Journal of Building Engineering 90 (2024) 109497

Contents lists available at ScienceDirect

Journal of Building Engineering

journal homepage: www.elsevier.com/locate/jobe

# Reinforcement learning in building controls: A comparative study of algorithms considering model availability and policy representation

Ziyan Wu [a], Wenhao Zhang [a], Rui Tang [a,*], Huilong Wang [b], Ivan Korolija [a]

[a] Institute for Environmental Design and Engineering, The Bartlett, University College London, UK
[b] College of Civil and Transportation Engineering, Shenzhen University, China

ARTICLE INFO

*Keywords:*
Reinforcement learning
Model-free RL
Model-based RL
Building energy efficiency
Control optimization

ABSTRACT

Reinforcement Learning (RL) has presented considerable potential as an advanced control technique in building controls to enable buildings operating more energy-efficient. As various types of RL algorithms have been studied on their performance of building controls, benchmarking these algorithms across the entire spectrum of features is essential to provide an overview and deepen the understanding of RL applications. Therefore, this study aims to compare and analyze the effectiveness of various RL algorithms, encompassing the entire RL categories featured by value-based, policy gradient, actor-critic and model-based RL considering model availability and policy representation. To provide a comprehensive analysis, in addition to the control performance quantified by the cumulative rewards based on the cost function of RL, data demand and robustness of hyperparameter tuning were investigated. The open-source Gym-Eplus framework was selected as the virtual environment to train and test different RL agents. The results showed that both model-free and model-based RL agents outperformed the baseline rule-based control in terms of energy consumption and thermal comfort, and RL agents were capable of evaluating both short-term and long-term rewards to achieve adaptive control optimization continuously along with the online control process. Model-based RL agent improved the data sampling efficiency but presented a relatively sacrificed control performance during the tested summer days.

## 1. Introduction

### 1.1. Research background

Facing the challenges of climate change and energy crisis, countries worldwide have set intensive targets for achieving carbon neutrality. Buildings, contributing to over 36 % of energy consumption and approximately 30 % of greenhouse gas emissions globally, play an important role in realizing the targets of carbon neutrality [1]. In buildings, Heating, Ventilation, and Air Conditioning (HVAC) systems are normally responsible for almost 40 % of total energy consumption, in which operational stage of buildings account for around 30 % [2]. Thus, implementing effective controls to enable HVAC systems in buildings operating efficiently and meanwhile ensuring a comfortable indoor environment is essential [3].

Given the critical role of control strategies in buildings, many studies have been conducted to identify the optimal solutions for building controls. Currently, the predominant control strategy in use is rule-based feedback control (RBC), commonly implemented via

* Corresponding author.
*E-mail address:* rui.tang@ucl.ac.uk (R. Tang).

https://doi.org/10.1016/j.jobe.2024.109497
Received 1 February 2024; Received in revised form 26 April 2024; Accepted 29 April 2024
Available online 30 April 2024
2352-7102/© 2024 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

PID (proportional-integral-derivative) controllers [4]. This RBC strategy normally adheres to a pre-determined schedule set by experienced engineers or established guidelines (e.g., ASHRAE Guideline 36) [5]. The control actions derived from fixed rules lack responsiveness to the variations in operational conditions (e.g., outdoor weather), posing risks of buildings operating seriously deviating from optimal performance. In contrast, model predictive control (MPC) is more advanced than RBC methods. MPC can identify the optimal control actions by using a system model to predict the future evolution of the system performance over the prediction horizon [6]. Despite its advancement, establishing a system model that can precisely capture the dynamic behaviors of the building and its HVAC system has a high demand on domain knowledge. Additionally, a specific model developed is difficult to be generalizable for other buildings, which troubles the widespread adoption of MPC [7].

In recent years, the strengths of artificial intelligence have given rise to reinforcement learning (RL) as a promising technology poised to revolutionize building controls [8]. RL is a subset of machine learning that learns to obtain optimal sequences of actions through interacting with a dynamic environment. The mathematically idealized form of an RL problem is a Markov Decision Process (MDP) [9]. Fig. 1 shows the basic form of MDP represented by a quaternion $MDP = < S, A, R, P >$, where *S, A* and *R* stand for state space, action space, and reward respectively; $r_t$ denotes the reward received by the agent for the action $a_t$ in the state $s_t$ at the time step *t*; *P* means the probability of each action to be selected at the state $s_t$. At each time step, through observing the environment states, the agent adjusts its policy to output the actions that can maximize the cumulative rewards, that is, the sum of rewards an agent accumulates over a sequence of actions within an environment. The environment embodies the problem to be solved given that MDP is satisfied.

RL methods can be categorized into model-based and model-free approaches, depending on whether an environmental model is utilized. Model-based methods employ a model to simulate and understand the dynamics of the environment, while model-free reinforcement learning (MFRL) algorithms directly identify the optimal policy (i.e., control strategy) through the interactions with the environment, which owns the nature of the trial-and-error process [10]. MFRL algorithms mainly rely on three methods to realize policy representation, i.e., value-based, policy gradient (PG), and actor-critic (AC) [11].

### 1.2. Literature review

The literature review was conducted via *ScienceDirect* and *Google Scholar* using the searching keywords of 'Reinforcement Learning', 'HVAC' and 'Control' with targeting the past five years (exactly from January 2019 to March 2024). Rather than exhaustively reviewing all the publications within the period, the objectives of the literature are: (1) to introduce the current status of typical RL algorithms under different categories (i.e., value-based, policy gradient, actor-critic and model-based RL) applied in building controls; (2) to summarize the comparative studies of benchmarking different RL algorithms in building controls.

Value-based MFRL algorithms estimate the expected return of states or state-action pairs to derive the optimal policy. This method is stable and efficient, particularly in environments with discrete and finite action spaces. The versatility of value estimation and policy extraction methods results in a wide array of variations in value-based algorithms, ranging from Q-table algorithms to state-of-the-art Rainbow Deep Q-Network [12]. Qiu et al. [13] applied Q-learning to control a cooling water system and the results showed that the energy savings achieved by RL reached 12 %, surpassing the 7 % achieved by a RBC. Valladares et al. [14] utilized the deep Q-network (DQN) to control temperature setpoints of air-conditioning units. Compared to the simple RBC, the RL agent presented a better Predicted Mean Vote (PMV) and a 10 % lower $CO_2$ level with consuming 4–5% less energy. Yoon and Moon [15] proposed double deep Q-network (DDQN) integrated with an experience replay buffer to control a variable refrigerant flow (VRF) system and a humidifier, achieving a 32.2 % reduction of energy consumption without compromising the PMV.

When dealing with continuous or high-dimensional action spaces, PG algorithms can directly optimize the policy without the need to discretize the action space or approximate a value function encountered in value-based methods. Jia et al. [16] proposed REINFORCE with expert policy guidance to control the supply air temperature of a variable air volume (VAV) system, achieving a superior control performance over the control logic developed in ASHRAE Guideline 36. Azuatalam et al. [17] proposed Proximal Policy Optimization (PPO) and Trust Region Policy Optimization (TRPO) to control the heating/cooling setpoint of a VAV system in demand response events. The results showed that a maximum 22 % reduction of energy consumption was achieved by the RL agent against the RBC controller.

Actor-critic algorithms blend value-based approaches (critic) to estimate a value function with policy-based methods (actor) to directly learn a policy, harnessing the strengths of both to enhance the convergence properties and learning stability, especially in continuous action spaces. Zhang et al. [18] developed an Asynchronous Advantage Actor-Critic (A3C) to optimize the operation of a

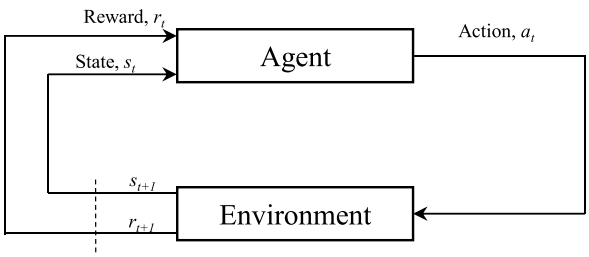



**Fig. 1.** Schematic of Markov Decision Process (MDP).

radiant heating system in an office building. The results showed that the RL agent achieved a 16.7 % heating demand reduction compared to the rule-based control. Schreiber et al. [19] proposed Deep Deterministic Policy Gradient (DDPG) to control the operation of a chiller system, resulting in a 14 % reduction of weekly billing costs. Zhang et al. [20] deployed A3C in a VRF system to achieve 16.1 % energy saving with better thermal comfort and demonstrate its robustness under varied weather conditions.

The nature of trial-and-error learning process in MFRL makes the training time-consuming and data-demanding. Also, the random control actions may pose risks to system security and robustness [21]. Compared with MPC that requires training data from several days to weeks, MFRL often spans from several months to years of training data [8]. To solve this problem, model-based reinforcement learning (MBRL) algorithms are emerging due to their higher data sampling efficiency over MFRL while maintaining asymptotic performance [22]. MBRL algorithms focus on addressing two key issues: first, how to construct an accurate environment model during learning, and second, how to utilize the learned environment model for policy optimization. Based on these two considerations, MBRL can be broadly categorized into two types, one of which is known as the "learned model". The Dyna framework falls into this category. It combines MBRL with MFRL to utilize the environment model to generate abundant virtual samples, which are then used by model-free algorithms for further policy optimization. The other type is known as the "given model". One example is AlphaZero, originating from the agent for the chessboard game AlphaGo [23]. It utilizes existing models through Expert Iteration and Monte Carlo Tree Search (MCTS). In addition, studies recently begun to fit the MPC into the MDP framework for solving optimization problems. This contributes to reducing the amount of data required for training and enhances the stability of the learning process. Gao and Wang [21] developed model-based DDQN and model-based Soft Actor-Critic (SAC) with Dyna framework to control a heat pump. The results showed that model-based RL achieved a control performance as good as model-free RL with a shorter training time. Chen et al. [24] proposed PPO integrated with policy learned by Differentiable MPC to control supply water temperature of a heat exchanger. After imitating, the agent achieved 6.6 % energy saving compared to the published result using the same environment. Zhang et al. [25] developed a model-based RL combining MPC and random-sampling shooting that learned the system dynamics using a neural network. Compared with PPO, the model-based approach improved the sample efficiency by 10 times with significantly reducing the energy consumption.

As various types of RL algorithms from different categories have been studied on their performance of building controls, benchmarking the training efficiency and control performance of these algorithms across the entire spectrum of features is essential to provide an overview and deepen the understanding of applying RL in building controls and revealing specific behavioral patterns of each algorithm. The studies targeting the comparison of different RL algorithms conducted in recent years are summarized in Table 1. Schreiber et al. [19] and Du et al. [26] compared the performance between DDPG and DQN and results showed that DDPG had a slight advantage over DQN, demonstrating the better capability of actor-critic architecture in handling high-dimensional state and action spaces against value-based algorithms. Gao and Wang [21] compared SAC and DDQN as well as Dyna-based MBRL and concluded that MBRL performed similarly as MFRL but with a relatively shorter training time. Chen et al. [22] and Zhang et al. [25] compared the MBRL (combining MPC approach) with PPO and the results showed that the MBRL algorithm could approach the performance of MFRL but with an enhanced sample efficiency. Ding et al. [27] compared the proposed MBRL approach with MBRL approach developed in Ref. [25] and PPO to show the method can reduce the training data set by 10.52 times while achieving approximate performance to PPO and obtain 8.23 % more energy savings compared to MBRL approach in Ref. [25]. These studies used PPO as a baseline algorithm of MFRL to illustrate the improvements in sample efficiency brough by their proposed MBRL methods.

### 1.3. Aim and objectives

According to the literature, benchmarking different RL algorithms is rarely conducted targeting the entire spectrum of RL categorization to reveal and characterize the features of diverse techniques of model availability and policy representation. This study, therefore, aims to benchmark and analyze the effectiveness of RL algorithms selected from each category of RL algorithms, consisting of Deep Q-Network (DQN), Double Deep Q-Network (DDQN), Proximal Policy Optimization (PPO), Asynchronous Advantage Actor-Critic (A3C) and model-based Gnu-RL. The training process, hyperparameter tuning, data demand and control performance in buildings are analyzed and compared. The main contributions of this work include: (1) the entire spectrum of RL algorithms featured by model availability and methods of policy extraction is compared in building controls; (2) the training efficiency and control accuracy are considered across the RL algorithms under different categories; (3) the robustness of hyperparameter tuning for different algorithms is analyzed and the impacts of target network under value-based algorithms, dominant algorithms in building controls based on the literature review, are investigated.

**Table 1**
Summary of recent comparative studies of RL algorithms for HVAC controls.

| Reference | Year of publishing | Different Ways of Policy Representation | | | Model Availability |
|---|---|---|---|---|---|
| | | Value-based | Policy gradient | Actor-critic | |
| [19] | 2020 | DQN | | DDPG | |
| [21] | 2023 | DDQN | | SAC | √ |
| [22] | 2022 | | PPO | | √ |
| [25] | 2019 | | PPO | | √ |
| [26] | 2021 | DQN | | DDPG | |
| [27] | 2023 | | PPO | | √ |

## 2. Reinforcement learning algorithms to be benchmarked

According to the literature introduced above, reinforcement learning is normally categorized via the features of model availability, i.e., whether a model of the environment is deployed in the decision-making process, and policy representation, i.e., how to estimate the actions and optimize the policy. Reinforcement learning algorithms are therefore divided into model-based and model-free algorithms, and within model-free algorithms, three categories are further realized: value-based, policy-gradient, and actor-critic methods. The selection of algorithms to be benchmarked are based on the above literature, considering the following criteria: (1) whether the algorithm is representative for its category; (2) whether the algorithm has already demonstrated effectiveness in HVAC control; (3) the inadequacy of comparisons with other categories; (4) and their potential for widespread application in the future. In this study, four typical algorithms are selected under the above four categories/subcategories. Value-based algorithms are relatively further developed and therefore two algorithms of Deep Q-Network (DQN) and Double Deep Q-Network (DDQN), covering the main techniques of improving the algorithm capabilities, are selected to explore their control performance in buildings. For Policy Gradient (PG) algorithms, Proximal Policy Optimization (PPO) is selected due to its promising strength of avoiding performance variance and unstable updates. Under the actor-critic (AC) algorithms, the Asynchronous Advantage Actor-Critic (A3C) algorithm is selected because of its faster convergence rate that are benefited from its unique characteristic of parallelism, utilizing multiple parallel agents to learn the environment simultaneously. Among model-based reinforcement learning (MBRL), integrating model predictive control (MPC) to help the updating of reinforcement learning policy has been demonstrated its effectiveness to realize building control with reducing the training process and Gnu-RL [24] is selected in this study to enable the elements of rule-based, MPC and RL involved in the study. The detailed principles of each selected algorithm are presented below.

### 2.1. Value-based algorithms

The DQN and DDQN are selected as typical algorithms in the category of value-based RL. Value-based RL algorithms use the temporal-difference (TD) method to estimate and update the action-value function. DQN algorithm is developed from Q-learning by substituting discrete Q-table with deep neural networks to avoid the size of Q-table growing exponentially when dealing with continuous or high-dimensional states [28]. DDQN is an extended version of DQN designed to address the issue of potential overestimation that arises in Q-learning when evaluating action values [29]. DQN and DDQN algorithms encompass the main techniques of experience replay and target networks to advance the performance in value-based RL algorithms [30]. The operational mechanism of DQN and DDQN are illustrated in Fig. 2.

In Fig. 2, the replay memory module stores previous state transitions $(s, a, r, s^{'})$ at different timesteps in a replay buffer and randomly samples a mini batch during training, which significantly improves data efficiency by considering past experiences. Also, this

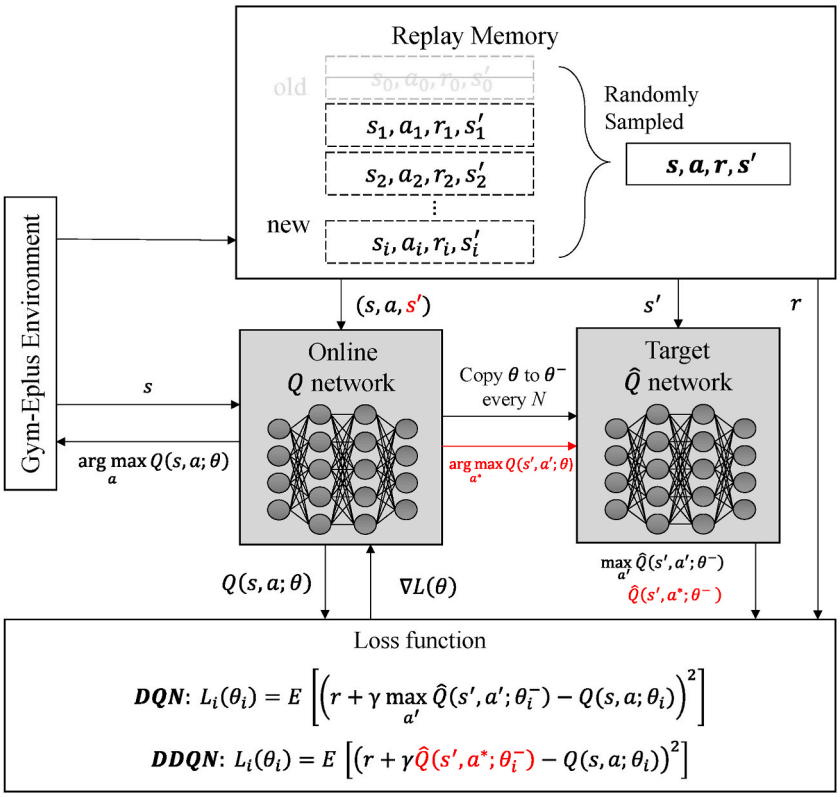



**Fig. 2.** The mechanisms of DQN and DDQN algorithms.

function breaks the temporal correlation between interactions with the environment by sampling experiences from a range of states and actions, reducing the risk of overfitting to recent experiences. In this study, the reuse time of samples in the replay memory was set as a low value (i.e.,2) to avoid the training over relying on the specific samples. The target $\widehat{Q}$ network, periodically updating the copy of online $Q$ network, aids in computing stable target Q-values to mitigate oscillations and divergence in the learning process. The deferred updates of the target network ensure it robust to the noise in the current Q-value estimates, yielding more reliable value approximation. For DQN, target network weighing is responsible for obtaining the TD target $\max_{a} \widehat{Q}(s', a'; \theta_i^-)$ from all possible actions $a'$ in the next state $s'$. For DDQN, the online $Q$ network weighing $\theta$ is deployed to select the action $a^*$ to be executed in the next state $s'$, and then target $\widehat{Q}$ network weighing $\theta_i^-$ is applied to evaluate the value of the chosen action $a^*$, i.e., $\widehat{Q}(s', a^*; \theta_i^-)$ in loss function of DDQN. The Q-value update rule in DQN and DDQN is shown in the loss function with different TD targets. The annotation in red in Fig. 2 shows the differences between DQN and DDQN algorithms. If the red annotations are removed, the framework then represents the DQN algorithm with replay memory and target network.

### 2.2. Policy gradient algorithm

PPO, as a policy gradient method, directly optimizes the policy to ensure reward improvement and training stability [31]. The PPO is further developed based on the basic policy gradient theorem, of which calculation is presented in Eq. (1) and Eq. (2):

$$J(\theta) = \sum_{S} d_{\pi_\theta}(s) V_{\pi_\theta}(s) = \sum_{S} d_{\pi_\theta}(s) \sum_{a} \pi_\theta(a|s) Q_\pi(s,a) \tag{1}$$

$$\nabla_\vartheta J(\theta) = E_{\pi_\theta}[\nabla_\vartheta \, ln \, \pi_\theta(a|s) Q_\pi(s,a)] \tag{2}$$

Eq. (1) decomposes the total expected return under the policy $\pi_\theta(a|s)$ into a weighted sum of the expected returns $Q_\pi(s,a)$ for each pair of state and action, where the weights are given by the stationary distribution $d_{\pi_\theta}(s)$ of the Markov chain for policy $\pi_\theta$. Eq. (2) describes how to adjust the policy parameter $\theta$ along the direction of gradient ascent $\nabla_\vartheta J(\theta)$ to maximize the total expected return, where $\nabla_\vartheta \, ln \, \pi_\theta(a|s)$ indicates the change in the probability of choosing the action $a$ in the state $s$ as $\vartheta$ changes. PPO aims to stabilizes the training by avoiding excessive updates of parameters $\theta$, which are susceptible to overly altering the policy in a single update. Therefore, PPO further optimizes the objective function by introducing a clipping mechanism [31]:

$$L(\theta) = \widehat{E}_t \left[ min\left( w_t(\theta) A_t^{GAE(\gamma,\lambda)}, \text{clip}(w_t(\theta), 1-\epsilon, 1+\epsilon) A_t^{GAE(\gamma,\lambda)} \right) \right] \tag{3}$$

Where, the clipped objective function chooses the smaller value between the original objective $w_t(\theta) A_t^{GAE(\gamma,\lambda)}$ and the clipped objective $\text{clip}(w_t(\theta), 1-\epsilon, 1+\epsilon) A_t^{GAE(\gamma,\lambda)}$. The clipped objective limits the ratio $w_t(\theta)$ of current policy and old policy into $1-\epsilon$ and $1+\epsilon$, $\epsilon$ is a hyperparameter, as shown in Eq. (4):

$$w_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{4}$$

The advantage function $A_t^{GAE(\gamma,\lambda)}$ quantifies the performance of the chosen action $a$ over the mean for a given state $s$, as shown in Eq. (5):

$$A(s,a) = Q(s,a) - V(s) \tag{5}$$

where, $Q(s,a)$ is the action-value function (expected return of taking action $a$ in state $s$); $V(s)$ is the state-value function (expected return of state $s$ under the current policy). In this paper, Generalized Advantage Estimation (GAE) method is used to estimate the advantage function. GAE introduces a trade-off parameter $\lambda$ to apply a weighted average that considers rewards over multiple steps $k$, thus smooth high variance in the estimation [32]:

$$A_t^{GAE(\gamma,\lambda)} = \sum_{k=0}^{\infty} (\gamma\lambda)^k \delta_{t+k}^V \tag{6}$$

where, $\gamma$ is the discount factor used to calculate the present value of future rewards. $\lambda$ is GAE parameter. TD residual $\delta_{t+k}^V$ utilizes immediate reward $r_t$ and value estimation of future states $V(s_{t+1})$ to update the value of current state $V(s_t)$, as shown in Eq. (7):

$$\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{7}$$

### 2.3. Actor-critic algorithm

The A3C algorithm [9], as an actor-critic method, operates on the principles of Policy Gradient and TD learning. It directly optimizes the policy (the "actor") through policy gradient and estimates the value function (the "critic") via TD learning. During the policy update process, A3C algorithm uses gradient ascent to update the policy parameters $\theta$ to move it in the direction of higher expected future rewards, as demonstrated in Eq. (8).

$$\theta \leftarrow \theta + \alpha \nabla log \, \pi(s,a;\theta) A(s,a) \tag{8}$$

where, $\theta$ is policy parameters; α is the learning rate; $\pi(s,a;\theta)$ is the policy; $\nabla log \, \pi(s,a;\theta)$ is the policy gradient; $A(s,a)$ is the advantage

function of action $a$ at the state $s$. In the policy gradient, the policy is the 'actor' which is used to select actions, and the advantage function is the 'critic' which is used to evaluate the learned control policy. The advantage function in A3C algorithm shares the same definition as Eq. (5). The value function $V(s)$ is updated based on the TD target and the TD error $\delta$, which are defined as Eq. (9) and Eq. (10) respectively [26].

$$V(s) \leftarrow V(s) + \alpha\delta \tag{9}$$

$$\delta = r + \gamma V(s') - V(s) \tag{10}$$

where, $r$ is the reward received after taking action $a$ in the state $s$; $\gamma$ is the discount factor; $s$ is the new state after taking action $a$; $V(s')$ is the estimated value of the new state. The agents update their local models independently and asynchronously, and these updates are occasionally synchronized with the global model, thereby enabling efficient exploration and learning, as shown in Fig. 3 [18].

### 2.4. Model-based algorithm

The Gnu-RL is a model-based RL algorithm that combines imitation learning, RL, and MPC [24]. Utilizing imitation learning enables the application of RL for building control without the need for priori information other than historical data. Moreover, since the Gnu-RL employs the differentiable MPC policy instead of neural networks in RL, it can encode domain knowledge in terms of system dynamics in a way that makes it both sample-efficient and interpretable. Initially, the Gnu-RL agent is pre-trained using the historical state-action pairs of the existing controller, aiming to match its behavior before interacting with the environment. Subsequently, the agent employs the PPO algorithm during testing to refine the building dynamics model and further improve its policy. The Gnu-RL has been tested on its potential in buildings in Ref. [24].

The whole framework of Gnu-RL can be divided into two parts, imitation learning through differentiable MPC and online learning through PPO. In the Gnu-RL, the building dynamics applied in the MPC is described as a linear model shown in Eq. (11):

$$x_{t+1} = Bx_t + B_a a_t + B_d d_t \tag{11}$$

where, $x_t$ is the zone temperature at the step $t$; $a_t$ is the action at the step $t$; $d_t$ is the disturbance at the step $t$ (such as weather and internal heat gain); $B, B_a$ and $B_d$ are the coefficient metrics characterizing the thermodynamics of the building, and are the only parameters needed to learn for the differentiable MPC policy., i.e. $\theta = \{B, B_a, B_d\}$. The mean squared error (MSE) loss between the trajectories of the existing controller and MPC controller is minimized through the loss function shown in Eq. (12):

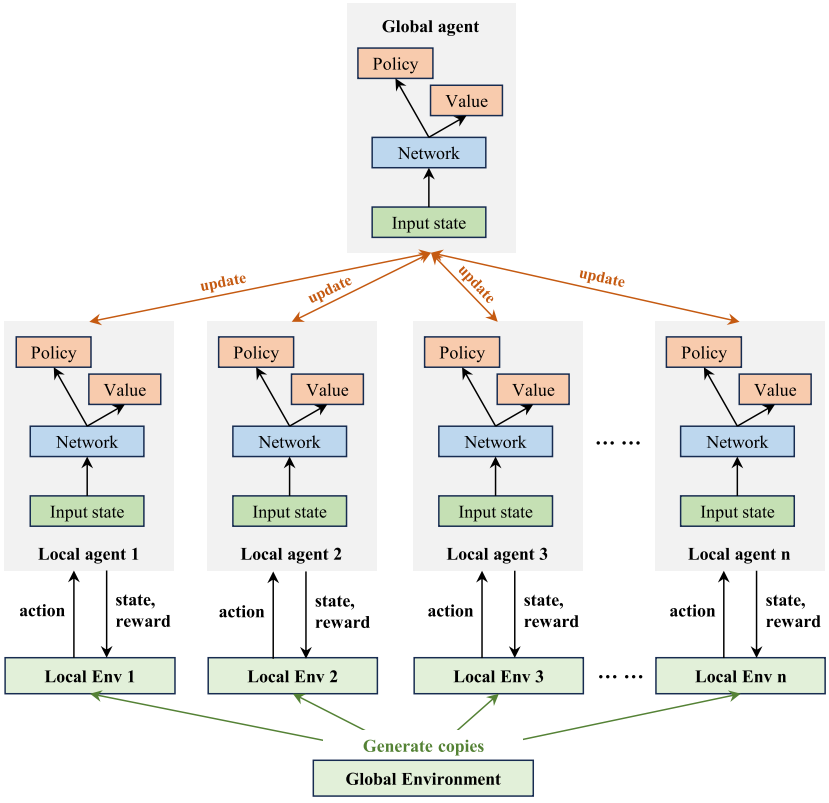



**Fig. 3.** Framework of A3C algorithm.

$$L_{imit}(\theta) = \sum_t \left(\lambda_t \left(x_t - x_t^*\right)^2 + \left(u_t - u_t^*\right)^2\right) \tag{12}$$

where, $x_t, u_t$ are the state and action for the existing controller, and $x_t^*, u_t^*$ are those for the MPC controller. Then the policy can be updated by Eq. (13):

$$\theta \leftarrow \theta - \alpha \nabla_\theta L_{imit}(\theta) \tag{13}$$

where, $\alpha$ is the learning rate. Once a pre-trained policy $\pi_{\tilde{\theta}}$ is obtained, the policy is updated online by PPO algorithm, which can be simplified as Eq. (14):

$$\theta \leftarrow \theta - \alpha \nabla_\theta L_{PPO}(\theta) \tag{14}$$

where, $L_{PPO}(\theta)$ is the same as defined in Eq (3).

## 3. Test platform

### 3.1. Framework of benchmarking RL algorithms in building controls

The flowchart of benchmarking different RL algorithms in building controls is shown in Fig. 4. The reference model of '5ZoneAutoDXVAV.idf' in the EnergyPlus was chosen as the target building in this paper. This building is a small office building located in Pittsburgh, which has one storey with 5 zones. The building has windows at each elevation and glass doors at south and north elevations. This building model was selected because it was well developed as a representative office building and had been used in many research works, including the studies on RL algorithms due to its proper level of complexity regarding the dimensions of states and actions spaces for RL tasks [ [17,33–35]]. The Heating, Ventilation, and Air Conditioning (HVAC) system in the building is a packaged Variable Air Volume (VAV) system, operating from 8:00 a.m. to 18:00 p.m. The room temperature was set as 24°$C$ during occupied periods. The period from 1st June to 31st August was used to train the RL agent with time interval of 15 min. The weather file used for training was pittsburgh_TMY2. epw.

In real-world applications of RL for building controls, uncertainties such as building dynamics and sensor inaccuracies could impact the performance of RL agents. These uncertainties may lead to unexpected environmental transitions, potentially degrading the performance of RL algorithms. The benchmarking of this study was from algorithmic perspective and these uncertain factors were not considered in detail. To evaluate the performance of trained RL agents in new environments with a certain level of uncertainties, in this study, a validation model was generated by changing the weather file to pittsburgh_TMY3. epw. The other settings for testing remained consistent with the training. The objective of the RL agent was to minimize the operational energy consumption of the HVAC system while maintaining the indoor temperature within a pre-set comfortable range. The Gym-Eplus was used for the implementation of RL in building controls [36]. Gym-Eplus is an open-source virtual environment for RL application based on the EnergyPlus software wrapped within OpenAI Gym [37]. It interfaces with the EnergyPlus model using the inter-program communication function available in EnergyPlus, in conjunction with the BCVTB (Building Controls Virtual Test Bed) middleware [38].

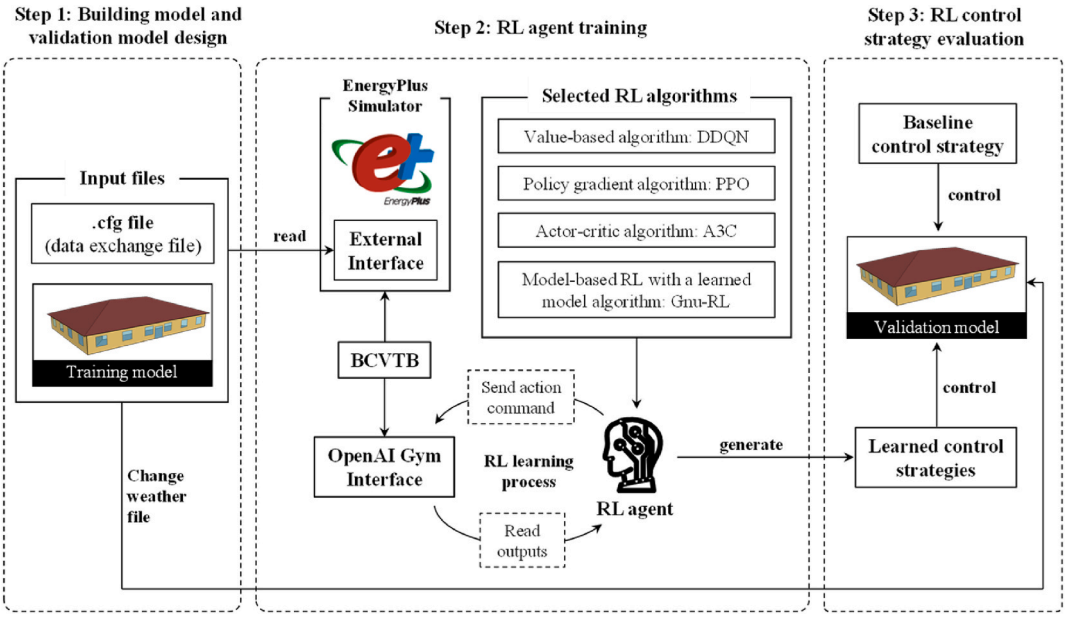



**Fig. 4.** Flowchart of benchmarking different RL algorithms.

### 3.2. Design of RL controls in buildings

#### 3.2.1. RL states

In the RL agent, the states reflect what is observed in the environment, which decides the action subsequently. In this study, the chosen states are shown in Table 2. The states were normalized in the range of [0, 1] to speed up the training process. This could reduce the scale differences among state features, preventing some features from dominating the learning process.

#### 3.2.2. RL actions

Actions are the targeting control variables that the RL agent comes to choose through the policy. In this study, the control variable was the set-point of supply air temperature to the zones, and the considered range is set to be [12,24]. The action space was discretized into uniform intervals with a gap of 0.5 $^{\circ}C$, i.e., [12, 12.5, 13.0, …, 24.0] to ensure sufficient control accuracy for the implementation of Deep Q-Network (DQN) and Double Deep Q-Network (DDQN) algorithms.

#### 3.2.3. RL reward

The reward defines the control objective of RL. In this study, the RL agent was to enhance the energy efficiency while ensuring acceptable indoor thermal comfort. The PPD (Predicted Percentage of Dissatisfied) representing the indoor thermal comfort and the total HVAC power demand were used to design the reward function. The total HVAC power demand was normalized to [0, 1] to match the scale of PPD in the objective function. The reward function of RL agent is shown as Eq. (15):

$$Reward = (1 - ob_{PPD}) \times OccupantFlag \times \rho + \left(1 - ob_{HVAC\ Energy}\right) \times \varepsilon \tag{15}$$

where, $ob_{PPD}$ is the RL agent observation of PPD; *Occupant Flag* is the occupancy status of the building and its value is 1 for occupied time and 0 for unoccupied; $ob_{HVAC\ Energy}$ is the normalized total HVAC power demand; $\varepsilon$ and $\rho$ are the weighting factors and set as 0.5 and 0.5 respectively in this study. Although the action of switching off (i.e., $ob_{HVAC\ Energy} = 0$) HVAC system when the building is not in use (i.e., *Occupant Flag* $= 0$) yields the maximum immediate reward, the RL agent is capable of learning to pre-cool the building to get a higher cumulative reward in the future.

#### 3.2.4. Hyperparameter tuning

The performance of RL relies on the hyperparameter settings in a given task and environment [39]. In general, each algorithm has an optimal set of hyperparameter settings that may differ from the others. The tuning of hyperparameters usually requires multiple trials and iterations to find the combination of hyperparameters that maximizes the performance of the algorithm. In RL algorithms, learning rate is crucial hyperparameter because it determines the extent that newly acquired information overrides old information. A well-tuned learning rate guides the convergence rate of the algorithm to optimal or near-optimal policies. Therefore, in this study, the learning rate were optimized for each algorithm and other hyperparameters in different algorithms were set based on their algorithm characteristics. The learning rate was optimized from 10 values: [0.000001, 0.000005, 0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05], which covered a sufficient range to find an optimal learning rate. The learning rate that enabled the algorithm to achieve maximum cumulative reward was selected, as shown in Table 3. The combination of best learning rate and other hyperparameters for each algorithm is shown in Table 4. The structure of neural network deployed in this study was kept simple because a complex neural network model was not necessarily leading to a better control performance but may trouble the performance of RL due to the risks of overparameterized and difficulties in the update of trained neural network. The hyperparameter tuning is designed as Fig. 5, considering the number of layers and the number of neurons under each test of learning rate. In total, 300 (i.e., 5 algorithms*10 learning rates*6 structures of neural network) tests were conducted for all algorithms. In Fig. 5, 'ReLu x-y' means a neural network equipped with y layers with x neurons and applied ReLu as activation function of each neuron. The hyperparameter tuning for each algorithm was conducted. The ReLu 128–2 was selected after the tests.

### 3.3. Baseline control

The baseline control is to use the cooling load of the hottest zone in the building to determine the supply air temperature ($T_{set}$), and its control logic is shown in Eq. (16):

$$T_{set} = T_i + \frac{Q_i}{C_{p,air} \times m_{i,max}} \tag{16}$$

where, $T_{set}$ is the supply air temperature of the VAV system, $i$ is the zone number; $T$ is the indoor air temperature, $Q$ is the zone cooling

**Table 2**
Selected states for the RL agent.

| No. | State | No. | State |
|---|---|---|---|
| 1 | Site Outdoor Air Drybulb Temperature, °C | 7 | Building Mean Temperature, °C |
| 2 | Site Outdoor Air Relative Humidity, % | 8 | Indoor Air Temperature Setpoint, °C |
| 3 | Site Wind Speed, m/s | 9 | Building Mean PPD, % |
| 4 | Site Wind Direction, ° | 10 | Occupancy Flag i.e., 0 or 1 |
| 5 | Site Diffuse Solar Radiation Rate per Area, W/m$^2$ | 11 | Cooling Coil Electric Power, W |
| 6 | Site Direct Solar Radiation Rate per Area, W/m$^2$ | 12 | Facility Total HVAC Electric Demand Power, W |

**Table 3**
Optimized learning rate of each algorithm.

| Algorithm | Learning Rate | Maximum Cumulative Reward |
|---|---|---|
| DQN | $5 \times 10^{-3}$ | 5878.99 |
| DDQN | $5 \times 10^{-3}$ | 5886.41 |
| PPO | $5 \times 10^{-4}$ | 5889.37 |
| A3C | $1 \times 10^{-6}$ | 5892.25 |
| Gnu-RL | $5 \times 10^{-4}$ | 5819.62 |

**Table 4**
Hyperparameter settings of each selected algorithm.

| Item | DQN & DDQN | PPO | A3C | Gnu-RL |
|---|---|---|---|---|
| Network Layers | 2 | 2 | 2 | / |
| Neurons Number (L1/L2) | 128/128 | 128/128 | 128/128 | / |
| Batch Size | 512 | 512 | / | 512 |
| Discount Factor | 0.99 | 0.99 | 0.99 | 0.99 |
| Clipping Ratio | / | 0.18 | / | 0.18 |
| Generalized Advantage Estimation Coefficient | / | 0.98 | / | / |
| Local Agent Number | / | / | 12 | / |
| Learning Batch Size | / | / | 5 | / |
| Planning Horizon | / | / | / | 12 |
| Update Episode | / | / | / | 1 |

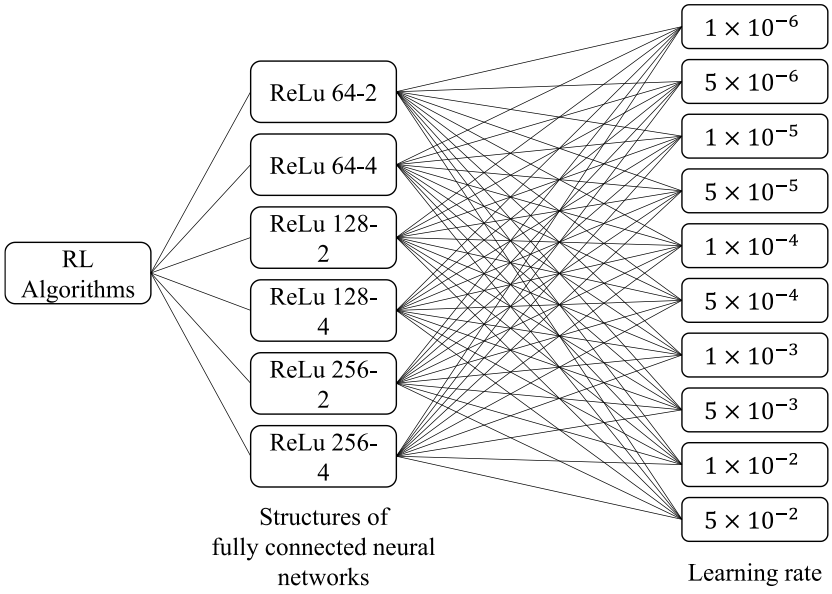



**Fig. 5.** Hyperparameter tuning of neural network and learning rate.

load, $C_{p,air}$ is the specific heat of air, and $m_{max}$ is zone maximum supply air mass flow rate.

## 4. Results and discussion

### 4.1. Convergence of training performance

In this study, all algorithms were executed on the same computer within Oracle VM VirtualBox to ensure a fair comparison [40]. The virtual machine was configured with an Ubuntu 16.04 operating system, allocated 20,000 MB of RAM, and 8 CPU processors, with the execution cap set at 90 %.

Fig. 6 presents the convergence curves of selected algorithms during the training and achieved cumulative rewards during the testing with best set of hyperparameters. The dotted lines parallel to the x-axis indicate the total rewards obtained by the agent in a testing episode. The training curves were clearly upward in the cumulative rewards along with the training process. After a few episodes, the Reinforcement Learning (RL) agents outperformed the baseline controller through the interaction with the environment. Since the Gnu-RL was designed for the agent not to be trained online after the imitation learning was completed, it only had the testing

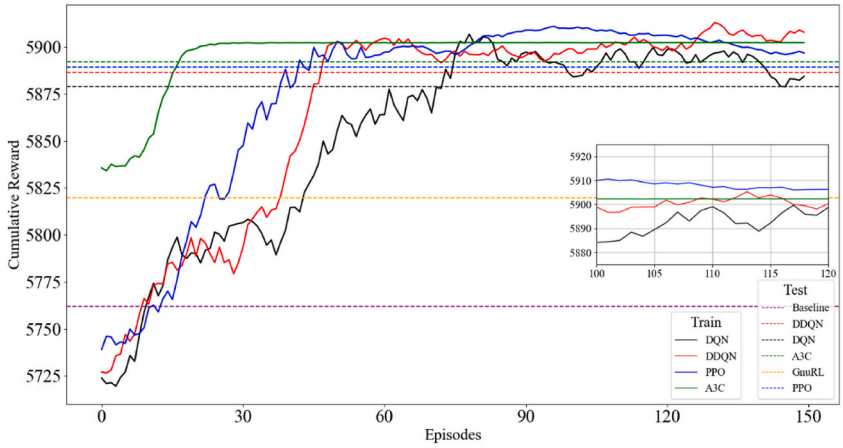



**Fig. 6.** Cumulative rewards of RL algorithms in the training and testing processes.

progress, of which cumulative reward was slightly lower than other model-free algorithms. As shown in the figure, the training curves of all algorithms were marginally higher than their testing curves, indicating the changes in the environment reduced the capability of the agents to maintain the same performance in the training environment, particularly under new transition probability or combinations of states that the agent had never encountered before [9].

The training curve of Asynchronous Advantage Actor-Critic (A3C) quite differed from the others, showing a good stability after convergence. This may be because the learning rate that yielded the best performance for A3C in this case was tuned to be $1 \times 10^{-5}$, which was very lower against the values of the other algorithms. The A3C outperformed the others at the start of training process, which was benefited by its characteristic of asynchronous that multiple sub-agents (i.e., set as 12) learned in parallel and each asynchronously updated the global parameters using temporal difference learning. As shown in Table 4, the learning batch size for a single agent is set as 5, which implies that updates to the global gradient are frequent and incremental. In contrast, for other algorithms, the amount of data from a single episode does not lead to such frequent updates, thereby preventing them from exploring effective strategies as early as A3C does. In addition, sub-agents explored different replica of the environment to learn different strategies, which often leads to the discovery of more rewards especially in the initial stages [41].

The training of Deep Q-Network (DQN) was converged at around 80 episodes, slower than Double Deep Q-Network (DDQN) at around 50 episodes. During the training, both algorithms exhibited oscillations but the oscillations in DDQN ended earlier than DQN, which could be attributed to that DDQN reduced the overestimation of action values by employing two separate networks to decouple the two steps of selecting the best action and evaluating its value. PPO (Proximal Policy Optimization) algorithm experienced minimal oscillation in the training. This indicates that the clipping mechanism avoids the pitfalls of large policy updates that could lead to performance degradation. In addition, the Generalized Advantage Estimation (GAE) method adopted in this study provides smoother and more stable estimates of the advantage function, leading to more efficient policy updates in most circumstances.

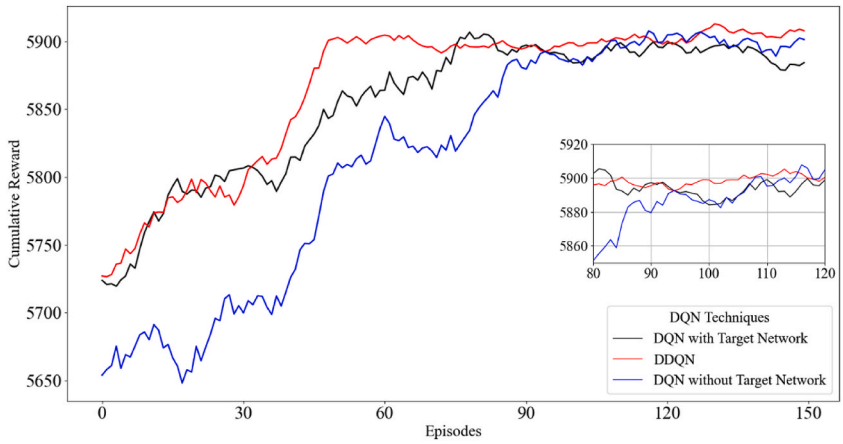



**Fig. 7.** Effectiveness of target network applied in value-based algorithms.

### 4.2. Analysis of training performance

#### 4.2.1. Necessity of deploying target network in value-based algorithms

As the value-based algorithm is dominant in building controls based on the literature, the strategy of target network was analyzed on the performance of RL training. In Fig. 7, it is evident that without the target network, the DQN algorithm was obviously slower and fluctuated during training. Its convergence required a higher volume of data. The target network replicated the architecture of the online network but with less frequent updates to its weights. Updating the Q-values directly using a single network results in highly variable and unstable updates. As shown in Fig. 7, DDQN converged most quickly because it utilized target network to reduce overestimation by separating action selection and value estimation for computing the TD (Temporal Difference) target, showing fastest convergence with less oscillations among value-based algorithms.

#### 4.2.2. Data demand for training of each algorithm

Fig. 8 shows the amount of data required by different algorithms for the convergence. The calculation of data required for the training determined by the convergence of RL algorithms. When the fluctuation of cumulative reward stayed within 0.3 % over the past 5 consecutive episodes, the algorithms were regarded as converged and the total number of data samples used was defined as data demand for training. Generally, model-free RL algorithms presented a better control performance, but high demand on the training data, compared with model-based RL. The high data demand of RL algorithms would trouble the agent training, especially when the training happened in real systems. Thus, leveraging domain knowledge, models and virtual environment to pre-train the RL agents could facilitate the applicability of RL algorithms in building controls.

#### 4.2.3. Robustness of hyperparameter tuning during implementation

Fig. 9 shows the robustness of RL algorithms towards the setting of learning rate. Learning rate is an influential parameter of RL algorithm. In this work, 10 different settings of learning rates, as shown in Fig. 5, were tested regarding its impact on the training process. In Fig. 9, the prominent line represented the performance using the optimal learning rate, the same curves in Fig. 6. The shadowed areas were generated by applying a ±15 % deviation for each profile of training produced under each learning rate. Darker regions indicated more overlapping among multiple trainings, meaning the performance towards different settings was more centered with less differences. The PPO was the most centered in the shallowed area, indicating the best robustness to the changes of learning rates. This was because PPO directly optimized the policy, i.e., the probability distribution of actions under current state, rather than estimating all the values of current state to choose the best action in value-based methods. Also, Generalized Advantage Estimation (GAE) in PPO enabled the critic network updated efficiently. These features helped PPO being robust to a relatively wider range of learning rates. The A3C algorithm, while achieving the best control performance at the optimal learning rate, was highly sensitive to the learning rate variation as the A3C explored the state space through multiple parallel working threads, each independently updating its gradients. Facing varying learning rates, the different threads may update the global network in different directions and therefore cause the fluctuations.

### 4.3. Control performance of RL algorithms

In this section, one week of control performance results is presented using different RL algorithms in a hot summer week (19th −25th of July). Fig. 10 shows the comparison of the rewards obtained by different RL control strategies and the baseline control in the week. The baseline control computed its reward at each timestep using the same reward function (i.e., Eq. (15)). As shown in Fig. 10, all model-free algorithms achieved a relatively better and stable control performance against the Gnu-RL. The Gnu-RL used imitation learning to save 150 episodes of training samples. It is worthy of noting that the RL controllers were capable of switching on the air-conditioning system ahead of occupied time to pre-cool the indoor spaces for a better control performance, i.e., cumulative rewards. Although this would result in decrease in the reward during non-occupied time, it could be compensated and further improved by subsequent system operation under occupied periods.

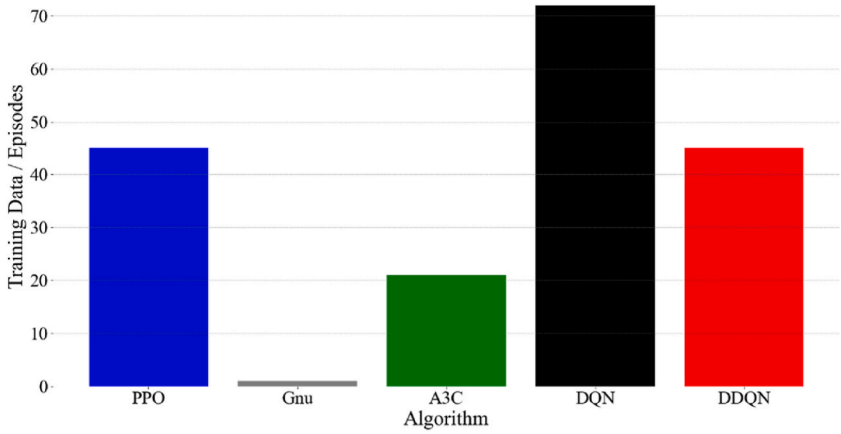



**Fig. 8.** Training data demand for the convergence of RL algorithms.

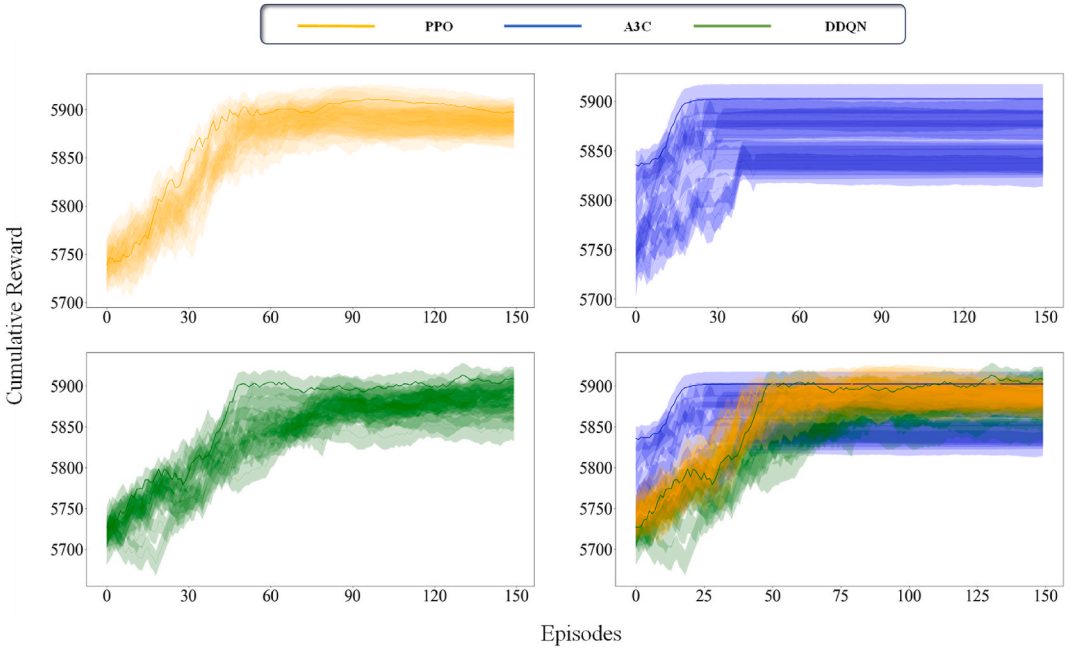



**Fig. 9.** Robustness of RL algorithms towards the changing learning rates.

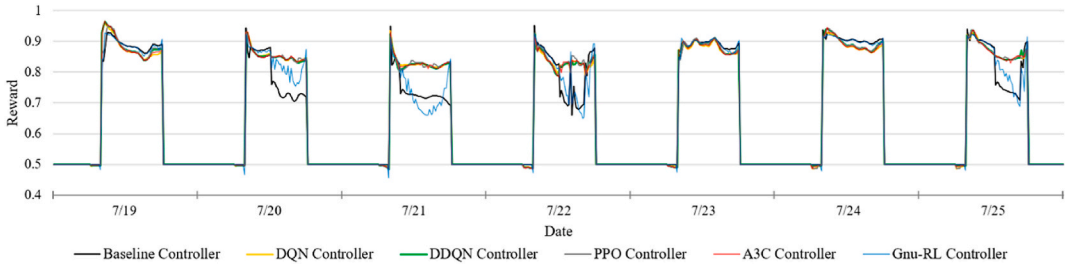



**Fig. 10.** Rewards per time step for different algorithms during a hot summer week.

To provide a better analysis of the model-based Gnu-RL algorithm, Fig. 11 shows its control behavior during the first week of testing. The Gnu-RL was a precocious RL agent without a training process, leading to a lower reward at the first two days of taking over building controls due to the overshooting of the RL agent. The Gnu-RL agent took around two days to learn to track the indoor temperature setpoint as well as the baseline controller through the online update. On the contrast, in Fig. 6, the model-free RL agents (DQN, DDQN and PPO) were around 10 episodes of training to achieve similar control performance, which was approximately equivalent to 900 days. Gnu-RL therefore traded training performance to reduce the amount of training data required.

Figs. 12 and 13 show the energy consumption and the Predicted Percentage of Dissatisfaction (PPD) in the building using different control strategies. The model-free agents had a good performance in reducing the energy consumption, especially during days with a relatively higher outdoor temperature (21st July, 22nd July and 25th July). The DDQN and PPO controllers had a better performance in reducing energy consumption. During hotter days (21st July and 22nd July), the indoor comfort level of the baseline controller and Gnu-RL agent fluctuated obviously and the model-free algorithms consistently maintained the PPD below 7.5 %. Fig. 14 illustrates the average indoor air temperature for each control strategy. On 20th July and 21st July, the baseline control strategy experienced indoor temperatures above the setpoint at the start of operation with a small peak. This does not happen with RL-based strategies, as they preemptively turned on the HVAC system to prevent overheating when occupants arrived, thus avoiding the temperature overshoot observed with baseline strategies.

Figs. 15 and 16 present supply air temperature and air flow rate delivered to the building. The baseline controller maintained the supply air temperature around 15 °C, while model-free RL agents maintained it within the range of 18–20 °C. The Gnu-RL agent continuously changed the outlet air temperature setpoint based on the current state. Additionally, the A3C algorithm exhibited more frequent but minor fluctuations in both curves, possibly due to noise added to the policy to encourage exploration or the characteristic of the algorithm's multiple threads asynchronously updating, where different threads may learn different strategies.

Table 5 showed the total rewards of five implemented algorithms compared to the baseline in the testing. The results demonstrate that all five RL algorithms could make significant improvements over baseline control. Within the model-free algorithms, A3C

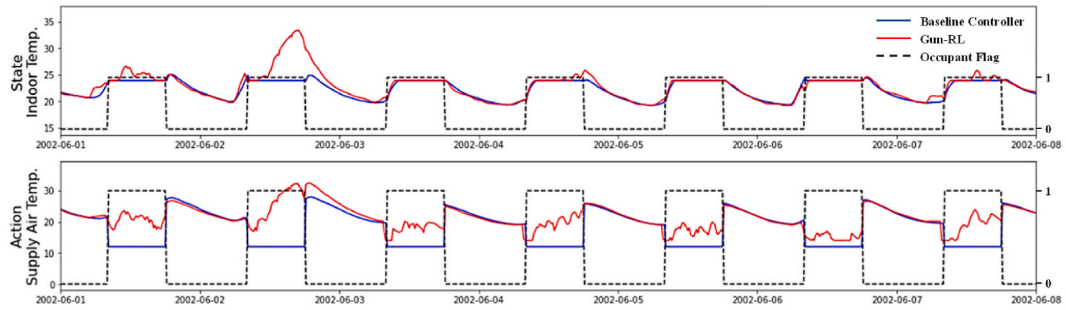



**Fig. 11.** Control behavior of the Gnu-RL during the first week of testing.

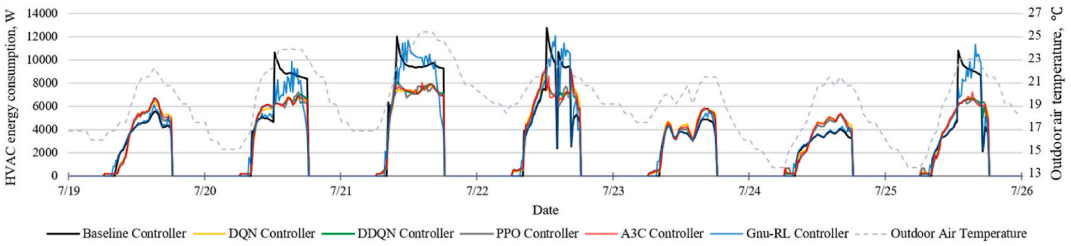



**Fig. 12.** HVAC energy consumption using different control strategies during a hot summer week.

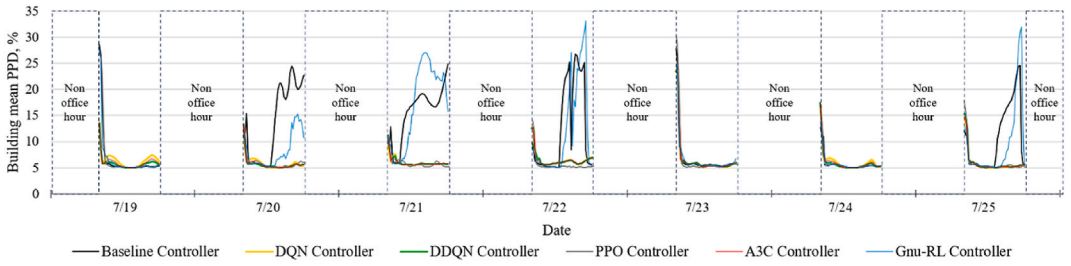



**Fig. 13.** Building mean PPD using different control strategies during a hot summer week.

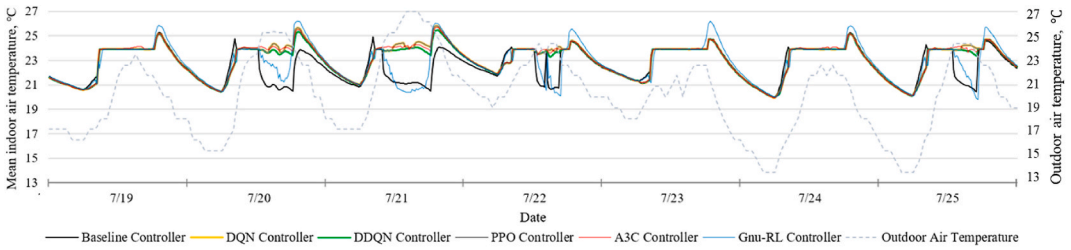



**Fig. 14.** Mean indoor temperature using different control strategies during a hot summer week.

performed the best, followed by PPO, DDQN, and DQN. The Gnu-RL, due to its characteristics, made significant progress in reducing the demand for training data with achieving satisfactory control performance.

## 5. Conclusion

In addressing control challenges within Heating, Ventilation, and Air Conditioning systems, this study has embarked on a comprehensive exploration of the application and performance of both model-based and model-free reinforcement learning algorithms considering model availability and policy representation. The results underscore the considerable potential of RL techniques in HVAC control, specifically in energy conservation and enhancing indoor comfort. The characteristics and control performance of each algorithm are demonstrated and discussed.

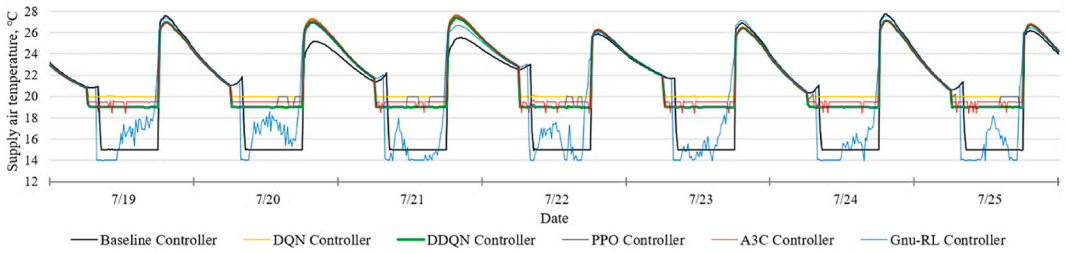



**Fig. 15.** Supply air temperature to the building using different control strategies.

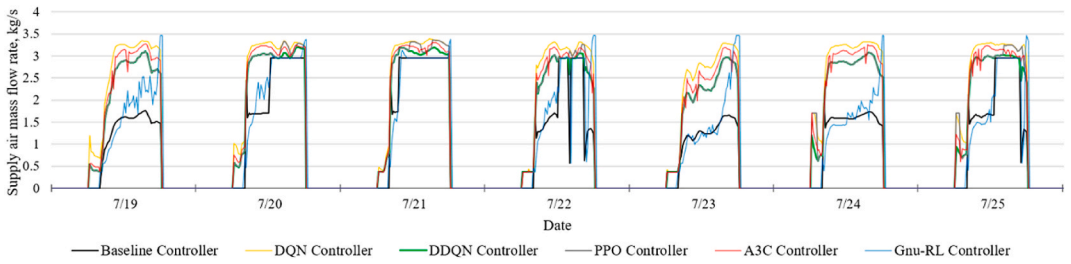



**Fig. 16.** Air flow rate delivered to the building using different control strategies.

**Table 5**
Total rewards of control performance using different control strategies.

| Control strategy | Baseline | DQN | DDQN | PPO | A3C | Gnu-RL |
|---|---|---|---|---|---|---|
| HVAC energy consumption | 18.02 GJ | 16.19 GJ | 15.91 GJ | 15.79 GJ | 16.06 GJ | 17.51 GJ |
| PPD during occupied time | 11.17 % | 6.90 % | 6.83 % | 6.83 % | 6.80 % | 8.94 % |
| Cumulative reward | 5761.87 | 5878.99 | 5886.41 | 5889.37 | 5892.25 | 5819.62 |

The Gym-Eplus framework was selected to train and test RL algorithms, including Deep Q-Network (DQN), Double Deep Q-Network (DDQN), Proximal Policy Optimization (PPO), Asynchronous Advantage Actor-Critic (A3C) and model-based Gnu-RL. The results showed that the oscillations during training in DDQN ended approximately 10 episodes earlier than in DQN, and target network could effectively reduce the training data. PPO algorithm successfully accelerated and stabilized training convergence by applying clipping mechanism to the objective function, showing best robustness to different hyperparameters. A3C algorithm exhibited a best rapid and stable convergence due to the superiority of asynchronous updates, but extra work for hyperparameter tuning and monitoring at the start of the training was needed to ensure the performance.

During the testing period, model-free and model-based RL agents could outperform the baseline control in thermal comfort and energy consumption. Model-based RL could perform as well as the baseline controller within two days of implementation, whereas model-free algorithms required simulation data equivalent to 900 days for training to achieve similar results. The RL-based controllers could learn the strategy of switching on the air conditioning earlier than the building operation time, indicating that RL algorithms were able to sacrifice short-term gains to obtain greater long-term rewards for a robust and adaptive control optimization.

One limitation of this study is that all deep reinforcement learning algorithms were trained and tested in a simulated environment. Although the trained control policy has demonstrated a certain degree of adaptability to handle the variations between training and testing models, the algorithms studied are required to test in real-world applications in the future. The study considered the model-based Gnu-RL algorithm in comparison with model-free algorithms. Other types of model-based reinforcement learning frameworks such as the Dyna and the AlphaZero are necessary to explore the control performance in buildings and analyze the difference from other RL algorithms.

Nomenclature

| | |
|---|---|
| **HVAC** | Heating, Ventilation, and Air Conditioning |
| **RBC** | Rule-Based Control |
| **PID** | Proportional Integral Derivative |
| **MPC** | Model Predictive Control |
| **RL** | Reinforcement Learning |
| **MDP** | Markov Decision Process |
| **MFRL** | Model-free Reinforcement Learning |

(*continued on next page*)

(*continued*)

| | |
|---|---|
| **PG** | Policy Gradient |
| **AC** | Actor-Critic |
| **DQN** | Deep Q-Network |
| **PMV** | Predicted Mean Vote |
| **DDQN** | Double Deep Q-Network |
| **VRF** | Variable Refrigerant Flow |
| **PPO** | Proximal Policy Optimization |
| **TRPO** | Trust Region Policy Optimization |
| **A3C** | Asynchronous Advantage Actor-Critic |
| **DDPG** | Deep Deterministic Policy Gradient |
| **MBRL** | Model-based Reinforcement Learning |
| **MCTS** | Monte Carlo Tree Search |
| **SAC** | Soft Actor-Critic |
| **VAV** | Variable Air Volume |
| **TD** | Temporal Difference |
| **GAE** | Generalized Advantage Estimation |
| **MSE** | Mean Squared Error |
| **PPD** | Predicted Percentage of Dissatisfied |
| **BCVTB** | Building Controls Virtual Test Bed |

## CRediT authorship contribution statement

**Ziyan Wu:** Writing – original draft, Methodology, Formal analysis, Data curation. **Wenhao Zhang:** Methodology, Data curation. **Rui Tang:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization. **Huilong Wang:** Writing – review & editing. **Ivan Korolija:** Writing – review & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## Acknowledgements

The research presented in this paper is financially supported by the Global Engagement Strategic Partner Funds at University College London.

## References

[1] Y. Zhang, B.K. Teoh, M. Wu, J. Chen, L. Zhang, Data-driven estimation of building energy consumption and GHG emissions using explainable artificial intelligence, Energy 262 (2023), https://doi.org/10.1016/j.energy.2022.125468.
[2] H. Wang, S. Wang, R. Tang, Development of grid-responsive buildings: opportunities, challenges, capabilities and applications of HVAC systems in non-residential buildings in providing ancillary services by fast demand responses to smart grids, Appl. Energy 250 (2019) 697–712, https://doi.org/10.1016/J.APENERGY.2019.04.159.
[3] R. Tang, S. Wang, S. Sun, Impacts of technology-guided occupant behavior on air-conditioning system control and building energy use, Build. Simulat. (2021) 209–217.
[4] G. Geng, G.M. Geary, On performance and tuning of PID controllers in HVAC systems, Proc. IEEE Conf. Control Appl. 2 (1993) 819–824, https://doi.org/10.1109/CCA.1993.348229.
[5] G. Ashrae, 36: High Performance Sequences of Operation for HVAC Systems, Am. Soc. Heating, Refrig, Air-Conditioning Eng., Atlanta, 2018.
[6] R. Tang, S. Wang, Model predictive control for thermal energy storage and thermal comfort optimization of building demand response in smart grids, Appl. Energy 242 (2019) 873–882, https://doi.org/10.1016/j.apenergy.2019.03.038.
[7] Z. Wang, T. Hong, Reinforcement learning for building controls: the opportunities and challenges, Appl. Energy 269 (2020) 115036, https://doi.org/10.1016/j.apenergy.2020.115036.
[8] Z. Nagy, G. Henze, S. Dey, J. Arroyo, L. Helsen, X. Zhang, B. Chen, K. Amasyali, K. Kurte, A. Zamzam, H. Zandi, M. Quintana, S. Mccullogh, J.Y. Park, H. Li, T. Hong, S. Brandi, G. Pinto, A. Capozzoli, Ten Questions Concerning Reinforcement Learning for Building Energy Management, 2023, p. 241.
[9] R.S. Sutton, A.G. Barto, Reinforcement Learning: an Introduction, MIT press, 2018.
[10] Q. Fu, Z. Han, J. Chen, Y. Lu, H. Wu, Y. Wang, Applications of reinforcement learning for building energy efficiency control: a review, J. Build. Eng. 50 (2022) 104165, https://doi.org/10.1016/J.JOBE.2022.104165.
[11] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, others, Human-level control through deep reinforcement learning, Nature 518 (2015) 529–533.
[12] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, D. Silver, Rainbow: combining improvements in deep reinforcement learning, in: Proc. AAAI Conf. Artif. Intell., 2018.
[13] S. Qiu, Z. Li, Z. Li, J. Li, S. Long, X. Li, Model-free control method based on reinforcement learning for building cooling water systems: validation by measured data-based simulation, Energy Build. 218 (2020) 110055.
[14] W. Valladares, M. Galindo, J. Gutiérrez, W.-C. Wu, K.-K. Liao, J.-C. Liao, K.-C. Lu, C.-C. Wang, Energy optimization associated with thermal comfort and indoor air control via a deep reinforcement learning algorithm, Build. Environ. 155 (2019) 105–117.

[15] Y.R. Yoon, H.J. Moon, Performance based thermal comfort control (PTCC) using deep reinforcement learning for space cooling, Energy Build. 203 (2019) 109420.
[16] R. Jia, M. Jin, K. Sun, T. Hong, C. Spanos, Advanced building control via deep reinforcement learning, Energy Proc. 158 (2019) 6158–6163.
[17] D. Azuatalam, W.-L. Lee, F. de Nijs, A. Liebman, Reinforcement learning for whole-building HVAC control and demand response, Energy AI 2 (2020) 100020.
[18] Z. Zhang, A. Chong, Y. Pan, C. Zhang, K.P. Lam, Whole building energy model for HVAC optimal control: a practical framework based on deep reinforcement learning, Energy Build. 199 (2019) 472–490.
[19] T. Schreiber, S. Eschweiler, M. Baranski, D. Müller, Application of two promising Reinforcement Learning algorithms for load shifting in a cooling supply system, Energy Build. 229 (2020) 110490.
[20] W. Zhang, Z. Zhang, Energy efficient operation optimization of building air-conditioners via simulator-assisted asynchronous reinforcement learning, IOP Conf. Ser. Earth Environ. Sci. (2022) 12006.
[21] C. Gao, D. Wang, Comparative study of model-based and model-free reinforcement learning control performance in HVAC systems, J. Build. Eng. 74 (2023) 106852.
[22] L. Chen, F. Meng, Y. Zhang, MBRL-MC: an HVAC control approach via combining Model-based deep reinforcement learning and model predictive control, IEEE Internet Things J. 9 (2022) 19160–19173.
[23] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, others, A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, Science 362 (2018) 1140–1144.
[24] B. Chen, Z. Cai, M. Bergés, Gnu-rl: a precocial reinforcement learning solution for building hvac control using a differentiable mpc policy, in: Proc. 6th ACM Int. Conf. Syst. Energy-Efficient Build. Cities, Transp., 2019, pp. 316–325.
[25] C. Zhang, S.R. Kuppannagari, R. Kannan, V.K. Prasanna, Building HVAC scheduling using reinforcement learning via neural network based model approximation, in: Proc. 6th ACM Int. Conf. Syst. Energy-Efficient Build. Cities, Transp., 2019, pp. 287–296.
[26] Y. Du, H. Zandi, O. Kotevska, K. Kurte, J. Munk, K. Amasyali, E. Mckee, F. Li, Intelligent multi-zone residential HVAC control strategy based on deep reinforcement learning, Appl. Energy 281 (2021) 116117.
[27] X. Ding, A. Cerpa, W. Du, Multi-zone HVAC control with model-based deep reinforcement learning, ArXiv Prepr. ArXiv2302 (2023) 00725.
[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, ArXiv Prepr. ArXiv1312 (2013) 5602.
[29] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proc. AAAI Conf. Artif. Intell., 2016.
[30] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay, ArXiv Prepr. ArXiv1511 (2015) 05952.
[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, ArXiv Prepr. ArXiv1707 (2017) 06347.
[32] P. Hämäläinen, A. Babadi, X. Ma, J. Lehtinen, PPO-CMA: Proximal policy optimization with covariance matrix adaptation, in: 2020 IEEE 30th Int. Work. Mach. Learn. Signal Process., 2020, pp. 1–6.
[33] X. Zhang, R. Chintala, A. Bernstein, P. Graf, X. Jin, Grid-interactive multi-zone building control using reinforcement learning with global-local policy search, in: 2021 Am. Control Conf., 2021, pp. 4155–4162.
[34] B. Chen, Z. Cai, M. Bergés, Gnu-rl: a practical and scalable reinforcement learning solution for building hvac control using a differentiable mpc policy, Front. Built Environ. 6 (2020) 562239.
[35] Z. Zhang, A Reinforcement Learning Approach for Whole Building Energy Model Assisted HVAC Supervisory Control, Carnegie Mellon University, 2019.
[36] Z. Zhang, K.P. Lam, Practical implementation and evaluation of deep reinforcement learning control for a radiant heating system, in: Proc. 5th Conf. Syst. Built Environ., 2018, pp. 148–157.
[37] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, ArXiv Prepr. ArXiv1606 (2016) 01540.
[38] M. Wetter, P. Haves, B. Coffey, Building Controls Virtual Test Bed, 2008.
[39] H. Gupta, R. Srikant, L. Ying, Finite-time performance bounds and adaptive learning rate selection for two time-scale reinforcement learning, Adv. Neural Inf. Process. Syst. 32 (2019).
[40] O.V.M. Virtualbox, Oracle vm virtualbox, Change 107 (2011) 1–287.
[41] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, J. Kautz, Reinforcement learning through asynchronous advantage actor-critic on a gpu, ArXiv Prepr. ArXiv1611 (2016) 06256.