Accelerate Online Reinforcement Learning for Building HVAC Control with Heterogeneous Expert Guidances

Shichao Xu
Northwestern University
Evanston, USA
shichaoxu2023@u.northwestern.edu

Yangyang Fu
Texas A&M University
College Station, USA
yangyang.fu@tamu.edu

Yixuan Wang
Northwestern University
Evanston, USA
yixuanwang2024@u.northwestern.edu

Zhuoran Yang
Yale University
Evanston, USA
zhuoranyang.work@gmail.com

Zheng O'Neill
Texas A&M University
College Station, USA
zoneill@tamu.edu

Zhaoran Wang
Northwestern University
Evanston, USA
zhaoran.wang@northwestern.edu

Qi Zhu
Northwestern University
Evanston, USA
qzhu@northwestern.edu

ABSTRACT

Building heating, ventilation, and air conditioning (HVAC) systems account for nearly half of building energy consumption and 20% of total energy consumption in the US. Their operation is also crucial for ensuring the physical and mental health of building occupants. Compared with traditional model-based HVAC control methods, the recent model-free deep reinforcement learning (DRL) based methods have shown good performance while do not require the development of detailed and costly physical models. However, these model-free DRL approaches often suffer from long training time to reach a good performance, which is a major obstacle for their practical deployment. In this work, we present a systematic approach to accelerate online reinforcement learning for HVAC control by taking full advantage of the *knowledge from domain experts in various forms*. Specifically, the algorithm stages include learning expert functions from existing abstract physical models and from historical data via offline reinforcement learning, integrating the expert functions with rule-based guidelines, conducting training guided by the integrated expert function and performing policy initialization from distilled expert function. Experimental results demonstrate up to 8.8X speedup over previous DRL-based methods.

CCS CONCEPTS

• **Computing methodologies** → **Reinforcement learning**; • **Computer systems organization** → **Embedded and cyber - physical systems**.

BuildSys '22, November 9–10, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9890-9/22/11…$15.00
https://doi.org/10.1145/3563357.3564064

KEYWORDS

HVAC control, Reinforcement learning, Deep learning

1 INTRODUCTION

Buildings account for around 40% of the energy consumption in the United States, of which nearly half is by the heating, ventilation, and air conditioning (HVAC) systems [32]. In addition, the operation of HVAC systems significant affects the physical and mental health of building occupants, as people spend around 87% of their time indoors [20, 46], and even higher during the COVID-19 pandemic in recent years [16]. It is thus a critical task to develop effective HVAC control strategies that can maintain a comfortable indoor environment while reducing energy cost [31, 42, 45].

In the literature, there are extensive works of developing *model-based approaches* for HVAC control. For instance, [26] uses RC-networks to model the building thermal dynamics and applies the linear quadratic regulator (LQR) method for controlling the HVAC system. [37] designs a model predictive control (MPC) method to minimize the energy consumption and cost of the building HVAC system combined with a solar power unit. Some other works on model-based approaches can be found in [25, 27, 35, 44]. However, to achieve good performance, these model-based approaches require the development of detailed and accurate physical models, which are often difficult and costly in practice. Thus, there has been significant interest in developing *learning-based, model-free approaches* for HVAC control, in particular those based on deep reinforcement learning (DRL). For example, [41] utilizes the deep Q-learning method for controlling the indoor air flow rate and leverages the EnergyPlus platform [6] for simulation-based training. And various other techniques have also been applied for DRL-based building HVAC control, including Deep Deterministic Policy Gradient (DDPG) [11], Proximal Policy Optimization (PPO) [1], Asynchronous Advantage Actor-Critic (A3C) [51], etc.

However, a major difficulty in adopting DRL-based methods for building HVAC control is that it could take a long time to train the RL agent in practice during building operation. For instance, it may

take more than 100 months of training to reach convergence for the Q-learning based methods in [7, 41], and around 500 months of training for the DDPG algorithm in [12] to converge on a laboratory building model. In [48], DDPG is used for temperature control and energy management, and it takes around 2.4×10^4 months to reach the best performance. In [47], the training time is almost 4×10^4 months in their multi-zone building environment. Clearly, such long training time would make it impossible to adopt DRL in practice for building control. While developing a detailed simulation model (e.g., in EnergyPlus) and conducting the training via simulation may help avoid this issue, the development of the simulation model itself is difficult and costly (in terms of both time and expertise), just as in the model-based methods.

Thus, recently researchers have been trying to improve the training efficiency for DRL-based building HVAC control. In [46], a transfer learning approach is proposed to extract and transfer the building-agnostic knowledge from an existing DRL controller of a source building to a new DRL controller of a target building, and only re-train the building-specific components for the new DRL controller. The work in [23] also leverages transfer learning, but for heat pump control in microgrid. However, the effectiveness of the transfer learning-based methods strongly relies on the similarity between the existing building target building and the transferred building, and may not be feasible when they are not similar [46]. There are also a few studies on the application of offline reinforcement learning for building HVAC control, where historical data on existing controllers are leveraged to train new RL-based controllers. For instance, [36] conducts conservative Q-learning (CQL) to train controllers for maintaining the room temperature setpoint. The problem of such offline RL methods, however, is that the learned agents' performance strong depends on the quality of the historical data. They tend to perform poorly due to the distributional shift between the historical data and the learned policy, and may have limited improvement even with fine tuning via online training [30].

In this work, to address the above challenge in DRL training efficiency, we propose a **unified framework that leverages the knowledge from domain experts in various forms** to accelerate online RL for building HVAC control. This is motivated by the observation that in established domains such as building control, there is extensive domain expertise, represented in various forms such as 1) *abstract physical models* (e.g., RC-networks [24] or ARX models [49]) of building thermal dynamics – they are not accurate enough for enabling training DRL or designing model-based methods with good performance, but nevertheless contain valuable information of building dynamics, 2) *historical data* collected from existing controllers – they may not be able to train DRL controllers with good performance due to distribution shift, but also contain useful information on building behavior, and 3) expert rules that reflect basic policies. We believe that leveraging these domain expertise can help accelerate the online RL process. In particular, our framework first learns *expert functions* from existing abstract physical models and from historical data via offline RL, and then combines those with expert rules to generate an *integrated expert function*, which will then be used to drive online RL with prior-guided learning and policy initialization from expert function distillation. In experiments, our framework is able to significantly reduce the convergence time for DRL training by up to 8.8X, while maintaining similar performance (in terms temperature violation rate and energy cost).

To summarize, our work makes the following contributions:

- We propose a novel framework to accelerate online RL for building HVAC control with heterogeneous expert guidances, including abstract physical models, historical data, and expert rules. These various guidances are unified in our framework via the expert functions.
- We conducted a series of experiments for evaluating the effectiveness of our framework. The results demonstrate that our approach can effectively reduce the DRL training time while maintaining good performance.

The rest of the paper is organized as follows. Section 2 introduces the related literature. Section 3 presents our approach, and Section 4 presents the experimental results. Section 5 concludes the paper.

2 RELATED WORKS

Reinforcement Learning for HVAC Control: Building HVAC control is a critical and challenging problem as it significantly affects both building energy efficiency and occupants' physical and mental health. In traditional model-based approaches, detailed and accurate physical models are needed for control optimization, but are often difficult and costly to develop and slow to run. Such limitations have motivated the exploration of model-free approaches in recent years, particularly those based on deep reinforcement learning [41, 46, 51, 52]. These DRL-based HVAC control approaches leverage a variety of RL algorithms including DQN [41], A3C [51], DDPG [11], PPO [1], etc. For instance, Wei *et al.* [41] convert the building HVAC control into a Markov decision process (MDP) problem and leverage the DQN method to intelligently learn the operation strategy based on offline simulations. Gao *et al.* [11] adopt the neural network to predict occupants' thermal comfort for part of their reward function design, and then apply the standard DDPG algorithm to learn from their building simulation environment. Abrazeh *et al.* [1] develop a real-time digital twin with a PPO-based backstepping controller to maintain the relative humidity and temperature in buildings. However, a major obstacle in applying these DRL-based control algorithms is that they often require *dozens of months or more* for training to reach the desired performance [7, 11, 12]. Such long time is clearly not feasible for direct training during real building operation (i.e., sensing the real building environment and sending the actuation signals to HVAC equipment). It may be possible to avoid this by developing accurate and detailed building models and conducting training via simulations on tools such as EnergyPlus and Modelica [28]-based tools [7], however, this again requires the development of those detailed and costly physical models and somewhat defeats the original purpose of using model-free approaches. Thus, it is critical to improve the efficiency of online RL for HVAC control *without* the development of detailed physical models.

Transfer Learning for HVAC Control: One way to speed up RL is to transfer the learned policy between different buildings. For instance, [46] reduces the DRL training time by re-designing the learning objective and decomposing the neural network to a building-agnostic sub-network and a building-specific sub-network. The

building-agnostic sub-network can be directly transferred from an existing DRL controller of a source building, and only the building-specific sub-network needs to be (re)-trained on the target building. This can reduce the DRL training time from months/years to weeks. [23] utilizes the direct policy transfer between different houses with the same state/action space for heat pump control in microgrids. [50] applies the transfer learning to a PPO-based controller for smart home to reduce the training cost. The main limitations of these approaches is that the effectiveness of the transfer strongly relies on the similarity between the source and the target buildings. When the buildings are not similar or not operating in similar environment, the transfer may have poor performance [46].

Offline Reinforcement Learning: Another way to accelerate online RL is through *offline RL*, by leveraging historical data collected under existing control policies. Recent offline RL works focus on two aspects: offline policy optimization, and offline policy evaluation. The former aims to learn an optimal policy for maximizing a notion of cumulative reward, while the latter is intended to evaluate the accumulated reward (or the value function) of a given policy.

For offline policy optimization in particular, a major challenge is that the agent cannot directly explore the environment. And the error (called extrapolation error [10]) that is caused by selected actions not contained in the historical dataset could occur and propagate during the training. This is one of the reasons that limits the effectiveness of existing offline RL approaches for building HVAC control [36]. The approaches that address this challenge mainly utilize regularization or constraint-based methods to help the policy stay near to the existing actions in the historical dataset. For instance, the batch-constrained Q-learning (BCQ) approach [10] restricts its action space to make the learned behavior similar to the actions in the historical dataset. [17] penalizes divergence between the prior learned from the historical dataset and the Q-network policy using KL-control. [40] learns the policy by filtered behavioral cloning, which utilizes critic-regularized regression to filter out low-quality actions. And other related investigations can be found in [2, 4, 8, 9, 13, 22]. And from the prior experiments, we notice that not all offline RL algorithms can be chosen for building the expert function. The method like TD3+BC [9] may not always provide a good value estimation for the given states, as it only aims to make the learned policy closer to the behavior in the offline dataset and tend to overestimate the Q-value. So in this work, we use historical data as one of the expert guidance and conduct offline RL to build an expert function. We leverage the idea from [21] to estimate the value function from historical dataset because of its effectiveness, by directly setting regularization on the Q-function and generating the Q-value estimation in a conservative way to reduce overestimation.

3 OUR PROPOSED FRAMEWORK

3.1 System Model

We use the building model with the fan-coil system from [7], which is extended from a single-zone commercial building with manipulable internal thermal mass. The internal air is conditioned by an idealized fan coil unit (FCU) system, and the fan airflow rate is chosen from multiple discrete levels $\{f_1, f_2, \cdots, f_m\}$ (which can be viewed as m control actions; f_1 is to turn off the cooling system, and f_m is to run it at full speed.). There are two different working modes in this system: the occupied time (daily from 7 am to 7 pm), and the unoccupied time (rest of the day). The HVAC system will run in a low-power mode during the unoccupied time for the energy-saving purpose (with the cooling system almost turned off). And the setting of comfortable temperature bound is different in these two modes. The system conducts control with a period of Δt. Each training episode contains 2 days data, so there are $\frac{2880}{\Delta t}$ control steps in each episode. Other experiment-related settings can be found in Section 4.1. The system state contains the following elements:

- Current physical time t,
- Indoor air temperature T_t^{in},
- Outdoor air temperature T_t^{env},
- Solar irradiance intensity q_t^{sun},
- Power consumption during the current control interval P_t,
- Outdoor air temperature forecast in the next three control steps $\{T_{t+1}^{env}, T_{t+2}^{env}, T_{t+3}^{env}\}$, and
- Solar irradiance intensity forecast in the next three control steps $\{q_{t+1}^{sun}, q_{t+2}^{sun}, q_{t+3}^{sun}\}$.

One thing to note is that we add one additional variable in the implementation to the system state design, which is the remainder after dividing the current physical time t by $24 * 60 * 60$. This is to help the RL agent figure out the time position within one day (morning, noon, afternoon, etc.), and may help it reach better performance as observed in our preliminary experiments.

3.2 Our Online DRL Framework with Heterogeneous Expert Guidances

As stated in Section 1, to accelerate online DRL for HVAC control, we propose a unified framework that leverages heterogeneous expert guidances including abstract physical models, historical data, and expert rules. Figure 1 shows the overview of our framework design. Specifically, the framework includes the following major components:

- An expert function h_u learned from an expert model. The expert model could be an abstract physical model developed by domain experts (commonly exists in building domain), or in case such physical model is not available, a neural network with its parameters determined from historical data (but different from offline RL; more details later).
- Another expert function h_o learned via offline RL on historical data that was collected using existing controllers.
- An integrated expert function h by combining h_u and h_o as well as expert rules.
- Application of prior-guided learning and policy initialization from expert function distillation based on h.

The detailed flow of our approach is shown in Algorithm 1. Next, we will first explain the underlying DRL algorithm we use, and then introduce the details of each component in our approach to improve the DRL efficiency with heterogeneous expert guidances.

Underlying DRL algorithm: Similarly as in recent works [7, 41, 46], we utilize double Deep Q-learning (DDQN) [39] as the underlying DRL algorithm for our framework and also the baseline

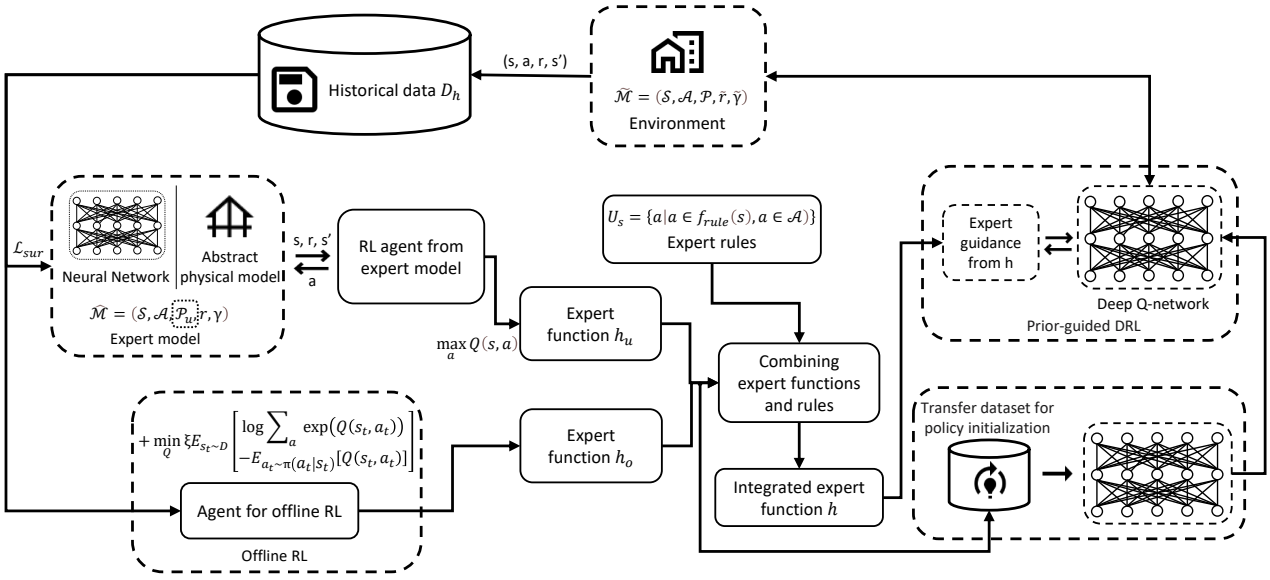


Figure 1: Overview of our online DRL framework with heterogeneous expert guidances. The framework includes the following major components: (1) An expert function h_u learned from an expert model, which can be an abstract physical model or a neural network with its parameters determined from historical data. (2) Another expert function h_o learned from offline RL based on historical data. (3) An integrated expert function h generated by combining h_u and h_o as well as expert rules. (4) Application of prior-guided learning and policy initialization from expert function distillation based on h.

Algorithm 1 Our Online DRL Framework with Heterogeneous Expert Guidances

n_{ep1}, n_{ep2}: number of training epochs
n_{max}: maximum training time of an epoch
n_{tar}: time interval to update target network
Randomly initialize Q-network Q
Learn expert function h_u from expert model using Algorithm 2
Learn expert function h_o from offline RL using Algorithm 3
Generate integrated expert function h from h_u, h_o and expert rules, following Equation (10)
Calculate initialization dataset D_{init}^{y} by h_u, h_o
Train Q-network Q by loss function $\mathcal{L}_{init}$
for $Epoch$ = 1 to n_{ep2} **do**
  Reset building environment Env
  for t = 0 to n_{max} **do**
    Select action a_t using epsilon-greedy
    $s_t, s_{t+1}, r_t \leftarrow Env.execute(a_t)$
    Update $\lambda \leftarrow \lambda_0 + (1 - \lambda_0) \tanh(\alpha_\lambda((Epoch - 1) * n_{max} + t))$
    $\tilde{r}_t = r_t + (1 - \lambda)\gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[h(s')]$, $\tilde{\gamma} = \lambda\gamma$
    Add transition $(s_t, s_{t+1}, a_t, \tilde{r}_t)$ to replay buffer
    Randomly sample a batch $B = (\mathcal{S}, \mathcal{S}', \mathcal{A}, \mathcal{R})$ from replay buffer
    Update Q-network Q with B and $\tilde{\gamma}$
    Update target network Q' with interval n_{tar}
  end for
end for

method for comparison in our experiments. We choose DDQN mainly for its convenience in leveraging the value function and the good performance it has shown for HVAC control in those recent works, but our expert-guidance approach can also be applied to improve the efficiency for many other DRL algorithms.

We assume that the next state of the building HVAC system only relies on the current system state, and thus HVAC control can be treated as a Markov decision process (MDP). As stated in Section 3.1, the state $s = (t, T_t^{in}, T_t^{env}, q_t^{sun}, P_t, T_{t+1}^{env}, T_{t+2}^{env}, T_{t+3}^{env}, q_{t+1}^{sun}, q_{t+2}^{sun}, q_{t+3}^{sun})$. The discrete action space $\mathcal{A}$ contains the normalized air flow rate (0 to 1) with $m - 1$ intervals. The reward is designed with consideration of indoor temperature violation and energy cost, as shown below:

$$r_t = \alpha \cdot \epsilon_t + \beta \cdot c_t, \tag{1}$$

where ϵ_t represents the temperature violation for the current time step, c_t is the energy cost for the current time step, and α, β are the scaling factors. More specifically, ϵ_t is defined as:

$$\epsilon_t = \max(T_i^{in} - T_{upper}, 0) + \max(T_{lower} - T_i^{in}, 0), \tag{2}$$

where T_{upper} is the upper bound of a given comfortable temperature range (which could be based on standards such as ASHRAE [34] or OSHA [33]) and T_{lower} is the lower bound. Moreover:

$$c_t = p_t P_t, \tag{3}$$

where p_t is the energy price at time t, and P_t is the power consumption during the current control interval at time t.

The goal of the DRL is to minimize total energy cost while maintaining indoor temperature within the comfortable temperature range. The loss function $\mathcal{L}_Q$ for updating the Q-network is:

$$\mathcal{L}_Q = \mathbb{E}_{(s_t, a_t, s_t') \sim D}\left[(r_t + \gamma \max_{a_{t+1}} Q'(s_{t+1}, a_{t+1}) - Q(s, a))^2 \right], \tag{4}$$

where $s_t, s_{t+1} \in \mathcal{S}$, $a_t \in \mathcal{A}$, Q is the Q network and Q' is the target Q network. Then, the components introduced in the rest of this section will generate expert functions to provide prior guidance and policy initialization for this underlying DRL algorithm.

Algorithm 2 Learning Expert Function from Expert Model

1: n_{ep1}: number of training epochs
2: n_{max}: maximum training time of an epoch
3: n_{tar}: time interval to update target network
4: Randomly initialize Q-network Q_u
5: Prepare input samples and corresponding labels $\{x^*, y^*\}$ from historical dataset D_h for training an expert model
6: Train expert model Env_u using dataset $\{x^*, y^*\}$ and loss function $\mathcal{L}_u$
7: **for** $Epoch$ = 1 to n_{ep1} **do**
8: Reset building environment Env
9: **for** t = 0 to n_{max} **do**
10: Select action a_t using epsilon-greedy
11: $s_t, s_{t+1}, r_t \leftarrow Env_u.execute(a_t)$
12: Add transition (s_t, s_{t+1}, a_t, r_t) to replay buffer RB_h
13: Randomly sample a batch $B = (\mathcal{S}, \mathcal{S}', \mathcal{A}, \mathcal{R})$ from RB_h
14: Update Q-network Q_u with B and $\tilde{\gamma}$
15: Update target network Q_u' with interval n_{tar}
16: **end for**
17: **end for**
18: Set expert function h_u using Q_u

Learning Expert Function h_u from Expert Model: An expert function h_u can be learned through an expert model. In many cases, such expert model already exists in the form of an abstract physical model for the building thermal dynamics, e.g., an ARX or RC-networks model. While these abstract models are typically not accurate enough to enable good performance for DRL or model-based methods, they can be effectively leveraged to generate an expert function.

If an abstract physical model is not available, we can build a neural network as the expert model, with its parameters decided from historical data collected under existing control policy, as shown in Algorithm 2 (Line 5 in Algorithm 1) and described in the following.

We denote the historical dataset as D_h, with n data samples. For each data sample $(x, y) \in D_h$, let input $x = \{t, T_t^{in}, T_t^{env}, q_t^{sun}, P_t, T_{t+1}^{env}, T_{t+2}^{env}, T_{t+3}^{env}, q_{t+1}^{sun}, q_{t+2}^{sun}, q_{t+3}^{sun}, a\}$ as defined in Section 3.1 and $a \in \mathcal{A}$, and let output label $y = \{T_{t+1}^{in}\}$. The neural network-based expert model consists of m_u fully-connected layers. All hidden layers are followed by a GELU activation function [14], and are sequentially connected (the detailed layer setting will be specified later in Table 1 of Section 4). As different variables may not be in the same order of magnitude (e.g., t can be 1000 times larger than T_t^{in}), we normalize the input x and the output label y. The preprocessed input and output can be written as $x^* = \frac{x - x_l}{x_h - x_l}$, $y^* = \frac{y - y_l}{y_h - y_l}$, where x_h and x_l are the upper bound and lower bound of the variable x, and y_h and y_l are the upper and lower bound of the variable y. We then train the expert model with a mean square error loss function

$$\mathcal{L}_u = \| y^* - y^*_{pred} \|^2, \tag{5}$$

where y^*_{pred} is the network prediction for the normalized y. When we apply this expert model after model training, we obtain the prediction of y by reversing the operation of previous-mentioned normalization step. Note that it may not be necessary to predict the entire system state. For example, the environment temperature T_t^{out} and solar irradiance q_t^{sun} may be obtained from weather forecast.

Once we have the expert model, either in the form of an abstract physical model or a neural network, the expert function h_u can be viewed as a prior guess of the optimal value function in the building HVAC control task and can be learned via DRL. More specifically, we define an MDP problem $\hat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_u, r, \gamma)$ where the definitions of state $\mathcal{S}$, action space $\mathcal{A}$ and reward function r are the same as defined at the beginning of Section 3.2. $\mathcal{P}_u$ is from the expert model. We then apply DDQN on $\hat{\mathcal{M}}$ and obtain a trained Q-network Q. And the expert function h_u can be set up as:

$$h_u(s) = \max_a Q(s, a), \tag{6}$$

where s is the state and a is the control action.

Learning Expert Function h_o from Offline RL: Another type of expert function h_o can be learned from the historical data via offline RL, as shown in Algorithm 3 (Line 6 in Algorithm 1). We leverage some of the techniques from conservative Q-learning (CQL) [21] because of its effectiveness in reducing a large number of hyper-parameters.

Algorithm 3 Learning Expert Function from Offline RL

1: n_{ep1}: number of training epochs
2: n_{max}: maximum training time of an epoch
3: n_{tar}: time interval to update target network
4: Randomly initialize Q-network Q_o
5: **for** $Epoch$ = 1 to n_{ep1} **do**
6: **for** t = 0 to n_{max} **do**
7: Randomly sample a batch $B = (\mathcal{S}, \mathcal{S}', \mathcal{A}, \mathcal{R})$ from D_h
8: Update Q-network Q_o with B and $\tilde{\gamma}$ following Equation 8
9: Update target network Q_o' with interval n_{tar}
10: **end for**
11: **end for**
12: Set expert function h_o using the learned Q-networks Q_o

First, we build an offline RL model based on DDQN, but with different Q-network updating rules as the DRL presented in the beginning of Section 3.2. In particular, compared with Equation (4), we add an extra regularization term:

$$\mathcal{L}_{reg} = \min_Q \mathbb{E}_{s_t \sim D} \left[\log \sum_{a_t} \exp(Q(s_t, a_t)) - \mathbb{E}_{a_t \sim \pi(a_t|s_t)} [Q(s_t, a_t)] \right], \tag{7}$$

where $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$. Q is the Q-network, and D is the dataset produced by the behaviour policy π. In the equation, the first part $\log \sum_{a_t} \exp(Q(s_t, a_t))$ describes a penalty term for minimizing the Q-value of the action produced by current policy on the states in the historical dataset. It helps learn a smaller and more conservative Q-value estimator. The second term $-\mathbb{E}_{a_t \sim \pi(a_t|s_t)} [Q(s_t, a_t)]$ counts average Q-value in the state-action pairs in the historical dataset and maximizes it to push the current learned policy closer to the behavior policy in the historical dataset.

Then the policy updating is changed as follows:

$$\mathcal{L}_{off} = \frac{1}{2} \mathbb{E}_{(s_t, a_t, s_t') \sim D} \left[(r_t + \gamma \max_{a_{t+1}} Q'(s_{t+1}, a_{t+1}) - Q(s_t, a_t))^2 \right] + \xi \mathcal{L}_{reg}, \tag{8}$$

where $s_t, s_{t+1} \in \mathcal{S}$, ξ is a mixing coefficient, and Q' is the target Q-network. With enough training iterations, the offline RL agent can

provide a good expert function h_o following the same procedure as in Equation (6).

Note that we observe that not all offline RL algorithms can be a suitable choice for our framework. For example, approaches like TD3+BC [9] may not always provide a good value estimation for the given states. We suspect that this may be due to two factors. One is related to the reward design, as the value function estimation in some offline RL algorithms is sensitive to the scale of the accumulated reward. The other is that because algorithms like TD3+BC only add regularization on the actor updating and do not set constraints on the Q function, which could enlarge the error in estimating the (Q-)value function when combined with possible numerical issues.

Generating Integrated Expert Function h from h_u, h_o and Expert Rules: The expert function h_u learned from the expert model and the expert function h_o learned via offline RL tend to perform differently because of the complexity of the system dynamic and the sufficiency of the data. Moreover, the accuracy of their Q-value estimation can vary at different states depending on the data distribution within the historical dataset. Thus, it is a natural thought to form an ensemble of the two. And the ensemble of multiple expert functions calculated in different ways can further reduce the overestimation of Q-values through a conservative way, which we will introduce in this section later.

To begin with, after having h_u and h_o, we can combine them with *expert rules* to generate an integrated expert function h. The expert rules are often set by domain experts or building operators based on past experience and domain expertise. They do not provide an optimized control action for a given state, but instead offer suggestions that could be viewed as guidance or soft constraints – e.g., not turning on the cooling system when the indoor temperature is below the lower bound of the comfortable temperature range by certain threshold. Formally, we define that the expert rules f_{rule} can generate an action candidate set $\mathbf{U}_s$ for each state:

$$\mathbf{U}_s = \{a | a \in f_{rule}(s), a \in \mathcal{A})\}. \tag{9}$$

We can then generated an integrated expert function h based on $\mathbf{U}_s$, h_u and h_o (Line 7 in Algorithm 1). Specifically, we apply a pessimistic ensemble strategy for selecting the value function estimation among different expert functions, and only choose corresponding actions from the expert rules' action candidate set $\mathbf{U}_s$. Thus, the integrated expert function h can be formulated as:

$$h(s) = \min_i (\max_{a \in \mathbf{U}_s} Q_i(s, a)), \tag{10}$$

where Q_i is the Q-value estimation from expert functions i. Note that this is a *general formulation* that can unify multiple expert functions – e.g., we may have more than one abstract physical models that provide multiple h_u expert functions.

Prior-guided Learning: Once we have the integrated expert function h, we can use it to guide the underlying DRL with prior-guided learning. There are several algorithms that could guide online RL with a single prior policy, such as HuRL [5] and JSRL [38]. Our framework is flexible in choosing those and we select HuRL [5] in our implementation. In the original HuRL, the Q-value estimation in the RL agent is guided by a simple heuristic function that is learned from the Monte-Carlo regression. In our work, we instead leverage the integrated expert function h from above. By dynamically changing a mixing coefficient λ that controls the trade-off between the bias from the expert function h and the complexity of a reshaped MDP, we are able to accelerate the DRL training with a shortened MDP horizon. Specifically, given the state space $\mathcal{S}$, action space $\mathcal{A}$, reward function r that are mentioned at the beginning of Section 3.2, as well as the transition dynamics of the building HVAC system $\mathcal{P}$ and a discount factor γ, we consider an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$. We use the learned integrated expert function h as a prior guess for the optimal value function of $\mathcal{M}$. Thus our online DRL can be described as a reshaped MDP $\tilde{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \tilde{r}, \tilde{\gamma})$, where λ is a mixing coefficient,

$$\tilde{r} = r + (1 - \lambda)\gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)}[h(s')] \tag{11}$$

and

$$\tilde{\gamma} = \lambda\gamma, \tag{12}$$

which is shown at Line 16 in Algorithm 1.

Policy Initialization from Expert Function Distillation: In the above section, we use the integrated expert function h to reshape the reward function and shorten the MDP horizon. In addition, we can also speed up the DRL training through better initialization, by leveraging the expert functions for determining the initial policy (Lines 8 and 9 in Algorithm 1).

Specifically, we initialize the deep Q-network through knowledge distillation [15] on the expert functions. The first step is to extract the knowledge from multiple expert functions (h_u and h_o in our case) to a dataset D_{init}. We set the input dataset as D^x_{init} and the corresponding label set as D^y_{init}. In setting D^x_{init}, we utilize all the unlabeled historical data, which only contain the system state. And the corresponding labels are calculated in a way that is similar to the strategy introduced earlier for integrating expert functions. That is, suppose we have n_h expert functions, then

$$D^y_{init} = \{y | y = (q_1, q_2, \cdots, q_m)\}, \tag{13}$$

$$q_j = \min_i (Q_i(s, f_j)), \tag{14}$$

where $s \in D^x_{init}$, $j \in [1 \cdots m]$, $i \in [1 \cdots n_h]$. As the expert functions we utilize are not as accurate as of the optimal (Q-) value function, we further add two mixing coefficients λ^α_{init}, λ^β_{init} for balancing the relative size of the Q value from different actions. So the new definition of D^y_{init} is

$$\begin{aligned} D^y_{init} = \{y | y = (&\frac{q_1 + (\lambda^\alpha_{init} - 1)\mu_q}{\lambda^\alpha_{init}\lambda^\beta_{init}}, \frac{q_2 + (\lambda^\alpha_{init} - 1)\mu_q}{\lambda^\alpha_{init}\lambda^\beta_{init}}, \\ &\cdots, \frac{q_m + (\lambda^\alpha_{init} - 1)\mu_q}{\lambda^\alpha_{init}\lambda^\beta_{init}})\}, \quad \mu_q = \frac{\sum_{j=1}^m q_j}{m}, \end{aligned} \tag{15}$$

where the definition of $q_j (j \in [1 \cdots m])$ remains the same. Then the next step is to train the deep Q-network of our DRL agent by using the obtained dataset D_{init}. As we consider a regression task, we apply the mean square error as the loss function

$$\mathcal{L}_{init} = \| y - y_{pred} \|^2, y \in D^y_{init}, \tag{16}$$

where y_{pred} is the deep Q-network prediction. We obtain the network weight initialization by training for n_{init} epochs. Moreover, with such policy initialization, we can use a smaller learning rate to tune the deep Q-network in the later DRL stages.

Parameter	Value	Parameter	Value
Expert-model	$[len(s \in \mathcal{S})$, 256,256,256, 256,256,256,2]	Deep Q-network	$[len(s \in \mathcal{S})$, 256, 256, 256, 256, 51]
m	51	Δt	15 mins
γ	0.99	α	1.0
T_{lower} (occupied)	22 ℃	T_{upper} (occupied)	26 ℃
T_{lower} (unoccupied)	12 ℃	T_{upper} (unoccupied)	30 ℃
β	100.0	m_u	7
ξ	1.0	n	5760

Table 1: Hyper-parameters used in our experiments.

4 EXPERIMENTAL RESULTS

4.1 Experiment Settings

We conduct our experiments on a Ubuntu 20.04 OS server equipped with NVIDIA RTX A5000 GPU cards. Docker [29] is utilized for the environment configuration, with Python 3.7.9 and learning framework Pytorch 1.9.0. All neural networks are optimized through the Adam optimizer [19].

We use the building simulation tool in [7] to simulate the behavior of single-zone commercial buildings, with an OpenAI-Gym [3] interface. We model two buildings as defined in the Building Energy Simulation Test validation suite [18]: one is with a lightweight construction (known as Case600FF) and the other is with a heavyweight construction (known as case900FF). Both buildings have the same model settings except that the wall and floor construction have either light or heavy materials. The floor dimensions are $6m$-by-$8m$ and the floor-to-ceiling height is $2.7m$. There are four exterior walls facing the cardinal directions and a flat roof. The walls facing east-west have the short dimension. The south wall contains two windows, each $3m$ wide and $2m$ tall. The use of the building is assumed to be a two-person office with a light load density. The lightweight building is assumed to be located at Riverside, California, USA, and the heavyweight building is assumed to be located at Chicago, Illinois, USA. The weather data for different locations are obtained from the Typical Meteorological Year 3 database [43]. In addition, the various parameters and hyper-parameters mentioned in the previous sections are listed in Table 1.

4.2 Evaluation of Our Framework and Comparison with Standard DDQN

We apply our proposed framework to building HVAC control and demonstrate its effectiveness in accelerating the DRL training, in particular the standard DDQN algorithm. We repeat each experiment 4 times and show the average results.

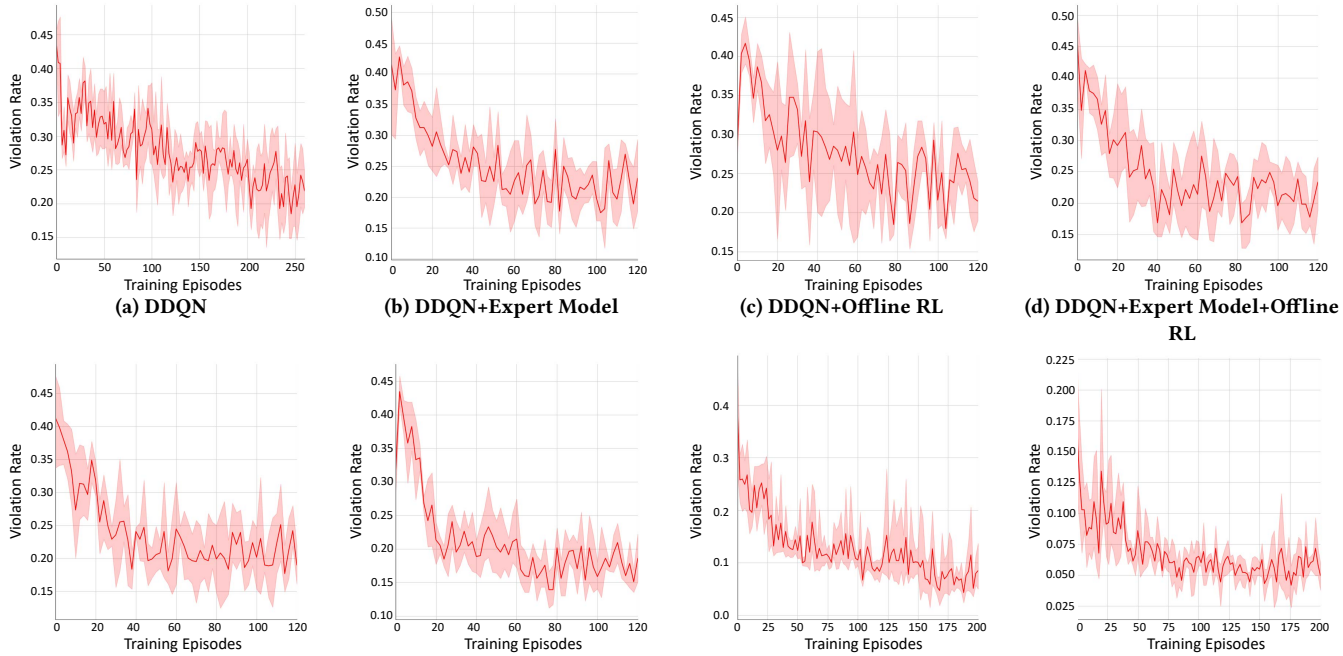
Comparison with Standard DDQN on Training Efficiency: Figure 2 demonstrates the temperature violation rate of the trained controller under different approaches for the lightweight building with weather data from Riverside. Temperature violation rate is one of the main objectives for DRL. It is defined as the percentage of the time the indoor temperature is outside of the comfortable temperature zone, similarly as used in [7, 41, 45, 46].

Method	Number of Episodes
DDQN	212
DDQN+Expert Model	68
DDQN+Offline RL	78
DDQN+Expert Model+Offline RL	40
DDQN+Expert Model+Offline RL +Expert Rules	36
DDQN+Expert Model+Offline RL +Expert Rules+Init	24

Table 2: Number of episodes required to reach the violation rate of 0.2 for the standard DDQN baseline and our approach with various techniques included (the last line being our approach with all techniques in Algorithm 1).

Figure 2a shows the training process of the standard DDQN, and the model needs **about 212 episodes to reach a violation rate at around 20%** for this building from [7] (20% may seem high, but it is due to the limitation of this particular building and its cooling-only HVAC system; more explanation on this later with Figure 4). Figure 2b shows the training process when we add a neural network-based expert model that generates the expert function h_u. About 68 episodes are needed to reach the same violation rate. Figure 2c shows the training process when we add offline RL that generates the expert function h_o, and about 78 episodes are needed to reach the violation rate of 20%. Figure 2d shows the results when we apply both expert functions h_u and h_o, but without the expert rules. We can see that about 40 episodes are needed. Figure 2e shows the results when we integrate the two expert functions h_u and h_o, as well as an expert rule f using the method introduced in Section 3.2. f is defined as follows: when the indoor temperature is below 22℃, the control action is suggested to be set within the set of $\{f_0, f_1, f_2, f_3\}$; if the indoor temperature is above 27℃, the control action is suggested to be set within the set of $\{f_{m-3}, f_{m-2}, f_{m-1}, f_m\}$. We can see that the number of episodes needed is about 36. Finally, Figure 2f shows the training process when we apply all of our proposed techniques, including integrating the expert functions from expert model and offline RL as well as the expert rules, using the integrated expert function to guide DRL training, and conducting policy initialization with the expert functions. We can see that **now only 24 episodes are needed to reach the same violation rate as the standard DDQN, an 8.8X reduction in training time**. Table 2 summarizes the above number of episodes required to reach the violation rate of 0.2 for the standard DDQN baseline and our approach with various techniques included.

For further evaluation, we also conduct experiments on the heavyweight building with weather data from Chicago. In this set of experiments, the major change of the parameters is that the scaling factor β is set to 1.0 in Equation (1). This is because that the average energy consumption of this HVAC system is much higher than that of the previous building, and we need to re-balance the energy cost and the temperature violation in the reward design. Figure 2g and Figure 2h shows the comparison between our approach and the standard DDQN. And the experiments show that the number of episodes needed to reach a violation rate of 5% is reduced from 160 to 80. The improvement, while still significant, is



(a) DDQN (b) DDQN+Expert Model (c) DDQN+Offline RL (d) DDQN+Expert Model+Offline RL

(e) DDQN+Expert Model+Offline RL+Expert Rules (f) DDQN+Expert Model+Offline RL+Expert Rules+Init (g) DDQN (h) DDQN+Expert Model+Offline RL+Expert Rules+Init

Figure 2: Figure 2a to Figure 2f show the comparison between our approach (in different settings with various techniques included) and the standard DDQN method on the lightweight building. The weather data is from Riverside, CA, USA. The x-axis shows the training episodes. The y-axis shows the temperature violation rate. Figure 2a shows the training process under the standard DDQN method. About 212 episodes are needed to reach a violation rate of 0.2. Figure 2b, Figure 2c, Figure 2d, and Figure 2e show the results when we gradually add an expert model that generates expert function h_u, offline RL that generates expert function h_o, an expert rule, and policy initialization based on expert functions, respectively. And we can observe the improvement on the required episodes step by step. Figure 2f shows the training process when we apply all of our techniques. In this case, only 24 episodes are needed to reach the violation rate of 0.2, an $8.8X$ improvement over standard DDQN. Then Figure 2g and Figure 2h show the comparison between our approach with all techniques included (right) and the standard DDQN baseline (left) on the heavyweight building with larger thermal capacity under the weather data from Chicago, IL, USA.

much less than the lightweight building. We suspect that this may be due to the quality of the historical data and plan to investigate it further in future work.

Energy Cost and Other Details: Besides temperature violation rate and the number of episodes for reaching the goal of violation rate below 0.2 (i.e., training efficiency), we also assess the energy cost of the learned controllers during our experiments. We observed that different methods, including the standard DDQN baseline and our approach with various techniques included, achieve very similar energy cost for the learned controllers – in fact within 1% for both the lightweight building and the heavyweight building we tested.

Figure 3 shows the normalized energy cost of our approach with all techniques included for the lightweight building with weather data from Riverside. We can observe that the energy cost quickly decreases to a lower value within 5 to 10 epochs and slightly fluctuates in the later training epochs.

Figure 4 illustrates the building temperature over 2 days, under the controller learned with our approach with all techniques

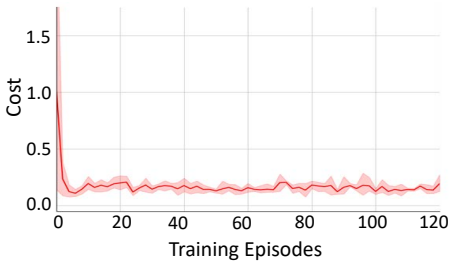


Figure 3: Normalized energy cost during training for our approach with all techniques included for the lightweight building with weather data from Riverside.

included, for the lightweight building with weather data from Riverside. We can see that the temperature violation rate is around 20%. It is relatively high because some violations are very hard to avoid for this particular building. Specifically, the HVAC system is set to only work during the occupied hours (from 7am to 7pm) and

the comfortable temperature range is much more strict during that time (22℃ to 26℃) compared to during the unoccupied time (12℃ to 30℃) [7]. This makes it almost impossible to meet the comfortable temperature range early in the morning since the HVAC system only provides cooling. We can see that after the early morning hours, the temperature is controlled well within the comfortable range by our controller.

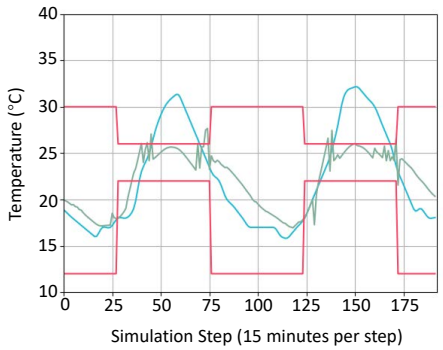


Figure 4: An illustration of the building temperature over 2 days under the controller learned from our approach with all techniques included. The red lines bound the comfortable temperature range. The blue line is the outdoor temperature in Riverside, CA. The green line is the indoor temperature under the learned controller.

4.3 Ablation Studies

Impact of the Historical Data Quantity: We are interested in knowing how the quantity of the historical data may affect the performance of our approach. We conduct a series of experiments that have the quantity of the historical data chosen from $\{5760, 2880, 1440, 720\}$ (i.e., from 2 months of data to 7.5 days of data). The results are shown in Table 3. We can observe that the training becomes faster as the quantity of the historical data becomes larger, as what we would expect.

#Samples	720	1440	2880	5760
#Episodes	116	78	62	24

Table 3: The number of epochs needed by our approach (with all techniques included) for reaching the violation rate of 20% for the lightweight building, under different quantity of the historical data.

Impact of the Historical Data Quality: We also study the performance of our approach under different level of quality for the historical data. Previously we use the historical data collected from an existing controller on the target building. To study different historical data quality, we choose to take random actions with a probability of p. Table 4 shows the results. Our approach performs better with a smaller p, i.e., when our approach learns from historical data based on more reasonable control actions.

p	1.0	0.8	0.4	0.2	0.0
#Episodes	110	104	88	60	24

Table 4: The number of epochs needed by our approach (with all techniques included) for reaching the violation rate of 20% for the lightweight building, under different quality of the historical data.

The Usage of Abstract Phyiscal Model: In addition, we also try to utilize an abstract physical model, i.e., the ARX model from [45], as the expert model to generate h_u, instead of learning a neural network. The training process is shown in Figure 5. About 64 episodes are needed to reach the same violation rate, more than the case where the expert model is a neural network learned from historical data. We think that this is due to the simplicity of the ARX model, and plan to investigate the performance of other abstract physical models in future. Nevertheless, it still provides considerable improvement over the standard DDQN.

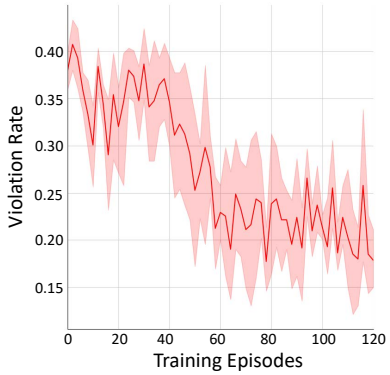


Figure 5: Training result for the lightweight building when the expert model in our approach (with all techniques included) is constructed from an abstract physical model.

5 CONCLUSIONS

In this paper, we present a systematic, unified framework to accelerate online RL for building HVAC control with heterogeneous expert guidances, including abstract physical models, historical data, and expert rules. These guidances are unified through the learning of expert functions, which are then used to accelerate DRL with prior-guided learning and policy initialization. A series of experiments demonstrate that our approach can significantly reduce the training time over previous DRL methods. We believe that our approach not only addresses a critical challenge in applying DRL to building domain, but also has the potential in other domains where existing expertise could be leveraged in improving learning efficiency and performance. We plan to investigate this further in future work.

ACKNOWLEDGMENTS

We gratefully acknowledge the support from Department of Energy (DOE) award DE-EE0009150 and National Science Foundation (NSF) awards 1834701 and 2038853.

REFERENCES

[1] Saber Abrazeh, Saeid-Reza Mohseni, Meisam Jahanshahi Zeitouni, Ahmad Parvaresh, Arman Fathollahi, Meysam Gheisarnejad, and Mohammad-Hassan Khooban. 2022. Virtual Hardware-in-the-Loop FMU Co-Simulation Based Digital Twins for Heating, Ventilation, and Air-Conditioning (HVAC) Systems. *IEEE Transactions on Emerging Topics in Computational Intelligence* (2022).

[2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. 2020. An optimistic perspective on offline reinforcement learning. In *ICML*. PMLR.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[4] Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, and Keith Ross. 2020. BAIL: Best-action imitation learning for batch deep reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 18353–18363.

[5] Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. 2021. Heuristic-guided reinforcement learning. *NeurIPS* (2021).

[6] Drury B Crawley, Linda K Lawrie, Curtis O Pedersen, and Frederick C Winkelmann. 2000. Energy plus: energy simulation program. *ASHRAE journal* 42, 4 (2000), 49–56.

[7] Yangyang Fu, Shichao Xu, Qi Zhu, and Zheng O'Neill. 2021. Containerized framework for building control performance comparisons: model predictive control vs deep reinforcement learning control. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 276–280.

[8] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. 2019. Benchmarking batch deep reinforcement learning algorithms. *arXiv preprint arXiv:1910.01708* (2019).

[9] Scott Fujimoto and Shixiang Shane Gu. 2021. A minimalist approach to offline reinforcement learning. *NeurIPS* (2021).

[10] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*. PMLR, 2052–2062.

[11] Guanyu Gao, Jie Li, and Yonggang Wen. 2019. Energy-efficient thermal comfort control in smart buildings via deep reinforcement learning. *arXiv preprint arXiv:1901.04693* (2019).

[12] Guanyu Gao, Jie Li, and Yonggang Wen. 2020. DeepComfort: Energy-efficient thermal comfort control in buildings via reinforcement learning. *IEEE Internet of Things Journal* 7, 9 (2020), 8472–8484.

[13] Yijie Guo, Shengyu Feng, Nicolas Le Roux, Ed Chi, Honglak Lee, and Minmin Chen. 2020. Batch reinforcement learning through continuation method. In *International Conference on Learning Representations*.

[14] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).

[15] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* 2, 7 (2015).

[16] Bo Huang, Yimin Zhu, Yongbin Gao, Guohui Zeng, Juan Zhang, Jin Liu, and Li Liu. 2021. The analysis of isolation measures for epidemic control of COVID-19. *Applied Intelligence* 51, 5 (2021), 3074–3085.

[17] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. 2019. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456* (2019).

[18] R Judkoff and J Neymark. 1995. International Energy Agency building energy simulation test (BESTEST) and diagnostic method. (2 1995). https://doi.org/10.2172/90674

[19] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[20] Neil E Klepeis, William C Nelson, Wayne R Ott, John P Robinson, Andy M Tsang, Paul Switzer, Joseph V Behar, Stephen C Hern, and William H Engelmann. 2001. The National Human Activity Pattern Survey (NHAPS): a resource for assessing exposure to environmental pollutants. *Journal of Exposure Science & Environmental Epidemiology* 11, 3 (2001), 231–252.

[21] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 1179–1191.

[22] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

[23] Paulo Lissa, Michael Schukat, Marcus Keane, and Enda Barrett. 2021. Transfer learning applied to DRL-Based heat pump control to leverage microgrid energy efficiency. *Smart Energy* 3 (2021), 100044.

[24] C Lombard and EH Mathews. 1992. Efficient, steady state solution of a time variable RC network, for building thermal analysis. *Building and Environment* 27, 3 (1992), 279–287.

[25] Y. Ma, F. Borrelli, B. Hencey, B. Coffey, S. Bengea, and P. Haves. 2012. Model Predictive Control for the Operation of Building Cooling Systems. *IEEE Transactions on Control Systems Technology* 20, 3 (2012), 796–803.

[26] Mehdi Maasoumy, Alessandro Pinto, and Alberto Sangiovanni-Vincentelli. 2011. Model-based hierarchical optimal control design for HVAC systems. In *Dynamic Systems and Control Conference*, Vol. 54754. 271–278.

[27] 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.

[28] Sven Erik Mattsson, Hilding Elmqvist, and Martin Otter. 1998. Physical system modeling with Modelica. *Control Engineering Practice* 6, 4 (1998), 501–510.

[29] Dirk Merkel. 2014. Docker: lightweight linux containers for consistent development and deployment. *Linux journal* 2014, 239 (2014), 2.

[30] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. 2020. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359* (2020).

[31] Aviek Naug, Ibrahim Ahmed, and Gautam Biswas. 2019. Online energy management in commercial buildings using deep reinforcement learning. In *2019 IEEE SMARTCOMP*. IEEE, 249–257.

[32] U.S. Department of Energy. 2011. *Buildings energy data book*.

[33] United States Department of Labor. 2021. OSHA Technical Manual (OTM) Section III: Chapter 2.

[34] Bjarne W Olesen and Gail S Brager. 2004. A better way to predict comfort: The new ASHRAE standard 55-2004. (2004).

[35] Saran Salakij, Na Yu, Samuel Paolucci, and Panos Antsaklis. 2016. Model-Based Predictive Control for building energy management. I: Energy modeling and optimal control. *Energy and Buildings* 133 (2016), 345–358.

[36] Jorren Schepers, Reinout Eyckerman, Furkan Elmaz, Wim Casteels, Steven Latré, and Peter Hellinckx. 2021. Autonomous Building Control Using Offline Reinforcement Learning. In *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. Springer, 246–255.

[37] Mohamed Toub, Chethan R Reddy, Meysam Razmara, Mahdi Shahbakhti, Rush D Robinett III, and Ghassane Aniba. 2019. Model-based predictive control for optimal MicroCSP operation integrated with building HVAC systems. *Energy Conversion and Management* 199 (2019), 111924.

[38] Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, et al. 2022. Jump-Start Reinforcement Learning. *arXiv preprint arXiv:2204.02372* (2022).

[39] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *AAAI 2016*.

[40] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. 2020. Critic regularized regression. *Advances in Neural Information Processing Systems* 33 (2020), 7768–7778.

[41] Tianshu Wei, Yanzhi Wang, and Qi Zhu. 2017. Deep reinforcement learning for building HVAC control. In *54th Annual Design Automation Conference*.

[42] Tianshu Wei, Qi Zhu, and Nanpeng Yu. 2015. Proactive demand participation of smart buildings in smart grid. *IEEE Trans. Comput.* 65, 5 (2015), 1392–1406.

[43] Stephen Wilcox and William Marion. 2008. Users manual for TMY3 data sets. (2008).

[44] Qingqing Xu and Stevan Dubljevic. 2017. Model predictive control of solar thermal system with borehole seasonal storage. *Computers & Chemical Engineering* 101 (2017), 59–72.

[45] Shichao Xu, Yangyang Fu, Yixuan Wang, Zheng O'Neill, and Qi Zhu. 2021. Learning-based framework for sensor fault-tolerant building HVAC control with model-assisted learning. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 1–10.

[46] Shichao Xu, Yixuan Wang, Yanzhi Wang, Zheng O'Neill, and Qi Zhu. 2020. One for many: Transfer learning for building hvac control. In *Proceedings of the 7th ACM international conference on systems for energy-efficient buildings, cities, and transportation*. 230–239.

[47] Liang Yu, Yi Sun, Zhanbo Xu, Chao Shen, Dong Yue, Tao Jiang, and Xiaohong Guan. 2020. Multi-agent deep reinforcement learning for HVAC control in commercial buildings. *IEEE Transactions on Smart Grid* 12, 1 (2020), 407–419.

[48] Liang Yu, Weiwei Xie, Di Xie, Yulong Zou, Dengyin Zhang, Zhixin Sun, Linghua Zhang, Yue Zhang, and Tao Jiang. 2019. Deep reinforcement learning for smart home energy management. *IEEE Internet of Things Journal* 7, 4 (2019), 2751–2762.

[49] Kyungtae Yun, Rogelio Luck, Pedro J Mago, and Heejin Cho. 2012. Building hourly thermal load prediction using an indexed ARX model. In *Energy and Buildings*.

[50] Xiangyu Zhang, Xin Jin, Charles Tripp, David J Biagioni, Peter Graf, and Huaiguang Jiang. 2020. Transferable reinforcement learning for smart homes. In *Proceedings of the 1st International Workshop on Reinforcement Learning for Energy Management in Buildings & Cities*. 43–47.

[51] Zhiang Zhang, Adrian Chong, Yuqi Pan, Chenlu Zhang, and Khee Poh Lam. 2019. Whole building energy model for HVAC optimal control: A practical framework based on deep reinforcement learning. *Energy and Buildings* 199 (2019), 472–490.

[52] Zhiang Zhang, Adrian Chong, Yuqi Pan, Chenlu Zhang, Siliang Lu, and Khee Poh Lam. 2018. A deep reinforcement learning approach to using whole building energy model for hvac optimal control. In *2018 Building Performance Analysis Conference and SimBuild*, Vol. 3. 22–23.