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

Bingqing Chen
Carnegie Mellon University
Pittsburgh, PA, USA
bingqinc@andrew.cmu.edu

Zicheng Cai
Dell Technologies
Austin, TX, USA
zicheng.cai@dell.com

Mario Bergés
Carnegie Mellon University
Pittsburgh, PA, USA
mberges@andrew.cmu.edu

ABSTRACT

Reinforcement learning (RL) was first demonstrated to be a feasible approach to controlling heating, ventilation, and air conditioning (HVAC) systems more than a decade ago. However, there has been limited progress towards a practical and scalable RL solution for HVAC control. While one can train an RL agent in simulation, it is not cost-effective to create a model for each thermal zone or building. Likewise, existing RL agents generally take a long time to learn and are opaque to expert interrogation, making them unattractive for real-world deployment.

To tackle these challenges, we propose Gnu-RL: a novel approach that enables practical deployment of RL for HVAC control and requires no prior information other than historical data from existing HVAC controllers. To achieve this, 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, which enables 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 on both an EnergyPlus model and a real-world testbed. In both experiments, our agents were directly deployed in the environment after offline pre-training on expert demonstration. In the simulation experiment, our approach saved 6.6% energy compared to the best published RL result for the same environment, while maintaining a higher level of occupant comfort. Next, Gnu-RL was deployed to control the HVAC of a real-world conference room for a three-week period. Our results show that Gnu-RL saved 16.7% of cooling demand compared to the existing controller and tracked temperature set-point better.

CCS CONCEPTS

• **Computing methodologies** → **Reinforcement learning**; *Control methods; Learning from demonstrations*; • **Computer systems organization** → *Sensors and actuators*.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

BuildSys '19, November 13–14, 2019, New York, NY, USA

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-7005-9/19/11...\$15.00

<https://doi.org/10.1145/3360322.3360849>

KEYWORDS

Deep Reinforcement Learning, HVAC Control

ACM Reference Format:

Bingqing Chen, Zicheng Cai, and Mario Bergés. 2019. Gnu-RL: A Precocial Reinforcement Learning Solution for Building HVAC Control Using a Differentiable MPC Policy. In *The 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '19)*, November 13–14, 2019, New York, NY, USA. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3360322.3360849>

1 INTRODUCTION

Advanced control strategies for operating HVAC systems have the potential to significantly improve the energy efficiency of our building stock and enhance comfort [8, 14, 23]. 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 [22], making their energy performance sub-optimal.

Optimal control strategies such as MPC address these drawbacks by iteratively optimizing an objective function over a receding time horizon. However, despite many successful applications of MPC (e.g., [4, 29, 40]), its wide-spread adoption has been limited by the need of accurate models [22, 39]. This is especially challenging because buildings are heterogeneous [28], e.g. they have different layouts, HVAC configurations, and occupancy patterns. Thus, custom models are required for each thermal zone or building, limiting the scalability of MPC. Another drawback is that MPC treats modeling and planning as two separate tasks. The quality of the model is evaluated by criteria such as prediction error, which may not lead to good control performance¹.

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 [47]. RL has achieved remarkable success in mastering many highly complex tasks such as playing Go [32] and StarCraft [49]. It has also been shown to be a feasible approach for optimal control of HVAC systems [11, 25]. However, the flexibility of the RL framework comes at the cost of increased sample complexity [20]. Taking recent published results as examples, an RL agent may need 5 million interaction steps (47.5 years in simulation) to achieve the same performance as a feedback controller on an HVAC system [53]. Even after pre-training an RL agent on expert demonstration, it may still require an additional year of training in simulation to achieve similar performance to a

¹We elaborate on this issue in Sections 5.3 and 5.4

baseline controller [18]. Furthermore, while these results show that RL agents can be trained successfully in simulation, high-fidelity models are generally not available for thermal zones or buildings. Thus, training RL agents in simulation shifts the focus back to modeling (or calibration) [52].

To overcome these limitations, we propose Gnu-RL²: 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 HVAC controllers. It leverages a recently-developed Differentiable MPC policy [2] that encodes domain knowledge on system dynamics and control. Prior to any interactions with the environment, a Gnu-RL agent can be pre-trained on historical data through imitation learning, thus enabling it to behave similarly to the existing controller once deployed. Following this it continues to improve its policy end-to-end using a policy gradient algorithm.

We evaluated the performance of Gnu-RL in the context of HVAC control in both a simulation scenario (an EnergyPlus model of a 600 m² office building from [53]) and a real-world deployment (a 20 m² conference room). First, Gnu-RL was directly deployed to control the simulation environment after offline pre-training on state-action pairs generated by a P-controller. In this setting, our algorithm saved 6.6% energy compared to the best-performing RL agent in [53], while maintaining a higher level of occupant comfort. Next, we deployed Gnu-RL in a real conference room for a three-week 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.

The rest of the paper is organized as follows. Section 2 reviews prior 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 (Section 5) and in a real-world testbed (Section 6), and present conclusions and possible future work in Section 7.

2 RELATED WORK

Because Gnu-RL is an RL algorithm with a Differentiable MPC policy, we review existing 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 [22].

However, it can be argued that the main roadblock preventing widespread adoption of MPC is its reliance on a model [22, 39]. By some estimates, modeling can account for up to 75% of the time and resources required for implementing MPC in practice [41]. Because buildings are highly heterogeneous, a custom model is required for each thermal zone or building [28].

²The 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.

Privara et al. [39] identified two paradigms for modeling building dynamics: physical-based and statistical-based. Physical-based models, e.g. EnergyPlus, utilize physical knowledge and material properties of a building to build detailed representations of the building dynamics, but they are generally not control-oriented [5]. Statistical-based 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 needed for accurate identification of building systems fall outside of normal building operations [1]. Furthermore, there are many sources of uncertainty in building dynamics [29], which makes accurate modeling more challenging.

Adaptive MPC is one of the attempts to overcome some of these challenges 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 and Unscented Kalman Filters to simultaneously estimate states and model parameters (e.g., [13, 29]). Alternatively, others have resorted to Learning-based MPC [4], an approach that more gracefully handles model uncertainties and input/constraint bounds. Learning-based MPC also decouples robustness and performance by using a statistical model to optimize performance and a deterministic model to impose comfort constraints.

2.2 Reinforcement Learning for HVAC

The early works by [11, 25] 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 [11, 25].

To reduce sample complexity, researchers have adopted different approaches, such as injecting domain knowledge, pre-training the agent, and incorporating a model of system dynamics. For example, [38] initialized a Q-learning algorithm with an MPC prior. However, they assumed a overly simple grey-box model, which has an analytical MPC solution. Such is generally not the case for practical application. Alternatively, [18, 24] used historical data to pre-train the agent. Specifically, [24] populated the replay memory with historical data and [18] used historical state-action pairs for expert demonstrations. Finally, [10, 34] used models to assist RL. [10] used a neural network (NN) based model to supply additional state-action pairs. [34] used a NN as a model for state transitions and incorporated multi-step planning into RL. The experiments in [34] showed that model-based RL was generally superior to model-free RL in terms of sample efficiency, energy performance, and occupants' comfort.

Despite the numerous publications on this topic, applications beyond simulation are numbered. Of the publications referenced earlier, for example, [25, 53] deployed their RL agents in real-world testbeds. However, they both assumed the existence of high-fidelity models for training their agents in simulation. Such approach shifts the focus back to modeling [52] and is not scalable. Similarly, [10] also reported a real-world deployment, but provided no quantitative

Figure 1: The Agent-environment Interaction in a Markov Decision Process

results. More recently, Google announced a 40% energy consumption reduction in their data centers [12], but to our knowledge no technical publication of this achievement is available yet. It should be noted, though, that in the broader scope of building systems there are real-world applications of RL in systems other than HVAC, including water heaters [21] and lighting systems [36].

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 is cast as a Markov Decision Process (MDP), as shown in Figure 1.

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 (Eq. 1). In modern RL, the policy is commonly approximated by a NN parameterized by θ [31]. When the agent takes the action u , there is a state transition $X_{t+1} = x'$ based the system dynamics f_θ (Eq. 2) and the agent receives a reward $R_{t+1} = r$. In our problem formulation, the reward is the negative of a cost function (Eq. 3). Due to the Markov property, both the reward and the state transition depend only on the previous state. RL then finds a policy π_θ that maximizes the expected reward or, equivalently, minimizes the expected cost (Eq. 4).

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

$$x' \sim f_\theta(x, u) = \mathbb{P}(X_{t+1}|X_t = x, U_t = u) \quad (2)$$

$$r = -C(x, u) \quad (3)$$

$$\theta^* = \arg \max_{\theta} \mathbb{E}_{\pi_\theta}[r] = \arg \min_{\theta} \mathbb{E}_{\pi_\theta}[c] \quad (4)$$

There are three general approaches to RL, i.e. value-based methods, policy gradient methods, and actor-critic methods. Value-based methods, e.g. Q-learning and its variations, update the value function of state-action pairs and take the action that maximizes the Q-value with some level of exploration. Value-based methods have been applied to HVAC control in works such as [6, 10, 11, 26, 38, 52]. Policy gradients methods directly search for an optimal policy π_θ^* , using stochastic estimates of policy gradients. In this work, we used Proximal Policy Optimization (PPO) [43] as our online learning algorithm, which is elaborated in Section 4.3. Actor-critic methods, e.g. Advantage Actor-Critic (A2C), are hybrids of the value-based and policy gradient approaches, which use a policy network to select actions (the actor), and a value network to evaluate the action (the critic). Actor-critic methods have also been used for HVAC control (e.g., [24, 53]).

Figure 2: Overall Framework

3.2 Differentiable MPC

Since the success of [31], it is common to approximate the policy $\pi_\theta(u|x)$ with a NN. However, a generic NN does not encode any domain knowledge on planning or system dynamics, which is abundant in existing HVAC control literature. Encoding such knowledge in the policy would expedite the learning process.

In this paper, we took advantage of a newly developed Differentiable MPC policy [2] in place of a generic NN. In the forward pass, the Differentiable MPC policy solves a standard MPC problem. In the backward pass, the differentiable nature of the policy allows us to update the model parameters end-to-end. Specifically, MPC is a constrained optimization problem and the Differentiable MPC policy solves the Karush-Kuhn-Tucker (KKT) conditions of the problem in a computationally efficient manner, using the techniques developed in [3].

The Differentiable MPC policy solves this optimization problem:

$$\tau_{1:T}^* = \arg \min_{\tau_{1:T}} \sum_t C_{\theta,t}(\tau_t) \quad (5)$$

subject to

$$x_1 = x_{\text{init}}, x_{t+1} = f_\theta(\tau_t) \quad (6)$$

$$\underline{u} \leq u \leq \bar{u} \quad (7)$$

In other words, it finds the optimal trajectory, $\tau_{1:T}^* = \{x_t, u_t\}_{1:T}$, which minimizes the cost function, $C_{\theta,t}(\tau_t)$ over the time horizon $1 : T$ (Eq. 5). The optimal trajectory also satisfies system dynamics, f_θ , given in Eq. 6, and constraints on the control action, in Eq. 7.

4 APPROACH

The overall framework of our approach is shown in Figure 2. A Gnu-RL agent is a RL agent based on a Differentiable MPC policy, which encodes domain knowledge on planning and system dynamics. Training a Gnu-RL agent includes two phases: offline pre-training and online learning. In the offline pre-training phase, the agent is initialized by imitating the existing controller. In the online learning phase, the agent interacts with the environment and updates its policy using a policy gradient algorithm.

We first formulate the HVAC control problem, (Section 4.1) and then elaborate on the procedures used to train the agent during the pre-training phase (Imitation Learning, Section 4.2) and the online learning phase (Proximal Policy Optimization, Section 4.3).

4.1 Problem Formulation

We adapt the generic problem formulation for the Differentiable MPC policy (i.e., Eqs. 5-7) to the HVAC control problem. First, we define a cost function (Eq. 8) that uses the L2-norm of the state deviations from setpoint as a proxy for comfort, and the L1-norm of control actions as a proxy for energy consumption, and balances their relative importance with the hyperparameter η . In our experiments, we used different values of η for occupied and unoccupied periods. The Differentiable MPC allows for a learnable cost function $C_{\theta,t}$, but we assume the cost function to be specified by engineers.

$$C_t(\tau_t) = \frac{\eta}{2} \sum_{i=1}^{\# \text{ states}} (x_{t,i} - x_{i,\text{setpoint}})^2 + \sum_{i=1}^{\# \text{ actions}} |u_{t,i}| \quad (8)$$

We then use a linear model for the system dynamics, as shown in Eq. 9. Though building dynamics are non-linear, it is reasonable to assume that they are locally linear for the region of the state-space that we are interested in [15, 39]. Besides the state x_t and the control action u_t , we also consider uncontrollable disturbances d_t , such as weather and occupancy.

$$x_{t+1} = f_{\theta}(\tau_t) = Ax_t + B_u u_t + B_d d_t \quad (9)$$

The parameters A , B_u , and B_d are the only parameters we need to learn for the Differentiable MPC policy, i.e. $\theta = \{A, B_u, B_d\}$. Compared to using a NN-based policy, the number of free parameter is drastically reduced and the policy is interpretable to engineers. In both of our experiments, we consider the allowable range of control actions using Eq. 7. Also, at each time step t , we provide the agent with predictive information on disturbance for the planning horizon, i.e. $d_{t:t+T-1}$.

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 imitation learning and online policy learning. Moreover, since the HVAC control problems we are interested in have continuous action spaces, we re-parameterize the policy with a Gaussian distribution around the optimal action u_t^* (Eq. 10).

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

It should be noted that [2] 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 unsupervised and generally considered as a harder problem.

4.2 Imitation Learning

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 behaviour, compared to asking the expert to encode or fine-tune a policy [44]. 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. We minimized the mean squared error (MSE) loss between the trajectories of the expert and our agent. The loss function is given by Eq. 11, where x_t , u_t are the state and action for the expert, and $\hat{x}_t$, $\hat{u}_t$ are those for the learner. The hyperparameter λ balances the relative importance of states and actions. In particular, 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.

$$\mathcal{L}_{\text{imit}}(\theta) = \sum_t \lambda(x_t - \hat{x}_t)^2 + (u_t - \hat{u}_t)^2 \quad (11)$$

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 $i = 0, \dots, \# \text{ Episodes}$ **do**
 for $t = 0, \dots, \# \text{ Steps}$ **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, $\pi_{\hat{\theta}}$

4.3 Proximal Policy Optimization

We adopt a policy gradient approach for this paper, because it integrates naturally with the Differentiable MPC policy. Policy gradient methods achieve the objective in Eq. 4 by computing an estimate of the policy gradient and optimizing the objective through stochastic gradient ascent (Eq. 12).

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} \mathbb{E}_{\pi_{\theta}}[r] \quad (12)$$

REINFORCE [47] is the most commonly known policy gradient algorithm. However, it suffers from large performance variance and unstable policy updates [43]. Proximal Policy Optimization (PPO) [43] is the most recent approach in a long line of publications focused on improving upon vanilla policy gradient methods, including Natural Policy Gradients [19] and Trust Region Policy Optimization (TRPO) [42]. The intuition behind these methods is that in each update the policy π_{θ} cannot change too much from the policy before the update, $\pi_{\theta_{old}}$. PPO is known to be stable and robust to hyperparameters and network architectures. It was shown to outperform methods, such as A3C [30], Actor-Critic with Experience Replay (ACER) [50], and TRPO [42].

PPO minimizes the loss function in Eq. 13, which is equivalent to maximizing the Monte Carlo estimate of rewards with regularization. The advantage function $\hat{A}_t$ is given by Eq. 14 and the importance weighting $w_t(\theta)$ is given by Eq. 15. The procedures for

online learning with PPO are outlined in Algorithm 2. For more details, see [43].

$$\mathcal{L}_{PPO}(\theta) = -\hat{\mathbb{E}}_t [\min(w_t(\theta)\hat{A}_t, \text{clip}(w_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)] \quad (13)$$

$$\hat{A}_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \quad (14)$$

$$w_t(\theta) = \frac{\pi_{\theta}(u_t|x_t)}{\pi_{\theta_{old}}(u_t|x_t)} \quad (15)$$

Algorithm 2: Online Learning - PPO (Modified from [43])

Input: A pretrained policy, $\pi_{\hat{\theta}}$;

for $i = 0, \dots, \# \text{Episodes}$ **do**

$\theta_{old} \leftarrow \theta$;

for $t = 0, \dots, \# \text{Steps}$ **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

We trained and evaluated our agent using the EnergyPlus model from [53]. For offline pre-training, we used a baseline P-controller for expert demonstration and simulated the state-actions pairs under the Typical Meteorological Year 3 (TMY3) weather sequence [51], from Jan. 1st to Mar. 31st. We pre-trained 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 [53], our results are directly comparable. However, [53] assumed the existence of a high-fidelity model for training, while we only assumed the existence of historical data.

We took advantage of the simulation environment to experiment with and compare different approaches. For offline pre-training, we compared the performance of two different initialization schemes: system identification and imitation learning. System identification is the class of methods that estimate model parameters of a dynamic system based on input and output signals [27]. Specifically, prediction error methods (PEM) look for parameters that minimize the difference between predicted states and observed states. In addition to learning system dynamics, imitation learning also tries to minimize the difference between actions from the learner and those from the expert, which is of importance for a control problem.

For online learning, we compared two different objectives: minimizing prediction error and maximizing expected reward. 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 approaches such as Adaptive MPC, with the objective of minimizing prediction error [13, 29].

Figure 3: Simulation Testbed

Table 1: Variables in Simulation Experiment

X - State	D - Disturbance
Zone Temperature ($^{\circ}\text{C}$)	Outdoor Air Temperature ($^{\circ}\text{C}$)
U - Control Action	Outdoor Air Relative Humidity (%)
SW Temperature ($^{\circ}\text{C}$)	Diffuse Solar Radiation (W/m^2)
	Direct Solar Radiation (W/m^2)
	Occupancy Flag
	Wind Speed (m/s)

The difference in the objective functions of the two approaches motivated the comparison.

5.1 Simulation Testbed

The simulation testbed, as shown in Figure 3a was modeled after the Intelligent Workspace (IW) on Carnegie Mellon University (CMU) campus. The IW is a 600m² multi-functional space, including a classroom, a common area, and offices. We used the same EnergyPlus model used in [53], which was calibrated against operational data. In this experiment, we controlled the water-based radiant heating system. Figure 3b shows a schematic of the HVAC 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 $^{\circ}\text{C}$. The variables considered for this problem are listed in Table 1.

5.2 Implementation Details

We used the OpenAI Gym [9] wrapper for EnergyPlus developed in [53] to interface with the simulation environment. RL was implemented in PyTorch [37].³ The EnergyPlus model has a 5-minute simulation time step. Following [53], each action was repeated for 3 times (a 15-min control time step). The agent plans ahead for 12 steps (a 3-hour planning horizon). We considered each natural day as an episode and used min-max normalization to normalize all disturbance terms to 0-1. We provided the agent with ground truth information on future disturbances, i.e. weather and occupancy.

³The code is available at <https://github.com/INFERLab/Gnu-RL>.

Figure 4: Comparison of Two Initialization Schemes (Evaluated on TMY3 Weather Sequence)

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. Following [2], we used RMSprop [48] as the optimizer.

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} . We re-scaled the reward by 0.1, i.e. $\tilde{r} = 0.1r$, for better performance. For hyperparameters, we used $\gamma = 0.8$, $\epsilon = 0.1$, and $M = 48$.

5.3 Results: Offline Pretraining

Existing vs. Baseline P-controller: The existing P-controller for supply water temperature operates 24/7, which is not the intended behaviour for our agent and is not a fair comparison.⁴ 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 [39]. We assumed the same model (Eq. 9) and used the same time series for both initialization schemes. We evaluated the performance of the two initialization schemes by letting the pre-trained agents control the simulation environment under the TMY3 weather sequence, with fixed parameters. Figure 4 shows the behavior of the initialized agents over a five-day 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

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

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 [1]. In practice, excitation signals from actuators were often necessary to identify model parameters [1, 4, 40]. However, such procedure requires careful design of experiments [1] 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.

5.4 Results: Online Training

After pre-training, our agent controlled the environment using the actual weather sequence in 2017. The left hand side of Figure 5 shows the behaviour of Gnu-RL at the onset of training for a four-day 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 [53] took 47.5 years in simulation to achieve similar performance to the existing controller.

The results with comparison to [53] are tabulated in Table 2. The heating demand and predicted percent dissatisfied (PPD) were calculated by EnergyPlus. Similar to [53], we only considered PPD during occupied periods. Our agent saved 6.6% energy and better maintained comfort compared to the best performing RL agent in [53]. 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 5. 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 towards the end of occupancy. This explains the savings with respect to the baseline P-controller. It is worth noting that the preheating behaviour was not present in the baseline P-controller. The knowledge embedded in the Differentiable MPC policy enabled our agent to extrapolate beyond expert demonstration.

Table 2: Comparison of Performance during Online Learning Phase

	Heating Demand (kW)	PPD Mean (%)	PPD SD (%)
Existing P-Controller [53]	43709	9.45	5.59
Agent #6 [53]	37131	11.71	3.76
Baseline P-Controller	35792	9.71	6.87
Gnu-RL	34687	9.56	6.39
Gnu-RL + $\mathcal{L}_{PEM}$	24901	18.77	12.48

Our approach finds model parameters that maximize expected reward using a policy gradient algorithm. Alternatively, Adaptive MPC updates model parameters online by minimizing prediction

Figure 5: Performance of the Gnu-RL agent at the onset of deployment

error. We compare the performance of two different objectives: minimizing prediction error (Eq. 16) and maximizing expected reward (Eq. 13). 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.

$$\mathcal{L}_{PEM}(\theta) = \sum_t (x_t - \hat{x}_t)^2 \quad (16)$$

Figure 6 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 residue reward. While optimizing $\mathcal{L}_{PEM}$ resulted in consistently smaller prediction error, optimizing $\mathcal{L}_{PPO}$ resulted in larger overall reward. Table 2 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 [2]. 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 5.3. Another common observation from both comparisons is that it is highly-effective to directly optimize the task objective, whether it is imitation or control.

6 EXPERIMENT 2: REAL-WORLD DEPLOYMENT

Given the promising results in our simulation study, we repeated our experiment in a real-world conference room on CMU campus, during Jun. 5th-Jun. 25st, 2019 to validate that our approach can make real-world deployment of RL for HVAC control possible 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

Figure 6: Comparison of Two Objective Functions

are not necessarily the same as the actions taken, e.g. there is a 1-2 minute delay with the BAS interface. Finally, RL is sensitive to hyperparameters and other implementation details [17]. 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 5.2) unless specified otherwise, although the implementation details may not be optimal for this specific problem.

6.1 Testbed

The conference room is a 20m² single-zone space (Figure 7a) controlled by a variable air volume (VAV) box. Figure 7b shows a schematic of the HVAC 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

Figure 7: Real-world Testbed

coil, which was kept closed throughout the experiment for energy efficiency. The variables used in the problem is listed in Table 3. 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.

Table 3: Variables in Real-world Deployment

X - State	D - Disturbance
Zone Temperature ($^{\circ}F$)	Outdoor Air Temperature ($^{\circ}F$)
U - Control Action	Discharge Air Temperature ($^{\circ}F$)
Supply Airflow (CFM)	Occupancy Flag
	Occupancy Count

6.2 Implementation Details

The implementation details are the same as in Section 5.2, unless specified otherwise. We used the same 15-minute 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. We did not rescale reward.

For offline pretraining, we used a learning rate of 1×10^{-3} and a $\lambda=0.1$. For online learning, we used PI DataLink [35] 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 [33], 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 [45]. 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 $56^{\circ}F$ to $65^{\circ}F$. 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.

6.3 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.

6.4 Results: Online Training

Figure 8 shows how our agent’s behavior evolved over the three-week experiment period. Each snapshot shows the state-action pairs over a one-day period. Initially (6/10), our agent 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 8, 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 towards the end of 6/17.

Table 4: Summary of Results for Real-World Testbed

		Cooling Demand (kWh)	OAT Mean ($^{\circ}F$)	SD ($^{\circ}F$)	IAT RMSE ($^{\circ}F$)
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

The performance of our agent with comparison to the existing controller, which follows a fixed occupancy schedule from 6am to 10pm, is summarized in Table 4. 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 demand⁵ 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 [46]. 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.

⁵The cooling demand is calculated as $\dot{Q} = c\dot{m}\Delta T$, where $\dot{Q}$ is the cooling demand, c is the specific heat of air, $\dot{m}$ 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.

Figure 8: Performance of Online Learning

7 DISCUSSION AND CONCLUSIONS

We proposed Gnu-RL, a precocial RL agent endowed with a Differentiable MPC policy 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 a simulation study and a real-world deployment that Gnu-RL had good initial performance and continued to improve over time. In the simulation study, our agent saved 6.6% energy compared to the best performing RL agent in [53], while better maintaining occupants' comfort. In the real-world conference room, where Gnu-RL was deployed for a three-week period, it saved 16.7% of cooling demand compared to the existing controller while keeping better track of the temperature setpoint.

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. Furthermore, the requirement of historical data does not preclude the usage of this method on new buildings. Since, there are only a number of free parameters and these parameters have well-defined physical meaning, it is straightforward to initialize these parameters with engineering calculations. For instance, occupancy count was not initially considered in the model we used for real-world deployment, because the record from the occupancy sensor did not date back long enough for pre-training. However, we observed that the amount of cooling required varied drastically with the number of occupants. So, we added a new term for occupancy count and initialized the parameter with back-of-the-envelope calculations. The agent adjusted the parameter on its own.

We also compared our approach to alternatives in the simulation study. For offline pretraining, we compared imitation learning and system identification. While we obtained a reasonable model from system identification judging by the prediction error, its control performance was disappointing. On the other hand, we successfully initialized our agent with imitation learning, using only observational data. For online learning, we compared the performance of optimizing two different objectives, i.e. maximizing expected

reward and minimizing prediction error. It was clear that small prediction error did not translate to good control performance, and vice versa. A common observation from both comparisons is that prediction error is only a proxy for the task objective [2]. With advancements in automatic differentiation [7] and its integration with physical-based problems, it is now possible and effective to directly optimize task objectives end-to-end.

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. Similarly, we assumed that building systems can be locally linearized. The assumption worked for the problems we considered, but it may not extrapolate to more complex problems.

ACKNOWLEDGMENTS

This material is 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. First and foremost, we would like to thank Brandon Amos, whose prior work was the primary inspiration for us. Secondly, we are indebted to Zhiang Zhang for sharing his model with us. Finally, we appreciate Michael Frenak's help in facilitating the real-world deployment. *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.*

REFERENCES

- [1] Clarence Agbi, Zhen Song, and Bruce Krogh. 2012. Parameter identifiability for multi-zone building models. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 6951–6956.
- [2] Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J. Zico Kolter. 2018. Differentiable MPC for End-to-end Planning and Control. (2018).
- [3] Brandon Amos and J Zico Kolter. 2017. Optnet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 136–145.

- [4] A. Aswani, N. Master, J. Taneja, D. Culler, and C. Tomlin. 2012. Reducing Transient and Steady State Electricity Consumption in HVAC Using Learning-Based Model-Predictive Control. *Proc. IEEE* 100, 1 (Jan 2012), 240–253. <https://doi.org/10.1109/JPROC.2011.2161242>
- [5] Ercan Atam and Lieve Helsens. 2016. Control-oriented thermal modeling of multizone buildings: methods and issues: intelligent control of a building system. *IEEE Control Systems Magazine* 36, 3 (2016), 86–111.
- [6] Sajjad Baghaee and Ilkay Ulusoy. 2018. User comfort and energy efficiency in HVAC systems by Q-learning. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 1–4.
- [7] Atılım Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. 2018. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research* 18 (2018), 1–43.
- [8] S Benga, A Kelman, Francesco Borrelli, Russell Taylor, and Satish Narayanan. 2012. Model predictive control for mid-size commercial building hvac: Implementation, results and energy savings. In *Second international conference on building energy and environment*. 979–986.
- [9] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).
- [10] Giuseppe Tommaso Costanzo, Sandro Iacovella, Frederik Ruelens, Tim Leurs, and Bert J Claessens. 2016. Experimental analysis of data-driven control for a building heating system. *Sustainable Energy, Grids and Networks* 6 (2016), 81–90.
- [11] Konstantinos Dalamagkidis, Denia Kolokotsa, Konstantinos Kalaitzakis, and George S Stavrakakis. 2007. Reinforcement learning for energy conservation and comfort in buildings. *Building and Environment* 42, 7 (2007), 2686–2698.
- [12] Richard Evans and Jim Gao. 2016. DeepMind AI reduces Google data centre cooling bill by 40%. <https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/>. Accessed: 2019-06-19.
- [13] Samuel F Fux, Araz Ashouri, Michael J Benz, and Lino Guzzella. 2014. EKF based self-adaptive thermal model for a passive house. *Energy and Buildings* 68 (2014), 811–817.
- [14] William Goetzler, Richard Shandross, Jim Young, Oxana Petritchenko, Decker Ringo, and Sam McClive. 2017. *Energy Savings Potential and RD&D Opportunities for Commercial Building HVAC Systems*. Technical Report. Navigant Consulting, Burlington, MA (United States).
- [15] Tomasz G Gorecki, Faran A Qureshi, and Colin N Jones. 2015. OpenBuild: An integrated simulation environment for building control. In *2015 IEEE Conference on Control Applications (CCA)*. IEEE, 1522–1527.
- [16] Rob Guglielmetti, Dan Macumber, and Nicholas Long. 2011. *OpenStudio: an open source integrated analysis platform*. Technical Report. National Renewable Energy Lab.(NREL), Golden, CO (United States).
- [17] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- [18] Ruoxi Jia, Ming Jin, Kaiyu Sun, Tianzhen Hong, and Costas Spanos. 2019. Advanced Building Control via Deep Reinforcement Learning. *Energy Procedia* 158 (2019), 6158–6163.
- [19] Sham M Kakade. 2002. A natural policy gradient. In *Advances in neural information processing systems*. 1531–1538.
- [20] Sham Machandranath Kakade et al. 2003. *On the sample complexity of reinforcement learning*. Ph.D. Dissertation.
- [21] Hussain Kazmi, Fahad Mehmood, Stefan Lodeweyckx, and Johan Driesen. 2018. Gigawatt-hour scale savings on a budget of zero: Deep reinforcement learning based optimal control of hot water systems. *Energy* 144 (2018), 159–168.
- [22] M Killian and M Kozek. 2016. Ten questions concerning model predictive control for energy efficient buildings. *Building and Environment* 105 (2016), 403–412.
- [23] Xiwang Li and Jin Wen. 2014. Review of building energy modeling for control and operation. *Renewable and Sustainable Energy Reviews* 37 (2014), 517–537.
- [24] Yuanlong Li, Yonggang Wen, Kyle Guan, and Dacheng Tao. 2017. Transforming cooling optimization for green data center via deep reinforcement learning. *arXiv preprint arXiv:1709.05077* (2017).
- [25] Simeng Liu and Gregor P Henze. 2006. Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory: Part 2: Results and analysis. *Energy and buildings* 38, 2 (2006), 148–161.
- [26] Simeng Liu and Gregor P Henze. 2007. Evaluation of reinforcement learning for optimal control of building active and passive thermal storage inventory. *Journal of solar energy engineering* 129, 2 (2007), 215–225.
- [27] Lennart Ljung. 1999. System identification. *Wiley Encyclopedia of Electrical and Electronics Engineering* (1999), 1–19.
- [28] Xiaoshu Lü, Tao Lu, Charles J Kibert, and Martti Viljanen. 2015. Modeling and forecasting energy consumption for heterogeneous buildings using a physical-statistical approach. *Applied Energy* 144 (2015), 261–275.
- [29] Mehdi Maasoumy, M Razmara, M Shahbakhti, and A Sangiovanni Vincentelli. 2014. Handling model uncertainty in model predictive control for energy efficient buildings. *Energy and Buildings* 77 (2014), 377–392.
- [30] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. 1928–1937.
- [31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
- [32] Christopher Moyer. 2016. How Google's AlphaGo beat a Go world champion. *The Atlantic* 28 (2016).
- [33] Sirajum Munir, Ripudaman Singh Arora, Craig Hesling, Juncheng Li, Jonathan Francis, Charles Shelton, Christopher Martin, Anthony Rowe, and Mario Berge. 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)*. IEEE, 295–306.
- [34] Adam Nagy, Hussain Kazmi, Farah Cheaib, and Johan Driesen. 2018. Deep Reinforcement Learning for Optimal Control of Space Heating. *arXiv preprint arXiv:1805.03777* (2018).
- [35] OSISOft. [n. d.]. Tools : PI DataLink. <https://www.osisoft.com/pi-system/pi-capabilities/pi-system-tools/pi-datalink/>. Accessed: 2019-06-19.
- [36] June Young Park, Thomas Dougherty, Hagen Fritz, and Zoltan Nagy. 2019. LightLearn: An adaptive and occupant centered controller for lighting based on reinforcement learning. *Building and Environment* 147 (2019), 397–414.
- [37] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).
- [38] Kuo Shiuang Peng and Clayton T Morrison. 2016. Model Predictive Prior Reinforcement Learning for a Heat Pump Thermostat. In *IEEE International Conference on Automatic Computing: Feedback Computing*, Vol. 16.
- [39] Samuel Privara, Jiří Cigler, Zdeněk Váňa, Frauke Oldewurtel, Carina Sagerschnig, and Eva Žáčková. 2013. Building modeling as a crucial part for building predictive control. *Energy and Buildings* 56 (2013), 8–22.
- [40] Samuel Privara, Zdeněk Váňa, Dimitrios Gyalistras, Jiří Cigler, Carina Sagerschnig, Manfred Morari, and Lukáš Ferkl. 2011. Modeling and identification of a large multi-zone office building. In *2011 IEEE International Conference on Control Applications (CCA)*. IEEE, 55–60.
- [41] Peter Rockett and Elizabeth Abigail Hathway. 2017. Model-predictive control for non-domestic buildings: a critical review and prospects. *Building Research & Information* 45, 5 (2017), 556–571.
- [42] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning*. 1889–1897.
- [43] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
- [44] David Silver, J Andrew Bagnell, and Anthony Stentz. 2010. Learning from demonstration for autonomous navigation in complex unstructured terrain. *The International Journal of Robotics Research* 29, 12 (2010), 1565–1592.
- [45] Dark Sky. [n. d.]. Dark Sky API. <https://darksky.net/dev>. Accessed: 2019-06-19.
- [46] Energy Star. 2018. Portfolio Manager Technical Reference: Climate and Weather. <https://portfoliomanager.energystar.gov/pdf/reference/Climate%20and%20Weather.pdf>. Accessed: 2019-06-19.
- [47] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.
- [48] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.
- [49] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. 2017. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782* (2017).
- [50] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. 2016. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224* (2016).
- [51] Stephen Wilcox and William Marion. 2008. *Users manual for TMY3 data sets*. National Renewable Energy Laboratory Golden, CO.
- [52] Lei Yang, Zoltan Nagy, Philippe Goffin, and Arno Schlüter. 2015. Reinforcement learning for optimal control of low exergy buildings. *Applied Energy* 156 (2015), 577–586.
- [53] Zhiang Zhang and Khée Poh Lam. 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)*. ACM, New York, NY, USA, 148–157.