *arXiv[Preprint].arXiv:1805.03777*.

OSIsoft (2019). *Tools: PI DataLink*. Available online at: https://www.osisoft.com/pi-system/pi-capabilities/pi-system-tools/pi-datalink/ (accessed June 19, 2019).

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., et al. (2017). *Automatic Differentiation in Pytorch*.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018). "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 3803–3810. doi: 10.1109/ICRA.2018.8460528

Privara, S., Cigler, J., Váňa, Z., Oldewurtel, F., Sagerschnig, C., and Žáčeková, E. (2013). Building modeling as a crucial part for building predictive control. *Energy Build.* 56, 8–22. doi: 10.1016/j.enbuild.2012.10.024

Prívara, S., Váňa, Z., Gyalistras, D., Cigler, J., Sagerschnig, C., Morari, M., et al. (2011). "Modeling and identification of a large multi-zone office building," in *2011 IEEE International Conference on Control Applications (CCA)* (Denver, CO: IEEE), 55–60. doi: 10.1109/CCA.2011.6044402

Rockett, P., and Hathway, E. A. (2017). Model-predictive control for non-domestic buildings: a critical review and prospects. *Build. Res. Inform.* 45, 556–571. doi: 10.1080/09613218.2016.1139885

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015a). "Trust region policy optimization," in *International Conference on Machine Learning* (Lille), 1889–1897.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2015b). High-dimensional continuous control using generalized advantage estimation. *arXiv[Preprint].arXiv:1506.02438*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv[Preprint].arXiv:1707.06347*.

Silver, D., Bagnell, J. A., and Stentz, A. (2010). Learning from demonstration for autonomous navigation in complex unstructured terrain. *Int. J. Robot. Res.* 29, 1565–1592. doi: 10.1177/0278364910369715

Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Tieleman, T., and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA Neural Netw. Mach. Learn.* 4, 26–31. Available online at: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., et al. (2017). Starcraft II: a new challenge for reinforcement learning. *arXiv[Preprint].arXiv:1708.04782*.

Wang, Y., Velswamy, K., and Huang, B. (2017). A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems. *Processes* 5:46. doi: 10.3390/pr5030046

Watter, M., Springenberg, J., Boedecker, J., and Riedmiller, M. (2015). "Embed to control: a locally linear latent dynamics model for control from raw images," in *Advances in Neural Information Processing Systems* (Long Beach, CA: The Neural Information Processing Systems Foundation), 2746–2754.

Wei, T., Wang, Y., and Zhu, Q. (2017). "Deep reinforcement learning for building HVAC control," in *Proceedings of the 54th Annual Design Automation Conference 2017* (New York, NY: ACM), 22. doi: 10.1145/3061639.3062224

Wilcox, S., and Marion, W. (2008). *Users manual for TMY3 data sets*. Golden, CO: National Renewable Energy Laboratory.

Yang, L., Nagy, Z., Goffin, P., and Schlueter, A. (2015). Reinforcement learning for optimal control of low exergy buildings. *Appl. Energy* 156, 577–586. doi: 10.1016/j.apenergy.2015.07.050

Yuan, Y., Liu, K. S., Munir, S., Francis, J., Shelton, C., and Lin, S. (2020). "Leveraging fine-grained occupancy estimation patterns for effective HVAC control," in *Proceedings of the ACM/IEEE Conference on Internet of Things Design and Implementation, IoTDI '20* (Sydney, NSW). doi: 10.1109/IoTDI49375.2020.00016

Zhang, Z., and Lam, K. P. (2018). "Practical implementation and evaluation of deep reinforcement learning control for a radiant heating system," in *Proceedings of the 5th Conference on Systems for Built Environments, BuildSys '18* (New York, NY: ACM), 148–157. doi: 10.1145/3276774.3276775

Zhao, J., Lam, K. P., and Ydstie, B. E. (2013). "Energyplus Model-Based Predictive Control (EPMPC) by Using Matlab/Simulink and mle+," in *Proceedings of 13th Conference of International Building Performance Simulation Association* (Chambery).

Disclaimer: The views expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.