Adaptive Policy Regularization for Offline-to-Online Reinforcement Learning in HVAC Control

Hsin-Yu Liu
University of California San Diego
La Jolla, CA
hyl001@ucsd.edu

Bharathan Balaji*
Amazon
USA
bhabalaj@amazon.com

Rajesh Gupta
University of California San Diego
La Jolla, CA
gupta@ucsd.edu

Dezhi Hong*
Amazon
USA
hondezhi@amazon.com

Abstract

Reinforcement learning (RL)-based control methods have been extensively studied to improve building heating, ventilation, and air conditioning (HVAC) efficiency. Data-driven approaches demonstrate better transferability and scalability, making them useful in real-world applications. Most prior works focus on online learning requiring simulators or models of environment dynamics. However, transferring thermal simulators between environments is inefficient and challenging. We build on recent works that employ offline training on static datasets from unknown policies. Pure offline RL is constrained by the replay buffer's distribution, we propose using offline-to-online RL to enhance pre-trained offline models through online adaptation to distribution shifts. We show that direct online fine-tuning deteriorates performance on offline policies. To address this, we propose automatically tuning the actor's regularization during training to optimize the exploration-exploitation tradeoff. Specifically, we leverage simple moving averages of mean Q-values sampled throughout training. Simulation experiments in building HVAC environments demonstrate our method outperforms state-of-the-art approaches under various conditions, improving performance by 32.9% and enhancing pre-trained models' capabilities online.

1 Introduction

Real-world building HVAC systems are primarily controlled via Rule-Based Control (RBC) policies that consist of a nuanced set of if-else rules crafted by domain experts. These rules do not generalize or scale to different operating environments and buildings. Control policies trained with Reinforcement Learning, on the other hand, adapt to changes in the environment. To do so, building control problems could be formulated as Markov Decision Processes (MDP) and optimized with RL methods [31]. The RL agent observes the states via sensors in the buildings (i.e., thermostats, hygrometers) and takes actions upon the control setpoints in a building management system via actuators. In this formulation, the objective of the agent is to increase energy efficiency while maintaining occupants' thermal comfort [41].

*Work unrelated to Amazon.

Tackling Climate Change with Machine Learning: workshop at NeurIPS 2024.

Much of the RL methods in prior works are trained and evaluated in the online paradigm, where an accurate simulator is required to train the RL agent. However, setting and calibrating these simulators requires expertise and is time-consuming. Also, in modern commercial buildings, the building management systems already store a significant amount of sensing and control data. These datasets capture not only the thermal dynamics of the systems but also the control policies that embody the best-known knowledge of the control defined by domain experts. Prior works have shown that offline data-driven methods can learn a more robust policy compared to the behavioral RBC policies that generated the data [19]. Nonetheless, without further online interactions, the improvement is limited by the diversity of the state-action distribution in the historical data.

Recently, researchers have explored the capability of offline-to-online (O2O) RL [22, 42, 39, 37, 18], where an offline RL policy is first learned on historical data and continues to be trained with online interactions in the deployed environment. These models are designed to adapt to the distribution drift between the static historical data and the deployment environments, continuing to improve in performance with judicious exploration. To our knowledge, O2O algorithms have not been studied in the building control domain. We show that off-the-shelf algorithms designed for other domains such as robotics degrade in performance compared to purely offline RL policies. We present a novel algorithm that overcomes these shortcomings and continues improve the deployed RL policy after deployment. We exploit the insight that building control systems do not experience rapid change in dynamics, and introduce a moving average term in the RL policy loss function to stabilize online updates. Simulation experiments with the well-established Sinergym environment show that our algorithms achieve state-of-the-art performance.

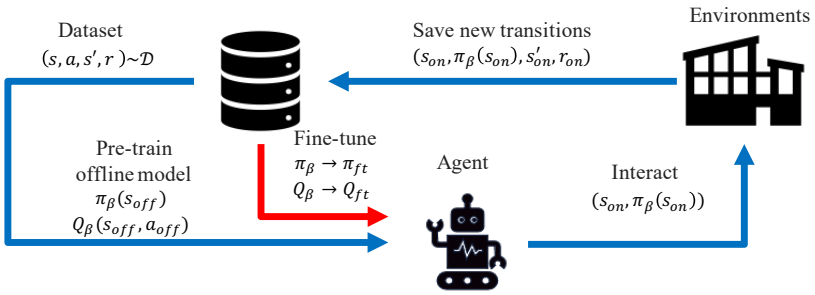

Figure 1: Flow chart of offline-to-online RL: (1.) The offline model learns from the existing dataset. (2.) After pre-training, the agent interacts with the environment online. (3.) The generated transitions are saved in the replay buffer(s) for further learning. (4.) The offline-to-online fine-tuning improves the agent's performance continuously.

We will open-source our code with a permissive license.

2 Problem Statement

The objective of the agent is to maintain a comfortable thermal environment with minimal energy use. The state consists of indoor/outdoor temperatures, time/day, occupant count, thermal comfort, and related sensor data. The action is to adjust the temperature setpoint of the thermostat. The reward is a linear combination of occupants' thermal comfort and energy consumption. The environment is a single-floor building divided into 5 thermal zones, with one interior and four exterior rooms. More details regarding the RL setup are at Appendix F.1

3 Methodology

3.1 WISMAQ Implementation

Our method WISMAQ (Weighted Increased Simple Moving Average of Q-value) is inspired by the preliminary experiments we performed with the B2RL [20] dataset. We start from the RBC buffers to train an offline RL policy using TD3+BC, and then use TD3 for offline-to-online fine-tuning. Detailed experimental results are shown in Appendix C. We observe that the trace of the episodic average Q-value is noisy and de-stabilizes policy updates.

We introduce a simple moving average (SMA) of the average Q-values in the policy update equation to yield a more statistically stable metric. We modify the policy update by adding loss term $\mathcal{L}_{WISMAQ}$:

$$\mathcal{L}_{WISMAQ} = ReLU\left(\frac{\bar{Q}^{t}_{SMA} - \psi * \bar{Q}^{t-d}_{SMA}}{\bar{Q}^{t}_{SMA} + \bar{Q}^{t-d}_{SMA}}\right) \tag{1}$$

where t is the current timestep, ψ is a hyperparameter to weight the reference Q-SMA, and d is the difference between the current timestep and the reference timestep. ReLU is the rectified linear unit activation function that automatically tunes this term based on the difference of the Q-SMA between the reference timestep and the current timestep. i.e., when $\bar{Q}^{t}_{SMA}$ is larger than $\psi * \bar{Q}^{t-d}_{SMA}$, this term is activated. On the opposite, when $\bar{Q}^{t}_{SMA}$ is smaller than $\psi * \bar{Q}^{t-d}_{SMA}$, this term is deactivated.

We build on the policy function used by the TD3+BC algorithm, and add the term $\mathcal{L}_{WISMAQ}$ to it in the following manner:

$$\pi = \arg\max_{\pi} \mathbb{E}_{(s,a)\sim\mathcal{D}}[\lambda\bar{Q}(s,\pi(s)) - (\pi(s) - \pi_{\beta}(s))^2 + \xi\mathcal{L}_{\mathbf{WISMAQ}}] \tag{2}$$

Where, ξ is a hyper-parameter for the coefficient of $\mathcal{L}_{WISMAQ}$. With the ReLU activation function, the added loss term is bounded, i.e. $\mathcal{L}_{WISMAQ}\in[0,1]$. And thus makes it auto-tuned without any heuristics.

To summarize, WISMAQ optimizes the policy by identifying if the current SMA of the averaged Q-value is higher than a previous reference SMA. If so, the WISMAQ loss term is activated, it encourages the policy to explore. Otherwise, when the WISMAQ loss term is lower than the reference value, we keep the actor loss as is since it means the Q-value is converging, i.e., leave it learning as the original offline model, which is conservatively trained with online transitions. We also adopt other two recent innovations to further increase performance: combined experience replay (CER) [40] and bootstrapped ensemble learning [24]. Appendix E has more details.

4 Experiments

We create a set of RBC buffers for the three different weather conditions. Considering two practical scenarios where we have a well-trained RL agent and an RBC policy written by human experts. This is the general case for most large commercial building sectors. (More details regarding the main experiments are at Appendix F.1).

We generate replay buffers for $250K$ time steps via Rule-Based Control policy. We follow the SinerGym [12]'s RBC policy and vectorize it for better computation efficiency (Appendix D). We train the state-of-the-art offline-to-online methods along with WISMAQ offline first for $50,000$ timesteps with the RBC buffers. We then train each algorithm online following the steps in Fig. 1 to compare the learning curves. We trained the models online for $35,000$ timesteps which is approximately one year in real-time. And the evaluation frequency is every $2,500$ timesteps. Each algorithm is run with three random seeds to capture variance in performance. We list below the baseline methods compared:

- **AWAC** [22]: Advantage Weighted Actor Critic (AWAC), it enables rapid learning of skills with a combination of prior demonstration data and online experience. The implicitly constrained actor-critic algorithm is able to both train offline and continue to improve with more experience.
- **REDQ** [42]: It combines the randomized ensemble Q-functions to improve sample efficiency along with a proportional-derivative (PD) controller to tune the hyperparameter of the weight of the behavioral cloning term α in Eq. 3 with a target score and current episodic return.
- **TD3+BC-FT**: From TD3+BC [6] to TD3 [7], we directly convert the offline TD3+BC to TD3 by removing the behavior cloning term at the beginning of the offline-to-online training, i.e., by setting $\alpha = 0$ in the fine-tuning stage.

From Figure 2 we can see that WISMAQ surpasses TD3+BC_FT in both the initial stages of the deployment, and the final perfomance towards the end. The performance WISMAQis stable across runs with lesser variance. These trends remain the same across the other two baseline algorithms, demonstrated in Appendix F. As shown in Figure 3, WISMAQ outperforms the state-of-the-art by

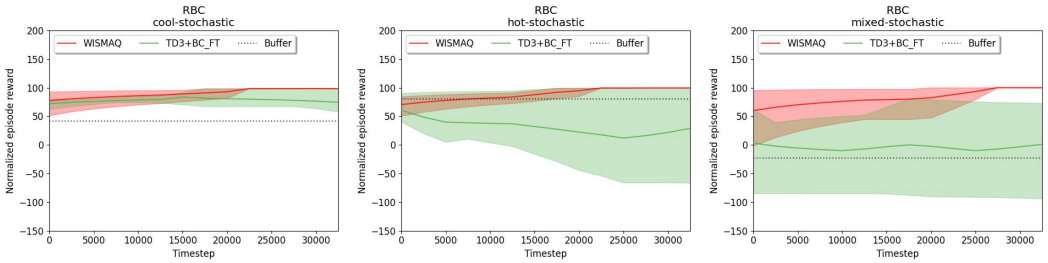

Figure 2: Learning curves of O2O models learn from RBC buffers
comparing with baseline model TD3+BC_FT

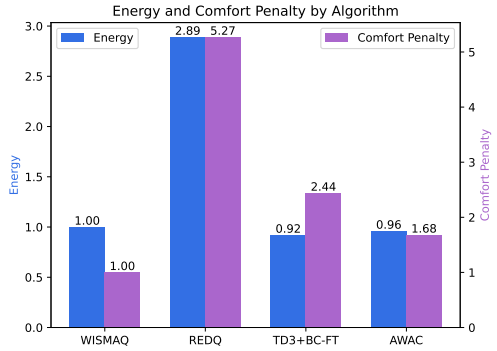

Figure 3: Compare the optimization objectives by algorithms,
WISMAQś optimization objectives are normalized as 1, and the data shown is the summation
of all six tasks across 3 different initialization conditions.

giving the best energy efficiency while maintaining the thermal comfort of occupants. In terms of reward score of the final 5 evaluations, WISMAQ improves on the state-of-the-art by 32.9%.

We have also conducted Data Efficient, Ablation, Sensitivity, and Scalability experiments. Please refer to Appendix F for more details.

5 Discussion and Conclusion

We have developed a novel approach WISMAQ to regularize the agent's policy in offline-to-online RL for use in HVAC control. It automatically tunes the actor loss that is designed to increase the average Q-value of the sampled batches in the experience replay. The use of a Simple Moving Average of the mean Q-value is key to our algorithm and could reflect the actual trend of the policy learning and its corresponding value estimation. Our experiments in the simulation environments indicate that WISMAQ not only maintains the pre-trained model capacity but also further learns a better policy as training goes on even with monotonic RBC buffers.

The limitation of our method is that with higher dimensions of the state-action spaces. The effect of the curse of dimensionality increases which might lead to less accurate value estimation. Thus, the actual trend of the mean Q-value would be inherited with higher uncertainty. However, this issue could be mitigated by learning a lower bound of the value estimation or the help of ensemble Q-network for a more accurate prediction.

We hope our study will encourage domain experts to explore the possibility of offline-to-online reinforcement learning applied in energy systems. Since a significant amount of data has been stored and should be utilized efficiently with data-driven methods.

References

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

[2] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948*, 2023.

[3] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948*, 2023.

[4] Xianzhong Ding, Wan Du, and Alberto Cerpa. Octopus: Deep reinforcement learning for holistic smart building control. In *BuildSys*, pages 326–335, 2019.

[5] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pages 3061–3071. PMLR, 2020.

[6] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

[7] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[8] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *ICML*, pages 2052–2062. PMLR, 2019.

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[10] Mengjie Han, Ross May, Xingxing Zhang, Xinru Wang, Song Pan, Da Yan, Yuan Jin, and Liguo Xu. A review of reinforcement learning methodologies for controlling occupant comfort in buildings. *Sustainable Cities and Society*, 51:101748, 2019.

[11] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[12] Javier Jiménez-Raboso, Alejandro Campoy-Nieves, Antonio Manjavacas-Lucas, Juan Gómez-Romero, and Miguel Molina-Solana. Sinergym: a building simulation and control framework for training reinforcement learning agents. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 319–323, 2021.

[13] Beomjoon Kim, Amir-massoud Farahmand, Joelle Pineau, and Doina Precup. Learning from limited demonstrations. *Advances in Neural Information Processing Systems*, 26, 2013.

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

[15] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

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

[17] National Renewable Energy Laboratory. Tmy3 datasets. `https://www.nrel.gov/docs/fy08osti/43156.pdf`, 2008.

[18] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pages 1702–1712. PMLR, 2022.

[19] Hsin-Yu Liu, Bharathan Balaji, Sicun Gao, Rajesh Gupta, and Dezhi Hong. Safe hvac control via batch reinforcement learning. In *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*, pages 181–192. IEEE, 2022.

[20] Hsin-Yu Liu, Xiaohan Fu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong. B2rl: an open-source dataset for building batch reinforcement learning. In *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 462–465, 2022.

[21] Yicheng Luo, Jackie Kay, Edward Grefenstette, and Marc Peter Deisenroth. Finetuning from offline reinforcement learning: Challenges, trade-offs and practical solutions. *arXiv preprint arXiv:2303.17396*, 2023.

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

[23] Department of Energy. Prototype building models, 2023.

[24] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[26] Bilal Piot, Matthieu Geist, and Olivier Pietquin. Boosted bellman residual minimization handling expert demonstrations. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II 14*, pages 549–564. Springer, 2014.

[27] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2016.

[28] David Silver. Lecture 3: Planning by dynamic programming. *UCL Course on RL*, 2015.

[29] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[30] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.

[31] José R Vázquez-Canteli, Jérôme Kämpf, Gregor Henze, and Zoltan Nagy. Citylearn v1. 0: An openai gym environment for demand response with deep reinforcement learning. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 356–357, 2019.

[32] Zhe Wang and Tianzhen Hong. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269:115036, 2020.

[33] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[34] Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. *arXiv preprint arXiv:2105.08140*, 2021.

[35] Lei Yang, Zoltan Nagy, Philippe Goffin, and Arno Schlueter. Reinforcement learning for optimal control of low exergy buildings. *Applied Energy*, 156:577–586, 2015.

[36] Liang Yu, Shuqi Qin, Meng Zhang, Chao Shen, Tao Jiang, and Xiaohong Guan. Deep reinforcement learning for smart building energy management: A survey. *arXiv preprint arXiv:2008.05074*, 2020.

[37] Zishun Yu and Xinhua Zhang. Actor-critic alignment for offline-to-online reinforcement learning. In *International Conference on Machine Learning*, pages 40452–40474. PMLR, 2023.

[38] Chi Zhang, Sanmukh Rao Kuppannagari, and Viktor K Prasanna. Safe building hvac control via batch reinforcement learning. *IEEE Transactions on Sustainable Computing*, 7(4):923–934, 2022.

[39] Haichao Zhang, We Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning. *arXiv preprint arXiv:2302.00935*, 2023.

[40] Shangtong Zhang and Richard S Sutton. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*, 2017.

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

[42] Yi Zhao, Rinu Boney, Alexander Ilin, Juho Kannala, and Joni Pajarinen. Adaptive behavior cloning regularization for stable offline-to-online reinforcement learning. *arXiv preprint arXiv:2210.13846*, 2022.

[43] Han Zheng, Xufang Luo, Pengfei Wei, Xuan Song, Dongsheng Li, and Jing Jiang. Adaptive policy learning for offline-to-online reinforcement learning. *arXiv preprint arXiv:2303.07693*, 2023.

Appendix

A Related Work

In this section, we summarize the related work on offline-RL, O2O-RL, and building RL controls. We summarize the comparison between different RL approaches in Table 1.

A.1 Building RL control

Prior research has demonstrated that building RL control policy could outperform RBC in both online and offline settings. Researchers have studied extensively online RL methods for HVAC control [10, 32, 36]. Zhang et. al. [41] developed a framework for whole building HVAC (heating, ventilation, air-conditioning) control in online settings to achieve a 16.7% heating demand reduction cf. RBC control. OCTOPUS holistically controls subsystems in modern buildings to get a 14.26% energy saving cf. RBC policy [4]. Yang et. al. [35] implemented an RL control for LowEx building systems with a 11.47% improvement on cumulative net power output than RBC.

With offline RL, Zhang et. al. [38] applied a state-of-the-art method and demonstrated a $12 \sim 35\%$ reduction in grid consumption change. Liu et. al. [19] incorporated a Kullback-Leibler (KL) divergence constraint during the training of an offline RL agent to penalize policies that are far away from the previous updates for stability, and achieve a 16.7% of energy reduction cf. the default RBC control. To our knowledge, we are the first to study O2O-RL in building HVAC control.

A.2 Offline RL

In many real-world settings, we have access to data generated with an existing 'behavioral' policy (RBC policy in our work) when there are no established simulators of the environment. These logged interactions are saved as experience replay or replay buffers. Offline RL learns exclusively from existing static datasets without interacting with an environment. Due to the lack of accurate value estimation of out-of-distribution (OOD) state-action pairs, these methods learn a more conservative policy or a pessimistic lower bound of the value estimation.

BCQ [8] mitigates the extrapolation errors induced by OOD actions via a variational autoencoder. BEAR [15] uses ensemble Q-functions to reduce the accumulation of bootstrapping errors. BRAC [33] regularizes the learned policy towards the behavioral policy with a KL divergence constraint between the distributions over actions. CQL [16] learns a lower bound of the true Q-function with SAC [9]-style entropy regularization. TD3+BC [6], derived from TD3 [7], uses a behavioral cloning regularization for policy learning. UWAC [34] down-weights the OOD state-action pairs' contribution to the training.

A.3 Offline-to-Online RL

Offline-to-online (O2O) RL follows the same assumption as offline RL, i.e., there is no access to the simulator of the system. However, we could further improve the model with online interactions since the pure offline method cannot yield accurate value estimation of the OOD state-action values. Hence, the goal is to enhance the capability of the model with online training without learning from scratch as in the traditional online setting.

Table 1: Comparison of different RL approaches

RL approaches	**Online**	**Offline**	**Offline-to-Online**
Strength	Ability to explore	Learning from existing policies	Improving from pre-trained policies
Weakness	Learning from random policies	Improvement is limited by behavioral policies	Suffering distribution drifts
Prerequisite	Models/Simulators	Experience replay	Pre-trained models

A.3.1 RL with Offline Data

Previous studies focus on online RL boosted with offline data. One branch in this research area is RL with Expert Demonstrations (RLED) with the assumption that a pre-trained offline model may not be necessary. APID [13] leverages few and/or sub-optimal demonstration data used as suggestions to guide the optimization performed by approximate policy iteration. DQfD [11] leverages demonstration data to accelerate the online learning process. Piot et al. [26] proposes a method to minimize the optimal Bellman residual guided by constraints defined by the expert demonstrations. Recently, RLPD [2] extends standard off-policy RL and achieves state-of-the-art online performance on a number of tasks using offline data not limited to expert prior knowledge [3]. This branch of research differs from our study in that we focus on fine-tuning the pre-trained offline models and not training the models from scratch with offline data to accelerate the learning.

A.3.2 Online fine-tuning with offline pre-training

Another branch, which is similar to our proposed method, assumes we fine-tune offline models in online settings to adapt to distribution drift and optimize the exploration-exploitation trade-offs. AWAC [22] trains an advantage-weighted actor-critic with an implicit policy constraint that balances replay between offline and online buffers, a pessimistic Q-ensemble [18], and a density ratio estimator to improve sample efficiency and prevent over-optimism. PEX [39] freezes the pre-trained offline model, expands the policy set with the fine-tuning model, and constructs a categorical distribution for selecting the final action. APL [43] obtains near-on-policy data and chooses an optimistic update strategy. On the other hand, it uses a pessimistic update strategy for sampled offline data. REDQ [42] uses randomized ensemble Q-functions to increase sample efficiency and adaptive hyperparameter tuning to adjust the degree of behavioral policy regularization with a normalized target episode reward. ACA [37] introduces a reconstruction of Q-functions for online fine-tuning as an alignment step so it is tamed to be consistent. TD3-C [21] considers conservative policy optimization as the approach for stabilizing finetuning when the offline dataset lacks diversity.

To our knowledge, we are the first to study offline-to-online reinforcement learning for building control systems. Prior methods developed for other domains need at least one of the following requirements that makes them resource-consuming [37, 42, 18, 22, 21] and most of them fail to maintain the pre-trained performance in building environments:

- **R.1** Require information on absolute scores of expert and random agents, typically not available for buildings.
- **R.2** Many suffer policy collapse at the very beginning of the transition from offline mode to online mode.

R.3 Introduce compute overhead with additional models and/or replay buffers.

To summarize, the key contributions of our work are:

- A fine-tuning based offline-to-online RL algorithm that maintains the pre-trained model's ability and continues to improve with online interactions (to tackle **R.1** and **R.2**).
- The add-on methods - Combined Experience Replay (CER [40]) and Bootstrapped Ensemble [24] help adapt the distribution drift with extreme low cost of $\mathcal{O}(1)$ time complexity.
- Our method requires no extra models, only a single replay buffer and works without differentiating the offline and online transitions (to tackle **R.3**).

The details of our method will be elaborated in Sec.3.

B Background

B.1 Reinforcement Learning

Reinforcement Learning problems are formulated as a Markov Decision Process (MDP), a sequential decision-making problem that aims to maximize the discounted cumulative reward. The MDP consists of a tuple: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathbb{P}$ is the transition dynamics. The next state $s_{t+1} \sim p(\cdot|s_t, a_t)$, is decided by the current state and the action selected by a policy $\pi(a|s), \pi : \mathcal{S} \to \mathcal{A}$ either in a stochastic or a deterministic fashion. The reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}, r \in \mathbb{R}$ is mapped as a scalar, and the discount factor $\gamma \in [0, 1]$. The agent's goal is to optimize the policy to maximize the discounted accumulated return: $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$ [29].

B.2 Offline RL training

In offline training, the replay buffer $\mathcal{D}$ is generated by an unknown behavioral policy (or a combination of multiple policies): $\pi_\beta(s)$. Then the offline model aims to learn an improved policy without interacting with the environment within the confined state-action visitations. Thus, when the trained offline RL policy is deployed in the real environment, it avoid any OOD actions that may lead to inaccurate value estimation due to extrapolation errors.

Our offline training follows TD3+BC [6], which is an offline version of TD3 [7]. The learned policy is regularized with a behavior cloning term to penalize policies that deviate from historical actions. The policy loss function is the following:

$$\pi = \arg\max_\pi \mathbb{E}_{(s,a)\sim\mathcal{D}}[\lambda \bar{Q}(s, \pi(s)) - (\pi(s) - \pi_\beta(s))^2] \tag{3}$$

$$\lambda = \frac{\alpha}{\frac{1}{N}\sum_{(s_i,a_i)} |Q(s_i, a_i)|}$$

Where the Q-value following the policy π given state s, action a, reward received at time t denoted as R_t, the transition probability p_π with policy π

$$Q^\pi(s, a) = \mathbb{E}_{s_i\sim p_\pi, a_i\sim\pi}[R_t|s, a]$$

And the Q-update follows:

$$Q^\pi(s, a) = r + \gamma \mathbb{E}_{s',a'}[Q^\pi(s', a')], a' \sim \pi(s')$$

Where N is the size of the minibatch, $\bar{Q}$ is the average Q-value in the sampled batch given s and $\pi(s)$, $\pi_\beta(s)$ denotes the behavioral policy given s, and α is a hyperparameter to balance between the online exploration and the exploitation of the behavioral policy.

C Preliminary Experiment

In Offline RL, we repeatedly sample a mini-batch from the replay buffer which stores the historical data, compute the loss function, and update the policy. After the policy converges, i.e., the loss function stabilizes, we switch to the O2O stage where the actions occur in the live building environment.

The actions taken by the updated policy for given states are stored back into the replay buffer, and the TD3 loss function is used to fine-tune the policy. Intuitively, the agent who learns a better policy will generate the transitions that have higher rewards, it can be validated by the estimations from the critic, i.e., higher Q-values.

During the offline-to-online stage, for each timestep i, we add up all the mean Q-values of the sampled data to get $\bar{Q}_{\mathcal{B}_i}$, the average Q-value of the batch. Then, when an episode ends at time t_e we store the episodic mean $\bar{Q}_e = \frac{1}{t_e}\sum_{i=1}^{t_e} \bar{Q}_{\mathcal{B}_i}$. We find that the value of the mean Q-value varies with each timestep, and introduce a new term called Q-SMA that represents the simple moving average of the mean Q-value. The Q-SMA for the timestep t:

$$\bar{Q}^t_{SMA} = \frac{\bar{Q}^t + \bar{Q}^{t-1} + ... + \bar{Q}^{t-w}}{w} \tag{4}$$

where w is the window size we use for calculating the SMA.

We plot the average episodic mean Q-values over time in Fig. 4 (where the solid blue curves are the average values and the shaded regions are the min/max range), and the red curve is the Q-SMA. The averaged Q-value of the sampled batch is noisy due to the random sampling from the buffers. However, the Q-SMA is more stable cf. the raw average Q-value. In this example, the agent fails to learn a better policy, thus, the Q-value decreases as training continues. On the contrary, when the historical data is from a random control policy, the agent continues to learn better policies as training goes on as shown in Fig. 5. Therefore, we observe that the average Q-value increases as training proceeds for buffers with lower behavioral policy performance. For the expert task, the model is unable to learn a better policy to yield a higher estimated Q-value during the training [28].

The mean Q-value in the batch is noisy due to random sampling and inherited uncertainty within the models, and destabilizes update to the policy. When deployed in building HVAC environments, we observe performance collapse as the agent encounters unseen environment state-action distributions. From this observation, we develop a modified loss function that updates the policy based on an increasing simple moving average (SMA) Q-value.

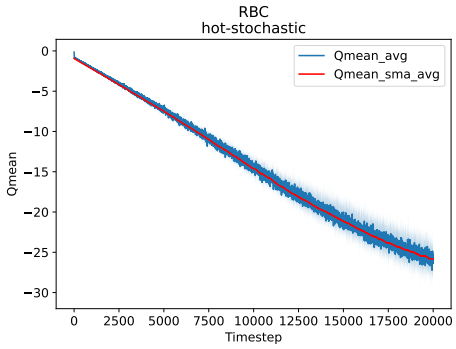

Figure 4: The averaged Q-value of the batches sampled from the RBC buffer during the training, the agent failed to improve its policy, thus the mean Q-value decreases.

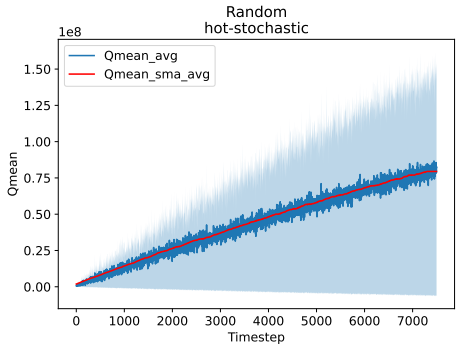

Figure 5: The averaged Q-value of the batches sampled from the random buffer during the training, the agent learns better policies, the mean Q-value is increased over time.

D Algorithms

Details of the WISMAQ algorithm:

And the RBC:

Algorithm 1: WISMAQ offline-to-online fine-tuning

Load pre-trained offline model as K ensemble double-Q networks $\{Q_{i,\theta_1}, Q_{i,\theta_2}\}_{i=1}^{K}$, the actor network π_ϕ, with random parameters $\{\theta_{i,1}, \theta_{i,2}\}_{i=1}^{K}$, ϕ, target networks $\{\theta'_{i,1} \leftarrow \theta_{i,1}, \theta'_{i,2} \leftarrow \theta_{i,2}\}_{i=1}^{K}$, $\phi' \leftarrow \phi$, policy update frequency f, horizon T, replay buffer $\mathcal{B}$
for $t = 0$ *to* T **do**
  Select actions with exploration noise
  $a \sim \pi_\phi(s) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$
  Observe reward r and next state s'
  Store transition $t = (s, a, r, s')$
  Delete the oldest one in $\mathcal{B}$ **for** $i = 1$ *to* K **do**
    Sample N transitions (s, a, r, s') from $\mathcal{B}$ in which $t \subseteq \mathcal{B}$ (blueCER)
    $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
    $y \leftarrow r + \gamma \min_{j=1,2} Q_{\theta'_{i,j}}(s', \tilde{a})$
    Update critics
    $\theta_{i,j} \leftarrow \arg\min_{\theta_{i,j}} N^{-1} \sum (y - Q_{\theta_{i,j}}(s, a))^2$
  if t *mod* f **then**
    Update ϕ by policy gradient:
    Policy update follows Eq. 1, 4, and 2(blueBootstrapped Ensemble-Q and WISMAQ)
    Calculate $\nabla_\phi J(\phi)$
    Update target networks:
    for $i = 1$ *to* K **do**
      $\theta'_{i,j} \leftarrow \tau\theta_{i,j} + (1 - \tau)\theta'_{i,j}$
    $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

Algorithm 2: Rule-based control policy

Input : Current datetime $current\_dt$, indoor air temperature IAT, zone thermostat heating setpoint temperature a_h, and zone thermostat cooling setpoint temperature a_c, obtained from the states with size N
Output : Actions selected by RBC
for i *in* N **do**
  $season\_comfort\_zone_i = get\_season\_comfort(current\_dt_i)$
  if $IAT_i >= max(season\_comfort\_zone_i)$ **then**
    $a_{h_i} = a_{h_i} - 1$
    $a_{c_i} = a_{c_i} - 1$
  if $IAT_i < min(season\_comfort\_zone_i)$ **then**
    $a_{h_i} = a_{h_i} + 1$
    $a_{c_i} = a_{c_i} + 1$
  $a_i = (a_{h_i}, a_{c_i})$
  if $current\_dt_i.weekday \geq 5$ *or* $current\_dt_i.hour$ *in range(22,6)* **then**
    $a_i = (18.33, 23.33)(°C)$

E Methods for Adapting Distribution Drift

E.1 Adapting to Distribution Drift

Another critical challenge in O2O transition is the distribution drift. Previously, several methods were proposed to accelerate RL learning by utilizing replay buffers [40, 27, 5]. We found some of them are suitable to deal with the distribution drift in the O2O setting. The first one we adopt is the combined experience replay (CER) [40], it adds the latest transition to the sampled batch in every training step and speed up the learning. Intuitively, it forces the model to learn from the latest state-action distribution of the environment combined with the previous ones.

The other technique we applied is to remove the oldest transition in the buffer faster by setting a smaller number of transitions stored in the replay buffer since it is implemented in queues [5]. When a policy is learning and improving, the transitions generated by the old policies might harm the convergence of the model due to its inferior performance cf. the current policy. Especially in

off-policy settings, we learn from the behavioral policy via replay buffer. Observations from our experiments (Sec. F.3) indicate that the performance of the models consistently improves with the reduced age of the oldest policy [5]. The CER technique could be implemented with minimal changes with only $\mathcal{O}(1)$ time complexity. And setting a smaller buffer size requires no modification of the algorithm itself, which is efficient and reasonable in O2O training. We detail all the steps of our method in Algorithm 1.

E.2 Bootsrapped Ensemble Learning

Due to the need for exploring uncharted state-action spaces, almost all previous O2O studies take advantage of certain kinds of ensemble learning. We use K bootstrapping ensemble double-Q networks via different combinations of the randomly sampled batches in each iteration of critic training, then we randomly select a value network at each policy training iteration. This is inspired by the bootstrapped DQN for deep exploration [24]. The authors claim that randomized value functions offer a promising approach to efficient exploration with generalization. Our experiment results agree with the statement with fewer variances across different runs and could learn faster than other methods.

F More Experiments

F.1 Details of Main Experiments

We conducted our experiments with the building RL simulation environments [12]. The environment is a single-floor building with an area of $463.6m^2$ divided into 5 zones, 1 interior, and 4 exteriors. The HVAC system is a packaged VAV (variable air volume) (DX (direct expansion) cooling coil and gas heating coils) with fully auto-sized input. The simulation period of one episode is a full year. The weather types are classified according to the U.S. Department of Energy (DOE) standard [23]. The weather type details and their representative geometric locations are listed below based on TMY3 datasets [17]:

- **Cool marine**: Washington, USA. The mean annual temperature and mean annual relative humidity are 9.3°C and 81.1% respectively.
- **Hot dry**: Arizona, USA with mean annual temperature of 21.7°C and a mean annual relative humidity of 34.9%
- **Mixed humid**: New York, USA with a mean annual temperature of 12.6°C and a mean annual relative humidity of 68.5%

We set up the following Markov Decision Process for the agent:

- **State**: Site outdoor air dry bulb temperature, site outdoor air relative humidity, site wind speed, site wind direction, site diffuse solar radiation rate per area, site direct solar radiation rate per area, zone thermostat heating setpoint temperature, zone thermostat cooling setpoint temperature, zone air temperature, zone thermal comfort mean radiant temperature, zone air relative humidity, zone thermal comfort clothing value, zone thermal comfort Fanger model PPD (predicted percentage of dissatisfied), zone people occupant count, people air temperature, facility total HVAC electricity demand rate, current day, current month, and current hour.
- **Action**: Heating setpoint and cooling setpoint in continuous settings for the interior zones.
- **Reward**: We follow the default linear reward setting, which considers the energy consumption and the absolute difference to temperature comfort.

  The reward function is described below:

$$r_t = -\omega\lambda_P P_t - (1-\omega)\lambda_T(|T_t - T_{up}| + |T_t - T_{low}|) \tag{5}$$

  where P_t represents power consumption; T_t is the current indoor temperature; T_{up} and T_{low} are the imposed comfort range limits (penalty is 0 if T_t is within the range); ω is the weight assigned to power consumption. Finally, λ_P and λ_T are scaling constants for energy consumption and comfort, respectively.

- **Episode:** The time between each action, called a timestep, is 15 minutes. The episode length is 1 year, after which the accumulated reward is reset.

Table 2: Scores comparison between WISMAQ and other state-of-the-art methods. Scores are normalized such that the expert policy is 100 and the random policy is 0. The format is avg.±std. between 3 random initialization conditions of the final 5 evaluations. The highest score in a particular task is highlighted in bold font.

Task/Algo.	AWAC	WISMAQ	REDQ	TD3+BC-FT
RBC-hot	75.9±21.5	**99.7±0.1**	74.7±3.4	26.9±69.3
RBC-mixed	42.7±42.2	**100.2±0.2**	91.6±13.7	5.3±73.7
RBC-cool	94.6±2.5	**98.6±0.1**	58.3±20.6	72.6±22.1
Sum	213.2±66.2	**298.5±0.4**	224.6±37.7	104.8±165.1

Table 3: Scores comparison on the first 5 evaluations from offline to online

Task/Algo.	AWAC	WISMAQ	REDQ	TD3+BC-FT
RBC-hot	30±53.3	**77.1±11.2**	74.3±7.9	45.1±34
RBC-mixed	75.3±14.6	69.2±33.9	**87.7±10.2**	-4.4±58.1
RBC-cool	32.3±25	**82.3±14.3**	44.7±15.9	75.8±5
Sum	137.6±92.9	**228.6±59.4**	206.7±34	116.5±97.1

Table 3 demonstrates the ability of WISMAQto adapt the distribution drift from offline datasets to live online environments.

F.2 Data Efficiency Experiment

We executed a series of data efficiency experiments to observe how the amount of data affects the training. We train with three different buffer sizes containing one year of data, four months, and one week:

- **Offline WISMAQ**: Pure offline training, RBC buffers are downsampled to the varied sizes as mentioned.
- **Online WISMAQ**: Pure online training, buffers initialized with zero transitions.
- **RBC**: Deploy RBC policy and evaluate.

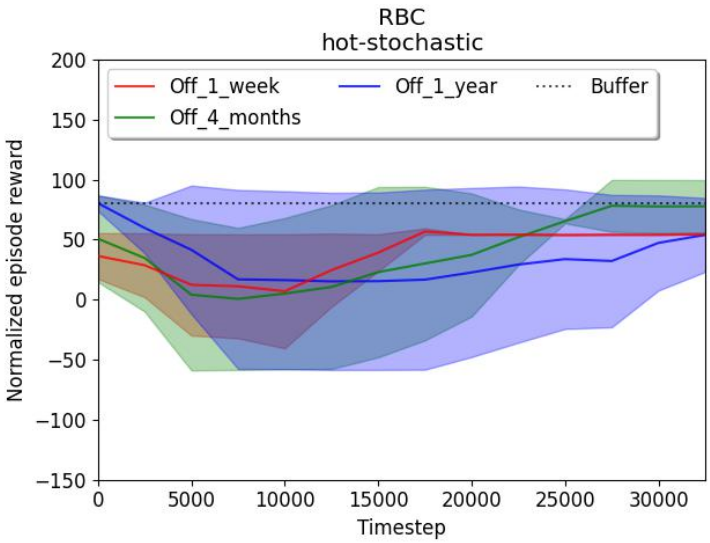

Figure 6: Offline training with varied max sizes of buffer.

Fig. 6 demonstrates that with pure offline WISMAQ training, we simulate the scenarios of different amounts of accessible data:{1 week, 4 months, and 1 year}. It is intuitive that with the smaller size of buffers, the agent would learn faster since as training continues, the better policies generate a higher quality of experience replay [5]. However, it comes with a less stable policy and could lead to catastrophic forgetting (1 week). And with too much data the model would learn from the old distribution which might damage the performance (1 year). Finding the optimized size of the buffer is also a crucial factor in off-policy learning.

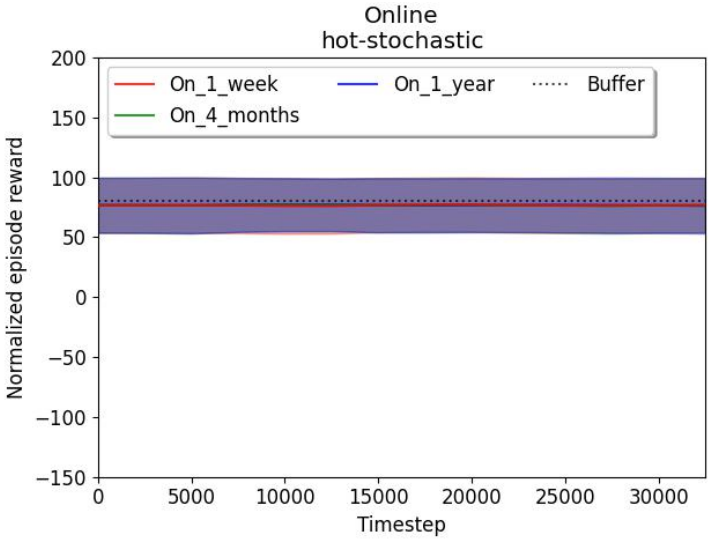

Figure 7: Online training with varied max sizes of buffer.

In Fig. 7 pure online WISMAQ are not able to learn a better policy with varied sizes of the buffer (all three learning curves overlapped). Compared with our main experiments where agents were pre-trained with offline data they have a better knowledge of entire state-action spaces and their corresponding values. It indicates WISMAQ requires prior knowledge of the environment for a more promising outcome.

F.3 Ablation Experiment

To demonstrate the ability of our add-on methods, we conducted a series of ablation experiments to validate their necessity (Fig. 8). We run WISMAQ "no_cer", and "no_WISMAQ" on the same task - cool weather with RBC buffers, the result indicates that without WISMAQ the model learns similarly to an offline model. Without CER, the model could adapt to the latest distribution drift and finally fail to improve itself in the long run. Combining these methods altogether will boost the entire learning ability than separately.

F.4 Sensitivity Experiment

Deep Reinforcement Learning methods are known for their sensitivity to hyperparameter settings [1]. Thus, we have conducted a series of experiments to demonstrate how the hyperparameter settings affect the models' performances.

To experiment on the number of bootstrapping ensemble models, Fig. 9 indicates the higher number of ensemble models yield an overall faster convergence of the policy. We should consider the trade-off between computation resources and the convergence speed to decide the number of ensemble model K.

For the hyperparameter ξ which weights the WISMAQ loss term in policy training, Fig. 10 shows with smaller weight $\xi = 1$, the model is unable to learn a better policy because the behavioral cloning term dominates. While higher ξ value leads to a more greedy fashion during policy learning, it might suffer from the inaccuracy of the value estimation. Thus, it is recommended to optimize this hyperparameter for each environment.

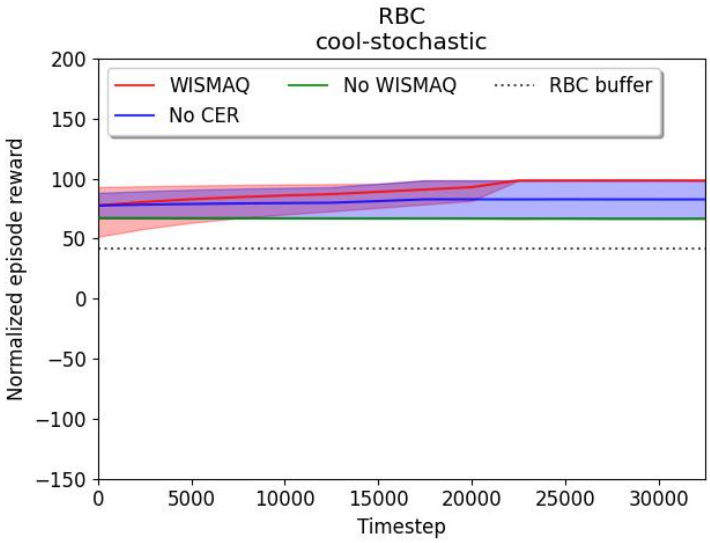

Figure 8: Ablation experiment.

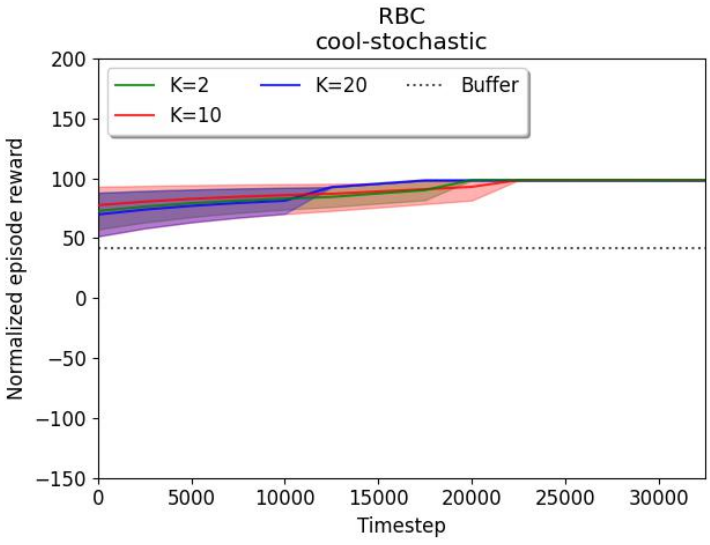

Figure 9: Sensitivity experiment on hyperparameter K, the number of ensemble models.

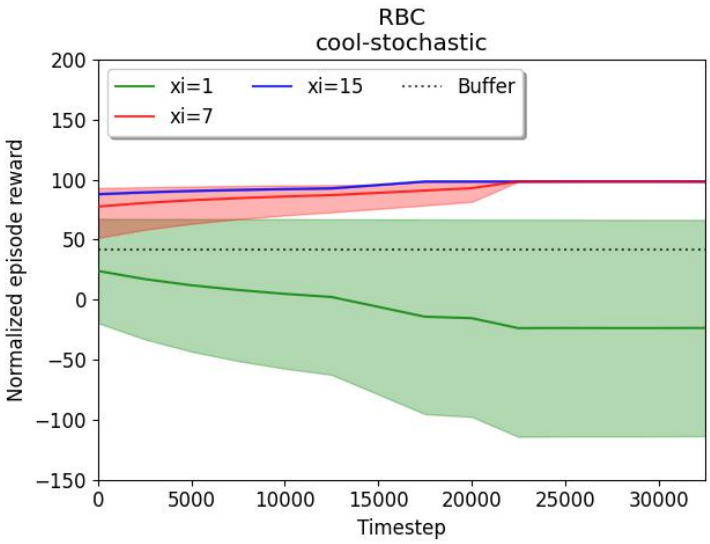

Figure 10: Sensitivity experiment on hyperparameter ξ, the weight of the WISMAQ loss term in actor training.

F.5 Scalability Experiment

Furthermore, we want to examine the scalability of our model when deployed to different environment settings. We conducted experiments with another environment - A datacenter. This environment contains 29 state space dimensions and 4 action space dimensions (for details please see Appendix.I). As the experimental result demonstrates, WISMAQ can learn a policy that is similar to expert policy while other methods could not adapt to the distribution drift as training goes on.

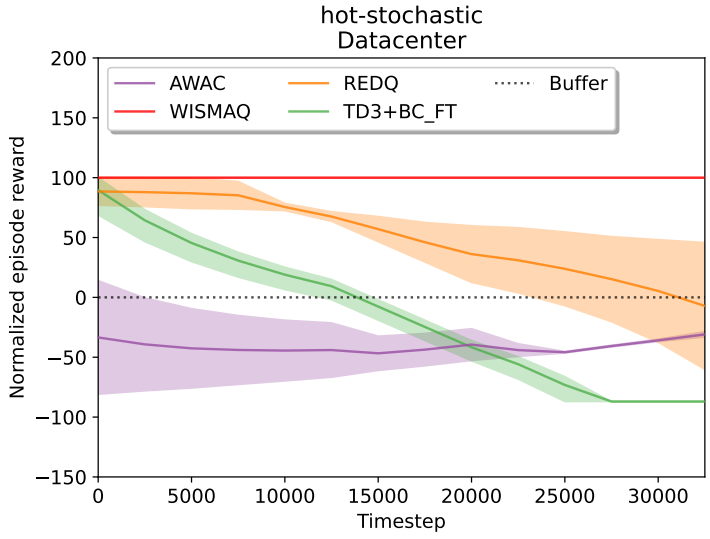

Figure 11: Scalability experiment with data center environment.

Table 4: Scores comparison of the scalability experiment.

Datacenter	AWAC	WISMAQ	REDQ	TD3+BC_FT
RBC-hot	-39.5±2.0	**100.0±0.0**	13.7±37.9	-78.0±3.7

G Experiment Details

In this section, we record the software configuration, algorithm implementation, hardware configuration, training/evaluation details, and the hyperparameters settings in our experiments for reproducibility.

- **Software**
  - **Python**: 3.9.12
  - **Pytorch**: 1.12.1+cu113 [25]
  - **Numpy**: 1.23.1 [30]
  - **CUDA**: 11.2
- **Algorithm implementation**
  - **TD3+BC**: Author-provided implementation
  - **REDQ**: Author-provided implementation
  - **AWAC**: d3rlpy
- **Hardware**
  - **CPU**: Intel Xeon Gold 6230 (2.10 GHz)
  - **GPU**: NVidia RTX A6000

H Hyperparameter Settings

Table 5: WISMAQ, TD3+BC hyperparameters

	Hyperparameter	Value
Algorithm hyperparameters	Optimizer	Adam [14]
	Critic learning rate	$3e^{-4}$
	Actor learning rate	$3e^{-4}$
	Mini-batch size	256
	Discount factor	0.99
	Target update rate	$5e^{-3}$
	Policy noise	0.2
	Policy noise clipping	(-0.5, 0.5)
	Policy update frequency	2
	Weight of BC term (α)	2.5
	Weight of WISMAQ(ξ)	7
	Weight of reference Q-SMA (ψ)	0.5
	Number of ensemble-Q (K)	10
	Window for SMA (w)	500
	Reference SMA distance (d)	500
Network architecture	Critic hidden dimension	256
	Critic hidden layers	2
	Critic activation function	ReLU
	Actor hidden dimension	256
	Actor hidden layers	2
	Actor activation function	ReLU

I RL Setup of Data Center Environment

- **State**: Year, month, day, hour, site outdoor air drybulb temperature(Environment), site outdoor air relative humidity(Environment), site wind speed(Environment), site wind direction(Environment), site diffuse solar radiation rate per area(Environment), site direct solar tadiation tate per area(Environment), zone thermostat heating setpoint temperature(West Zone), zone thermostat cooling setpoint temperature(West Zone), zone air temperature(West Zone), zone thermal comfort mean radiant temperature(West Zone PEOPLE), zone air relative humidity(West Zone), zone thermal comfort clothing value(West Zone PEOPLE),

zone thermal comfort Fanger model PPD(West Zone PEOPLE), zone people occupant count(West Zone), people air temperature(West Zone PEOPLE), zone thermostat heating setpoint temperature(East Zone), zone thermostat cooling setpoint temperature(East Zone), zone air temperature(East Zone), zone thermal comfort mean radiant temperature(East Zone PEOPLE), zone air relative humidity(East Zone), zone thermal comfort clothing value(East Zone PEOPLE), zone thermal comfort Fanger model PPD(East Zone PEOPLE), zone people occupant count(East Zone), people air temperature(East Zone PEOPLE), and facility total HVAC electricity demand rate(Whole Building)

- **Action**: Heating setpoint and cooling setpoint of West and East zones in continuous settings for the interior zones.
- **Reward**: We follow the default linear reward setting, which considers the energy consumption and the absolute difference to temperature comfort.