RULE-BASED POLICY REGULARIZATION FOR REINFORCEMENT LEARNING-BASED BUILDING CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Rule-based control (RBC) is widely adopted in buildings due to its stability and robustness. It resembles a behavior cloning methodology refined by human expertise. However, it is unlikely for RBC to exceed a reinforcement learning (RL) agent's performance as deep RL model constantly evolves and is scalable. In this paper, we explore how to incorporate rule-based control into reinforcement learning to learn a more robust policy in both online and offline settings with a unified approach. We start with state-of-the-art online and offline RL methods, TD3 and TD3+BC, then improve on them using a dynamically weighted actor loss function to selectively choose which policy RL models to learn from at each time step of training. With experiments across various weather conditions in both deterministic and stochastic scenarios, we empirically demonstrate that our rule-based incorporated control regularization (RUBICON) method outperforms representative baseline methods in offline settings by 40.7% and by 49.7% in online settings for building-RL environments.

1 INTRODUCTION

Most buildings implement rule-based control via building management systems, adjusting the setpoints of actuators to co-optimize occupants' thermal comfort and energy efficiency. These rule-based control systems codify the problem-solving know-how of human experts, akin to behavioral cloning policy learnt from expert demonstration without randomness and uncertainty (Hayes-Roth, 1985). While stable, such control lacks the flexibility to evolve over time.

Much research has demonstrated that RL can outperform RBC in both online and offline settings. Zhang et al. (2019) developed a framework for whole building HVAC (heating, ventilation, air-conditioning) control in online settings. Liu et al. (2022) incorporated Kullback-Leibler (KL) divergence constraint during training of an offline RL agent for stability, and deployed the policy in a real building. We focus on improving upon RL algorithms for HVAC control in buildings where a rule-based policy already exists. We use established building-RL simulation environments for our experiments, in both online and offline settings (Jiménez-Raboso et al., 2021).

Reinforcement learning is traditionally studied as an online paradigm. However, in real-world problems, configuring an accurate simulator or training environment dynamic models might be infeasible or time-consuming. Batch reinforcement learning (BRL, also known as offline reinforcement learning) learns policies using only historical data without simulators to interact with during training. RL regularization methods are typically tailored specifically to online or offline setting. For example, online methods encourage exploration to either improve the estimate of non-greedy actions' values or explore a better policy (Haarnoja et al., 2018; Ziebart et al., 2008; Haarnoja et al., 2017). On the other hand, offline methods favor exploitation since it is unlikely for BRL models to accurately estimate uncharted state-action values with a static dataset (Fujimoto et al., 2019; Wu et al., 2019).

In our work, we explore the questions: *Can we incorporate an existing rule-based control policy into training of a reinforcement learning policy to improve performance? Can this method be implemented in both online and offline settings?* TD3+BC (Fujimoto & Gu, 2021) makes minimal changes to convert an online method TD3 (Fujimoto et al., 2018) to offline mode with comparable performance as state-of-the-art BRL methods only by adding a behavior cloning term to regularize the policy. In TD3+BC, learning behavioral policy relies on historical data. We build on this idea

to regularize RL policy using an existing RBC policy combined with the behavioral policy. Our method can be incorporated into existing actor-critic RL algorithms with minimal changes.

RUBICON considers RBC as a safe policy based on which RL training can be improved. The actor selectively trains on either RBC or behavioral policy, depending on which policy yields a higher averaged Q-value in a mini-batch estimated by the critic network. Our proposed approach is distinct from prior works in three aspects: (1) We develop a unified regularization approach for both online and offline RL methods with minimal algorithmic modification. (2) Rule-based control policy is directly incorporated into the policy update step to provide stability and robustness. (3) We introduce a dynamic weighting method in actor-critic settings. The actor loss is varied from time step to time step depending on the Q-value estimate of behavioral policy and RBC policy predicted from the value networks. We empirically demonstrate that our method outperforms state-of-the-art methods in offline settings and improves on TD3 in online training in HVAC control environments. To our knowledge, previously RBC is only used as hard constraints or heuristics in RL settings, and we are the first to incorporate an existing reference policy directly into actor-critic algorithms.

2 Related Work

Rule-based systems Rule-based system is one of the first artificial intelligence (AI) methods that solves many real-world problems. *Planner* (Hewitt, 1971) is a problem-solving language that embeds real-world knowledge in procedures. It was used in robotic control and as a semantic base for English. MYCIN (Shortliffe, 1974) is a rule-based problem-solving system to assist physicians with an appropriate therapy for bacterial infections. XCON, a production-rule-based system, is used to validate the technical correctness (configurability) of customer orders and to guide the actual assembly of these orders. It has about 80K rules and achieved $95{\sim}98\%$ accuracy. It saved $\$25$M/year by reducing errors in orders, and gained increasing customer satisfaction (Kraft, 1984).

RL + RBC The combination of RL and RBC has been explored in many studies, where RBCs are primarily used as auxiliary constraints or guiding mechanism. Lee et al. (2020) propose to use two modules in their control flow, one for continuous control with RL agent and a discrete one controlled by RBC. Wang et al. (2019) improve RL with low level rule-based trajectory modification to achieve a safe and efficient lane-change behavior. Zhu et al. (2021) incorporate RBC for generating the closed-loop trajectory and reducing the exploration space for RL pre-processing. Berenji (1992) use a learning process to fine-tune the performance of a rule-based controller. Radaideh & Shirvan (2021) first train RL proximal policy optimization (PPO) (Schulman et al., 2017) agents to master matching some of the problem rules and constraints, then RL is used to inject experiences to guide various evolutionary/stochastic algorithms. Likmeta et al. (2020) learn RBC parameters via RL methods. These previous methods incorporate RBC in the flow as heuristic or as hard constraints. Instead, we directly incorporate RBC policy in RL training in an algorithmic way.

Online RL regularization The online baseline we will compare to in evaluation is a state-of-the-art algorithm: TD3. It applies target policy smoothing regularization to avoid overfitting in the value estimate with deterministic policies. TRPO (Schulman et al., 2015) uses a trust region constraint based on KL-divergence between old and new policy distributions for robust policy updates. SAC (Haarnoja et al., 2018) uses soft policy iteration for learning optimal maximum entropy policies. Munchausen-RL (Vieillard et al., 2020) regularizes policy updates with a KL-divergence penalty similar to TRPO, and adds a scaled entropy term to penalize policy that is far from uniform policy. Our method differs from these methods in that we incorporate an existing real-world policy for online RL training.

Offline RL regularization Offline RL is more conservative compared with online methods as it does not require interaction with the environment. It suffers from extrapolation errors induced by selecting out-of-distribution actions. Since offline RL policies are learnt entirely from a static dataset, it is unlikely for value networks to accurately estimate the values in regime where there is no sufficient state-action visitation. Thus, regularization methods become more prominent in offline settings. Batch-constrained deep Q-learning (BCQ) (Fujimoto et al., 2019), one of the pioneers of offline RL, ascribes extrapolation errors to three main factors: absent data, model bias, and training mismatch. It mitigates the errors by deploying a variational autoencoder (VAE) to reconstruct the action given a state using the data collected by the behavioral policy. The offline baseline method we will compare to in our study is TD3+BC. It starts from online method TD3, adds a behavior cloning term

in the policy update to regularize the actor to imitate the behavioral policy and avoid selecting out-of-distribution actions. BRAC (Wu et al., 2019) studies both value penalty and policy regularization with multiple divergence metrics (KL, maximum mean discrepancy (MMD), and Wasserstein) to regularize the actor's policy based on the behavioral policy. FisherBRC (Kostrikov et al., 2021) incorporates a gradient penalty regularizer for the state-action value offset term and demonstrates the equivalence to Fisher divergence regularization. CQL (Kumar et al., 2020) learns a conservative, lower-bound estimate in value network via regularizing Q-values. Model-based method, e.g. COMBO (Yu et al., 2021) regularizes the value function on out-of-support transitions generated via environment dynamic models' rollouts.

All of these prior works use data collected by a behavioral policy, and do not assume access to any existing policy. The behavioral policy used in experiments is typically a random agent or an RL agent trained partially (medium agent) or to convergence (expert agent). In contrast, we assume direct access to a robust behavioral policy in the form of rule-based control. While this assumption may not hold for robotic applications typically used to evaluate offline RL algorithms, rule-based control policies are routinely deployed in industrial control settings, such as building HVAC control.

The regularization methods in both online and offline settings regularize with ensemble models, divergence between partially trained policies, or with logged data. These algorithms still fall into the deadly triad of function approximation, bootstrapping, and off-policy training (Sutton & Barto, 2018). In our case, we incorporate a robust reference policy to improve RL policy performance. The behavioral policy is selectively trained with actor only when the averaged Q-value estimate is higher than the rule-based control policy actions in every mini-batch. The rule-based control policy reduces uncertainty due to its deterministic behavior. On the opposite, deep learning model is affected by random initialization conditions, even if trained on the same dataset, as varied initialization conditions might lead to different policies.

3 BACKGROUND

In the reinforcement learning paradigm, an agent acts in an environment, sequentially selects actions based on its policy at every time step. The problem can be formulated as a Markov Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $\mathcal{R}$, transition dynamics p, and discount factor $\gamma \in [0, 1)$. The goal is to maximize the expectation of the cumulative discounted rewards, denoted by $R_t = \sum_{i=t+1}^{\infty} \gamma^i r(s_i, a_i, s_{i+1})$(Sutton & Barto (2018)). The agent's behavior is determined by a policy $\pi : \mathcal{S} \to \mathcal{A}$, which maps states to actions either in deterministic way or with a probability distribution. The expected return following the policy from a given state s is the action-value function $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | s_0 = s, a_0 = a]$ by taking action a.

We use the building RL environment from Jiménez-Raboso et al. (2021). The objective of the agent is to maintain a comfortable thermal environment with minimum energy use. The state consists of indoor/outdoor temperatures, time/day, occupant count, thermal comfort, and related sensor data. The action adjusts the temperature setting of the thermostat. The reward is a linear combination of occupants' thermal comfort and energy consumption. The environment is a single floor building divided in 5 zones, with 1 interior and 4 exterior rooms. See Appendix B for details about our building RL settings.

4 RULE-BASED INCORPORATED CONTROL REGULARIZATION

Our goal is to improve an agent's ability to learn with the assistance of human expert's domain knowledge in both online and offline settings. In real-world problems, sometimes we have existing simulators as oracles so we can safely learn with online RL methods before deploying in the real environment, for example, in robot control (Todorov et al., 2012), Go (Silver et al., 2017), and video games (Mnih et al., 2013). However, in some real-world problems, it is time-consuming and requires domain expertise to build a functional simulation environment (e.g. building thermal simulations), or it can be dangerous or risky to evaluate partially trained policy (e.g. healthcare and financial trading). Offline RL algorithms, on the other hand, rely on historical data collected by an existing but unknown behavioral policy. The objective is to learn a policy that improves on the behavioral

policy measured through episodic rewards. In Fig. 1, we illustrate the process of RL training that accommodates both the online and offline paradigms.

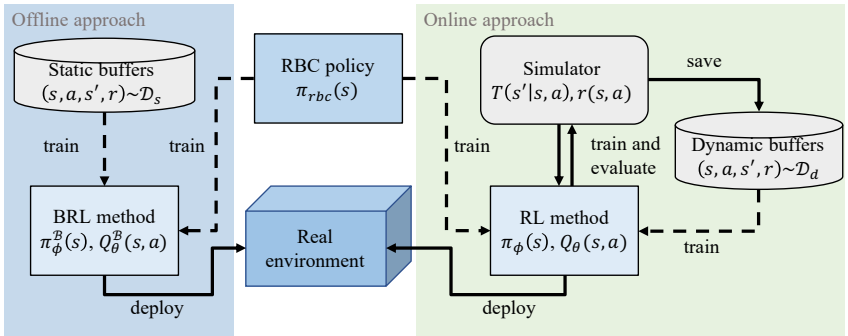

Figure 1: Our proposed method, RUBICON, incorporates RBC into RL to improve stability. Dotted lines indicate the dynamically weighted policies, and solid lines symbolize fixed actions.

Our algorithm builds on existing actor-critic algorithms TD3 and TD3+BC. We only modify the policy update strategy with the rule-based control policy, and use the critic as-is. Therefore, we focus our discussion on the policy update of the algorithm. TD3 starts from DDPG (Silver et al., 2014) and mitigates the function approximation error with double Q-learning and delayed policy updates. TD3+BC is an offline RL algorithm adapted from TD3, and is one of the state-of-the-art offline RL methods evaluated with D4RL datasets (Fu et al., 2020). TD3+BC adds a behavior cloning term to the policy update step to penalize the policy that is far away from behavioral policy (Eq. 1). The blue-colored terms indicate the changes from TD3 to TD3+BC.

$$\pi = \arg\max_{\pi} \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\lambda Q(s,\pi(s)) - (\pi(s)-a)^2\right] \tag{1}$$

$$\lambda = \frac{\alpha}{\frac{1}{N}\sum_{(s_i,a_i)} |Q(s_i,a_i)|} \tag{2}$$

In Eq. 1, λ is decided by the averaged mini-batch Q-estimate and a hyperaparameter α to adjust between RL and imitation learning (Eq. 2).

Our method, RUBICON, dynamically weighs both TD3 and TD3+BC's policy update steps with either RBC policy or behavioral policy in each training iteration. In Eq. 3, we replace the actions a sampled from the buffers in Eq. 1 with $\pi_{Q_{max}}(s)$ and add a hyperparameter ξ to integrate TD3 and TD3+BC methods as one. Red-colored terms indicate the changes from TD3 and TD3+BC to our method. We replace the notation of sampled actions a in TD3+BC with behavioral policy $\pi_b(s)$ to avoid confusion. Details of the hyperparameter settings in our work are in Appendix D.

$$\pi = \arg\max_{\pi} \mathbb{E}_{s\sim\mathcal{D}} \left[\lambda Q(s,\pi(s)) - \xi(\pi(s) - \pi_{Q_{max}}(s))^2\right] \tag{3}$$

$$\pi_{Q_{max}}(s) = \arg\max_{\pi} \{\bar{Q}(s,\pi_b(s)), \bar{Q}(s,\pi_{rbc}(s))\} \tag{4}$$

Every time when the policy is being updated, given the states s of the sampled mini-batch, the behavioral policy $\pi_b(s)$ and the RBC policy $\pi_{rbc}(s)$ select actions in a deterministic fashion. The state-action pairs' Q-values are estimated by the critic, the average of the Q-value estimations in the mini-batches are $\bar{Q}(s,\pi_b(s))$ and $\bar{Q}(s,\pi_{rbc}(s))$. We dynamically select the transitions with a higher average Q-value to be regularized from in each policy update step, i.e. the actor loss function is dynamically weighted. In online settings, the behavioral policies are the older versions of the policy we used to generate the buffer.

The reason we choose the average as the metric to decide which set of transitions to learn from instead of selecting each transition with higher estimated value (each batch is a combination of $\pi_b(s)$

and $\pi_{rbc}(s)$) is that if we choose by each transition we will lose the information on which state-action visitations lead to worse values, the model will then suffer from the imbalanced data problem. The credit/blame assignment is essential in RL learning convergence and the experience replay can help speed up the propagation process (Lin, 1992). The algorithm of our method is given in Alg. 1. Changes from baselines to our method are highlighted in blue. d is the policy update frequency, the noise ϵ added to the policy is sampled with Gaussian $\mathcal{N}(0, \sigma)$ and clipped by c. In both online and offline approaches, the policy update follows Eq. 3 and 4 with different hyperparameter settings.

Our rule-based control algorithm is described in Alg. 2. It is derived from the rule-based controller in Sinergym's (Jiménez-Raboso et al., 2021) example. For the purpose of computation efficiency and to fit the batch settings in our algorithm, we vectorize the original RBC policy. The rules are simple and intuitive, and can generalize well: First, we get the datetime information we need from the states. Then, we can get the seasonal comfort temperature zone for every transition. If the indoor air temperature (IAT) is below the lower bound of the comfort zone, then we set both cooling and heating setpoints a degree higher (measured in Celcius degree). On the opposite, if the IAT is above the upper bound of the comfort zone, then we set both the heating and cooling setpoints a degree lower than the current setpoints. Finally, we examine if the current datetime is in the office hours. If not, then the setpoints are set to be $(18.33, 23.33)$ (°C) for the purpose of energy reduction since occupants' thermal comfort is not important in these time periods assuming zero occupancy.

Algorithm 1: RUBICON

Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$,actor network π_ϕ, and RBC policy π_{rbc} with random parameters θ_1, θ_2, ϕ, target networks $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$ and replay buffer or load replay buffer $\mathcal{B}$
for $t = 1$ *to* T **do**
  if *online* **then**
    Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$
    Observe reward r and next state s'
    Store transition (s, a, r, s') in $\mathcal{B}$
  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), \text{-c,c})$
  $y \leftarrow r + \gamma \min_{j=1,2} Q_{\theta'_j}(s', \tilde{a})$
  Update critics $\theta_j \leftarrow \arg\min_{\theta_j} N^{-1}(y - Q_{\theta_j}(s, a))^2$
  if t *mod* d **then**
    Update ϕ by policy gradient:
    Policy update follows Eq. 3 and 4
    Calculate $\nabla_\phi J(\phi)$
    Update target networks:
    $\theta'_j \leftarrow \tau\theta_j + (1-\tau)\theta'_j$
    $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$

Algorithm 2: Rule-based control policy

Input : Current daytime $current_dt$, indoor air temperature IAT, zone thermostat heating setpoint temperature a_h, and zone thermostat cooling setpoint temperature a_c, all obtained from the states in the sampled mini-batch with size N
Output: Actions selected by RBC
for i *in* N **do**
  $season_comfort_zone_i = get_season_comfort(current_dt_i)$
  if $IAT_i >= min(season_comfort_zone_i)$ **then**
    $a_{h_i} = a_{h_i} - 1$
    $a_{c_i} = a_{c_i} - 1$
  else if $IAT_i < max(season_comfort_zone_i)$ **then**
    $a_{h_i} = a_{h_i} + 1$
    $a_{c_i} = a_{c_i} + 1$
  $a_i = (a_{hi}, a_{ci})$
  if $current_dt_i.weekday \geq 5$ *or* $current_dt_i.hour$ *in range(22,6)* **then**
    $a_i = (18.33, 23.33)$(°C)

5 EXPERIMENTS

In our experiments, there are two environment types: deterministic and stochastic. A Gaussian noise with μ=0 and σ=2.5 is added to the outside temperature from episode to episode in the stochastic environments. And three weather types: hot, cool, and mixed. In the results, "*hot-deterministic*" indicates that the task is learnt and evaluated with hot weather condition and deterministic environment. Similarly, we have all six combinations such as "*cool-stochastic*" and so on. More details about the RL setup is given in Appendix B. All scores in tables and figures in this paper are normalized with expert policy as 100 and random policy as 0.

5.1 OFFLINE APPROACH

First, we consider the offline approach, where no simulator exists but historical data is available. We follow the standard procedure for BRL evaluation (Fu et al., 2020): (1.) Train behavioral agents for

500K time steps, then compare the most representative algorithms DDPG, TD3, and SAC (learning curves are shown in Appendix C). The online methods we compare are described below:

- **DDPG**: Deep deterministic policy gradient is a method that combines the actor-critic approach and deep Q-network (DQN) (Mnih et al., 2013). It is capable of dealing with continuous action spaces problems via policy gradient in a deterministic approach which outperforms stochastic policy method in high dimensional tasks.
- **SAC**: Soft actor-critic, an off-policy maximum entropy RL algorithm to encourage exploration. They empirically show that SAC yields a better sample efficiency than DDPG.
- **TD3**: Twin delayed deep deterministic policy gradient algorithm, it reduces overestimation with double Q-learning, combines with target networks to limit errors from imprecise function approximation.

(2.) Select the best agent as our expert agent and generate buffers with it for 500K time steps. A medium agent is trained "halfway", it means that an agent is trained most closed to an agent with the evaluation performance half the performance as the expert agent. And a random agent which samples actions randomly and generate buffers.

(3.) Train BRL methods for 500K time steps and evaluate the policy every 25K time steps in all buffers mentioned above in step (2.). We show the detailed learning curves in Appendix C. Normalized and averaged scores across runs are shown in Table 1. The offline methods we compare with are listed below:

- **TD3+BC**: An offline version of TD3, it adds a behavior cloning term to regularize policy towards behavioral policy combined with mini-batch Q-values and buffer states normalization for stability improvement.
- **CQL**: Conservative Q-learning, derived from SAC, it learns a lower-bound estimate of the value function by regularizing the Q-values during training.
- **BCQ**: Batch-constrained deep Q-learning, it implements a variational autoencoder (VAE) (Kingma & Welling, 2013) to reconstruct the action given the state. And adds perturbation in actor on the policy, the degree of perturbation and size of mini-batch can be adjusted in order to behave more like a traditional RL method or imitation learning.
- **BC**: Behavior cloning, we train a VAE to reconstruct action given state. It simply imitates the behavioral agent without reward signals.

In Table 1, we observe that RUBICON outperforms all other benchmarks in overall score across weather types, random seeds, and environment types. Other BRL methods show good performance either in specific tasks or with a specific randomly initialized configuration; however, overall they are more unstable cf. RUBICON. Our method provides more robust and consistent performance across all variants and demonstrates the ability to generalize across various weather types and response modes of tasks. Also, as we can see in Fig. 2, learning from both medium buffer and RBC policy, RUBICON improves on their best performance. Our method is stable since the standard deviation is the least among the policies trained. We include the BRL learning curves with expert and random buffers in Appendix C

5.1.1 Data efficiency experiment

We conduct the experiments with buffers of only one year of data ($35,040$ transitions). Data efficiency is a challenge for RL to yield accurate value estimation. In Table 2, we observe that our method still outperforms its baseline overall. Although it dominates with random buffers and has comparable performance with expert buffers, it does not learn well with medium buffers. The root cause is the similarity of the quality of actions between medium buffers and RBC policy, which causes the critic to misjudge which action to pick between them. However, RUBICON still outperforms the baseline in other two types of buffers since the value estimation differences between $(\pi_b(s), s)$ and $(\pi_{rbc}(s), s)$ are more prominent in these scenarios.

Environment	Buffer	RUBICON	TD3+BC	CQL	BCQ	BC
hot-deterministic	Expert	86.14±15.28	**99.72±0.1**	**100.00±0.00**	-32.02±0.07	-89.2±3.95
hot-deterministic	Medium	**64.91±6.46**	-49.59±8.19	**67.65±17.06**	13.41±16.59	-12.55±7.27
hot-deterministic	Random	**62.71±5.12**	-45.73±15.13	-23.19±4.52	**69.21±18.52**	-26.74±15.91
mixed-deterministic	Expert	81.00±22.35	**94.67±2.04**	**100.00±0.00**	-6.22±5.24	-95.46±6.6
mixed-deterministic	Medium	**86.84±2.64**	36.23±4.31	37.36±19.31	64.46±0.65	-103.4±2.12
mixed-deterministic	Random	**68.84±1.32**	-13.72±22.25	-23.46±20.33	-65.30±20.40	-27.82±11.79
cool-deterministic	Expert	**98.01±1.23**	81.11±5.24	**100.00±0.00**	-29.75±3.18	27.76±0
cool-deterministic	Medium	**72.21±8.19**	-49.97±0.00	55.44±6.46	**70.19±17.06**	10.48±22.11
cool-deterministic	Random	**66.51±0.00**	-58.40±3.21	12.99±2.28	27.77±31.39	8.62±41.97
hot-stochastic	Expert	**99.02±0.51**	77.69±17.18	**99.49±0.20**	-15.35±5.92	-72.86±1.73
hot-stochastic	Medium	**59.72±4.31**	-14.85±0.00	39.93±2.64	-62.21±19.31	-10.45±12.85
hot-stochastic	Random	**68.84±0.83**	-1.82±2.68	36.65±11.95	-1.24±14.80	31.22±13.51
mixed-stochastic	Expert	94.16±2.21	**96.61±2.13**	**99.77±0.26**	-108.38±2.58	-102.02±9.32
mixed-stochastic	Medium	**87.23±0.00**	9.49±0.00	**80.13±8.19**	70.75±6.46	-107.41±3.41
mixed-stochastic	Random	67.03±1.63	28.02±8.69	**94.05±2.08**	-109.47±0.17	38.66±24.64
cool-stochastic	Expert	53.59±2.65	78.27±20.01	**99.97±0.12**	-115.86±0.41	28.15±0.35
cool-stochastic	Medium	68.07±0.00	16.09±0.00	**81.57±4.31**	-11.55±2.64	-50.37±2.45
cool-stochastic	Random	**67.56±1.38**	-44.33±16.01	-97.35±2.09	-53.92±10.07	25.44±13.42
Sum		**1352.40±76.11**	339.50±127.23	960.99±101.81	-295.49±175.47	-527.93±193.48

Table 1: BRL Methods Benchmark: Average normalized score over the final 5 evaluations and 3 random seeds. ± corresponds to standard deviation over the last 5 evaluations across runs.

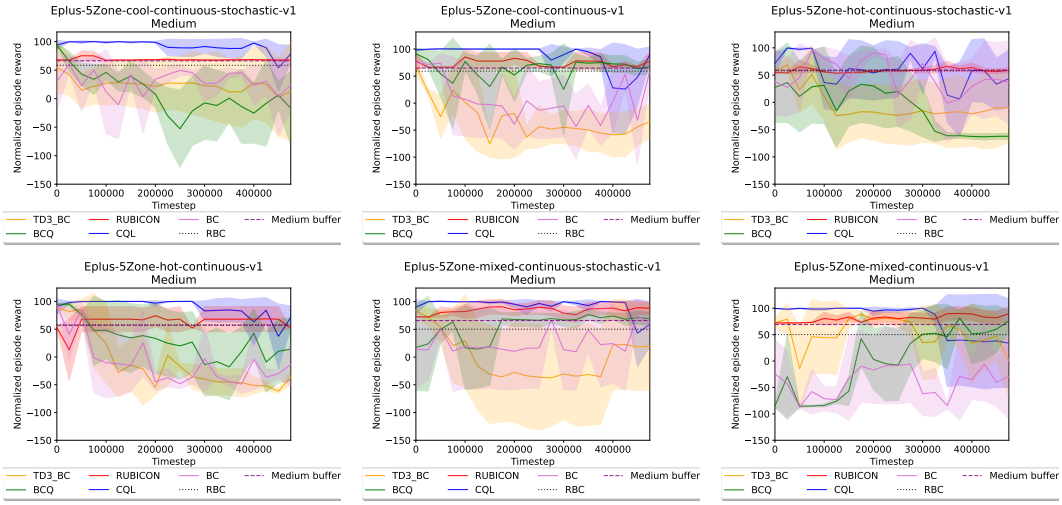

Figure 2: Learning curves of BRL models trained using medium buffers

5.1.2 Ablation Experiment

In this experiment, we remove the dynamically weighted regularization. Instead we regularize the behavioral policy and RBC policy simultaneously in every iteration of training (see Eq. 5). The experimental results are shown in Table 3. We observe regularizing both at the same time deteriorates policy training. Since in each iteration, one of RBC policy $\pi_{rbc}(s)$ and behavioral policy $\pi_b(s)$ yields a better action selection compared to the other. It emphasizes the necessity of dynamic weighting in the policy update steps.

$$\pi = \arg\max_{\pi} \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\lambda Q(s, \pi(s)) - (\pi(s) - a)^2 - (\pi(s) - \pi_{rbc}(s))^2\right] \tag{5}$$

5.1.3 Transfer Experiment

We consider a realistic scenario where in one weather condition we already have an existing buffer then we want to use it as prior knowledge to combine with RBC and transfer our model to another weather type where we have no data. We experiment with the medium buffers in stochastic environments. The results shown in Table 4 indicate that our method is capable of transferring from one weather condition to another with comparable performance without any hyperparameter changes. As the results indicate, due to the diversity of the mixed weather, it improves the learning in cool

Environment	Buffer	RUBICON	TD3+BC
hot-deterministic	Expert	**89.65±6.78**	74.96±0.98
hot-deterministic	Medium	-6.92±17.3	**79.37±1.53**
hot-deterministic	Random	**90.41±2.79**	-24.18±16.44
mixed-deterministic	Expert	43.09±5.2	**93.77±0.54**
mixed-deterministic	Medium	48.24±4.74	**98.67±0.32**
mixed-deterministic	Random	**82.89±3.77**	72.58±2.04
cool-deterministic	Expert	**89.52±3.11**	86.45±1.98
cool-deterministic	Medium	-11.56±2.92	**47.61±6.63**
cool-deterministic	Random	35.92±24.25	**42.59±3.97**
hot-stochastic	Expert	80.35±16.9	**89.02±1.29**
hot-stochastic	Medium	5.3±5.56	**10.79±15.25**
hot-stochastic	Random	**62.26±10.88**	26.43±1.19
mixed-stochastic	Expert	78.08±19.65	**85.25±0.89**
mixed-stochastic	Medium	66.02±1.14	**72.17±3.48**
mixed-stochastic	Random	**55.26±24.49**	-86.21±1.62
cool-stochastic	Expert	**98.18±2.22**	39.04±0.76
cool-stochastic	Medium	**73.16±1.31**	38.68±4.83
cool-stochastic	Random	**69.76±1.8**	-49.5±2.42
Sum		**1049.61±154.81**	797.49±66.16

Table 2: Data reduction experiment

Environment	Buffer	RUBICON	RUBICON w/o DW
hot-deterministic	Expert	**86.13±15.27**	-19.8±17.31
hot-deterministic	Medium	**64.91±6.46**	47.26±13.66
hot-deterministic	Random	**62.7±5.11**	-19.8±17.31
mixed-deterministic	Expert	**81±22.35**	-75.6±8.27
mixed-deterministic	Medium	**86.84±3.5**	42.99±29.04
mixed-deterministic	Random	**68.83±1.32**	-75.6±8.27
cool-deterministic	Expert	**98±1.22**	41.3±16.13
cool-deterministic	Medium	**72.2±4.67**	36.54±15.1
cool-deterministic	Random	**66.5±0**	41.3±16.13
hot-stochastic	Expert	**99.01±0.51**	57.68±1.67
hot-stochastic	Medium	**59.72±2.64**	29.26±24.63
hot-stochastic	Random	**68.83±0.82**	57.68±1.67
mixed-stochastic	Expert	**94.16±2.2**	40.57±8.49
mixed-stochastic	Medium	**87.23±2**	55.6±22.71
mixed-stochastic	Random	**67.03±1.63**	40.57±8.49
cool-stochastic	Expert	**53.58±2.65**	-68.84±0.98
cool-stochastic	Medium	**68.07±0.25**	8±15.23
cool-stochastic	Random	**67.55±1.37**	-68.84±0.98
Sum		**1352.37±74.06**	170.3±226.13

Table 3: Ablation experiment

Environment	Trans. from	RUBICON_Trans.	RUBICON
hot-stochastic	cool-stochastic	**71.42±1.64**	59.72±4.3
mixed-stochastic	cool-stochastic	84.93±4.92	**87.23±0**
cool-stochastic	hot-stochastic	54.07±0.46	**68.07±0**
mixed-stochastic	hot-stochastic	81.02±1.8	**87.23±0**
cool-stochastic	mixed-stochastic	**72.39±0.37**	68.07±0
hot-stochastic	mixed-stochastic	**75.26±1.16**	59.72±4.3
Sum		439.09±10.35	430.04±8.6

Table 4: Transfer experiment

and hot weathers. On the other hand, transfer from monotonic weather condition leads to worse returns.

5.1.4 REWARD ANALYSIS EXPERIMENT

Since Q-value estimation is usually overestimated, we want to use immediate reward as a reference to examine the quality of the predictions of Q-functions. We pre-train a reward model $R_\psi(s, a)$ to predict reward $\hat{r}$ given state s and selected action a with 200K iterations with the buffer as our training data. At each iteration of policy update, we record the policy $\pi(s)$ and the predict rewards in each batch $\hat{r} = R_\psi(s, \pi(s))$. We plot the distributions of reward in action spaces in Fig. 3. It demonstrates that RUBICON selects the actions in a wider range cf. TD3+BC, however, with a reward distribution of higher values. The distribution shown is with 10% of data randomly selected from the entire training for a better visualization.

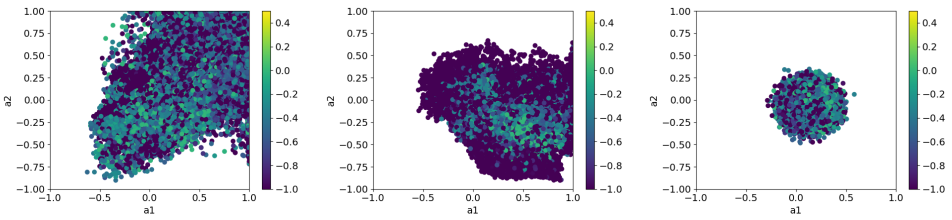

Figure 3: Reward distribution in action spaces of *hot-continuous* environment learns from medium buffer, from left to right: RUBICON, TD3+BC, and buffer.

5.2 ONLINE APPROACH

In online approach, we assume an oracle exists for accurate simulation. In real-world applications, researchers train online model in simulation before deployment in real building environments. We compare with TD3, the baseline we develop our method on. Experimental results comparing TD3 and our method can be found in Table 5. In five out of six tasks, our method outperforms TD3

in averaged scores and with a substantially smaller standard deviation across runs. The detailed learning curves are illustrated in Fig. 4.

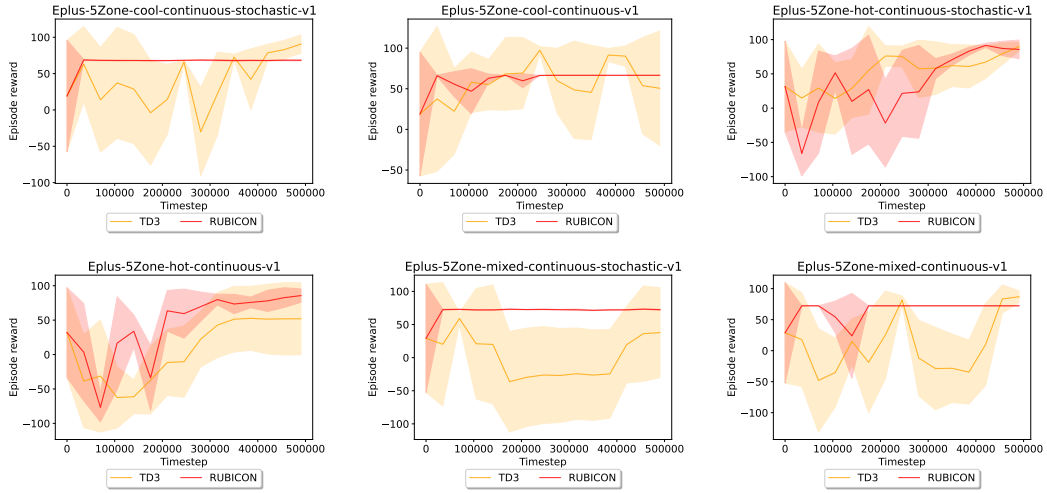

Figure 4: Learning curves comparing RUBICON and TD3, solid line indicates averaged values and half-transparent region is the range of one standard deviation.

Environment	RUBICON	TD3
hot-deterministic	**79.09±4.48**	51.84±0.45
mixed-deterministic	**72.35±0.00**	23.46±52.64
cool-deterministic	**66.52±0.00**	**66.31±20.19**
hot-stochastic	**83.65±6.99**	71.80±11.09
mixed-stochastic	**72.24±0.57**	8.50±28.47
cool-stochastic	68.15±0.18	**73.39±16.69**
Sum	**442.00±12.22**	295.29±129.53

Table 5: RUBICON and TD3 comparison

Environment	$\alpha = 1$	$\alpha = 2.5$	$\alpha = 4$
hot-deterministic	**79.09±4.48**	70.76±6.22	23.83±14.32
mixed-deterministic	**72.35±0.00**	71.92±0.36	**72.26±0.04**
cool-deterministic	**66.52±0.00**	53.28±12.03	**68.16±5.27**
hot-stochastic	**83.65±6.99**	73.92±5.97	65.13±26.54
mixed-stochastic	**72.24±0.57**	**72.24±0.56**	**72.24±0.56**
cool-stochastic	**68.15±0.18**	45.46±23.28	12.38±19.43
Sum	**442.00±12.22**	387.58±48.42	314±66.16

Table 6: Hyperparameter experiment.

5.2.1 Hyperparameter experiment

Since we integrate TD3 and TD3+BC as one, we need to introduce the weighting λ and its hyperparameter α (see Eq. 2 and Alg. 1) in the online settings. It is mentioned in TD3+BC paper that the value of α decides if the model learns similar to RL (α=4) or imitation learning (α=1) and the default value set in TD3+BC is α =2.5. We experiment on the values $\{1, 2.5, 4\}$ to observe how it affects the performance of our models in all tasks. The result (See Table 6) shows that when α=1 the model gives the highest scores and the least variance. Which indicates that the actor policy should imitate RBC policy more in order to yield a more stable and robust averaged score rather than a traditional RL configuration (α=4). In offline settings, we follow TD3+BC, use α=2.5.

6 Discussion

In this paper, we explore how rule-based control can be incorporated with online and offline reinforcement learning algorithms as policy regularization in a unified approach. Our method can be implemented on the baseline method with minimal changes. We empirically demonstrate that our method outperforms several state-of-the-art batch reinforcement learning methods and improves from its online baseline by a substantial amount in building HVAC tasks where rule-based control is robust and a standard in real-world settings. We expect our study would encourage both domain and RL experts to explore more opportunities for the combination of existing policies and RL and extend this concept to more real-world applications. Our future works aim to: (1.) Continue exploring the opportunities to integrate online and offline RL algorithms as one and (2.) Leverage RL to improve from existing best-known policies in real-world applications. (3.) Develop a scheduling methodology for the degree of the incorporation of RBC policy.

REFERENCES

Hamid R Berenji. A reinforcement learning—based architecture for fuzzy logic control. *International Journal of Approximate Reasoning*, 6(2):267–292, 1992.

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

Department of Energy. Prototype Building Models. `https://www.energycodes.gov/prototype-building-models#TMY3`.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

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

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

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

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

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*, pp. 1861–1870. PMLR, 2018.

Frederick Hayes-Roth. Rule-based systems. *Communications of the ACM*, 28(9):921–932, 1985.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Carl Hewitt. Procedural embedding of knowledge in planner. In *IJCAI*, pp. 167–182, 1971.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, and Jeff Braga. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *arXiv preprint arXiv:2111.08819*, 2021.

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*, pp. 319–323, 2021.

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

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pp. 5774–5783. PMLR, 2021.

Arnold Kraft. Xcon: An expert configuration system at digital equipment corporation. *The AI Business: The Commercial Uses of Artificial Intelligence, The MIT Press, Cambrige, Massachusetts*, 1984.

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.

Daeil Lee, Awwal Mohammed Arigi, and Jonghyun Kim. Algorithm for autonomous power-increase operation using deep reinforcement learning and a rule-based system. *IEEE Access*, 8:196727–196746, 2020.

Amarildo Likmeta, Alberto Maria Metelli, Andrea Tirinzoni, Riccardo Giol, Marcello Restelli, and Danilo Romano. Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving. *Robotics and Autonomous Systems*, 131: 103568, 2020.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321, 1992.

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)*, pp. 181–192. IEEE, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

National Renewable Energy Laboratory. TMY3 Datasets. `https://www.nrel.gov/docs/fy08osti/43156.pdf`.

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.

Majdi I Radaideh and Koroush Shirvan. Rule-based reinforcement learning methodology to inform evolutionary algorithms for constrained optimization of engineering applications. *Knowledge-Based Systems*, 217:106836, 2021.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Takuma Seno and Michita Imai. d3rlpy: An offline deep reinforcement learning library. *arXiv preprint arXiv:2111.03788*, 2021.

Edward Hance Shortliffe. Mycin: a rule-based computer program for advising physicians regarding antimicrobial therapy selection. Technical report, Stanford Univ Calif Dept of Computer Science, 1974.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. PMLR, 2014.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

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

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

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.

Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4235–4246, 2020.

Junjie Wang, Qichao Zhang, Dongbin Zhao, and Yaran Chen. Lane change decision-making through deep reinforcement learning with rule-based constraints. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6. IEEE, 2019.

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

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

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.

Yuanyang Zhu, Zhi Wang, Chunlin Chen, and Daoyi Dong. Rule-based reinforcement learning for efficient robot navigation with space reduction. *IEEE/ASME Transactions on Mechatronics*, 27 (2):846–857, 2021.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A EXPERIMENT DETAILS

Software

- **Python**: 3.9.12
- **Pytorch**: 1.12.1+cu113 (Paszke et al., 2019)
- **Sinergym**: 1.9.5 (Jiménez-Raboso et al., 2021)
- **Gym**: 0.21.0 (Brockman et al., 2016)
- **Numpy**: 1.23.1 (Van Der Walt et al., 2011)
- **CUDA**: 11.2

Hardware

- **CPU**: Intel Xeon Gold 6230 (2.10 GHz)
- **GPU**: NVidia RTX A6000

Benchmark implementations

- **DDPG**: We adopt the DDPG implementation in TD3 author-provided implementation
- **TD3**: Author-provided implementation
- **SAC**: We adopt CleanRL (Huang et al., 2021) implementation due to software version conflict with author-provided repository
- **TD3+BC**: author-provided implementation
- **CQL**: We adopt d3rlpy (Seno & Imai, 2021) implementation due to software version conflict with author-provided repository
- **BCQ**: Author-provided implementation

B BUILDING RL SETTINGS

In this section, we list all the details about the MDP settings in our problem.

- **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.
- **Reward**: We follow the default linear reward settings, it considers the energy consumption and the absolute difference to temperature comfort.
- **Environment**: A single floor building with an area of $463.6m^2$ divided in 5 zones, 1 interior and 4 exterior. 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 control variables are the cooling and heating temperature setpoints for the interior zone, and the simulation period is a full year (Jiménez-Raboso et al., 2021). The weather types are classified according to the U.S. Department of Energy (DOE) standard (Department of Energy). The weather type details and their representative geometric locations are listed below:
  - **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%

based on TMY3 datasets (National Renewable Energy Laboratory).

C LEARNING CURVES

We illustrate the learning curves of BRL methods learn from different quality of buffers for a better visualization of comparison in Fig. 5 2 and 6. The behavioral agents' learning curves are demonstrated in Fig. 8. And learning curves of hyperparameter experiments in online RUBICON are shown in Fig. 9. All learning curves are normalized with random policy as 0 and expert policy as 100, averaged with 3 random seeds and the scores shown in tables are the average and standard deviation last 5 evaluations unless mentioned otherwise.

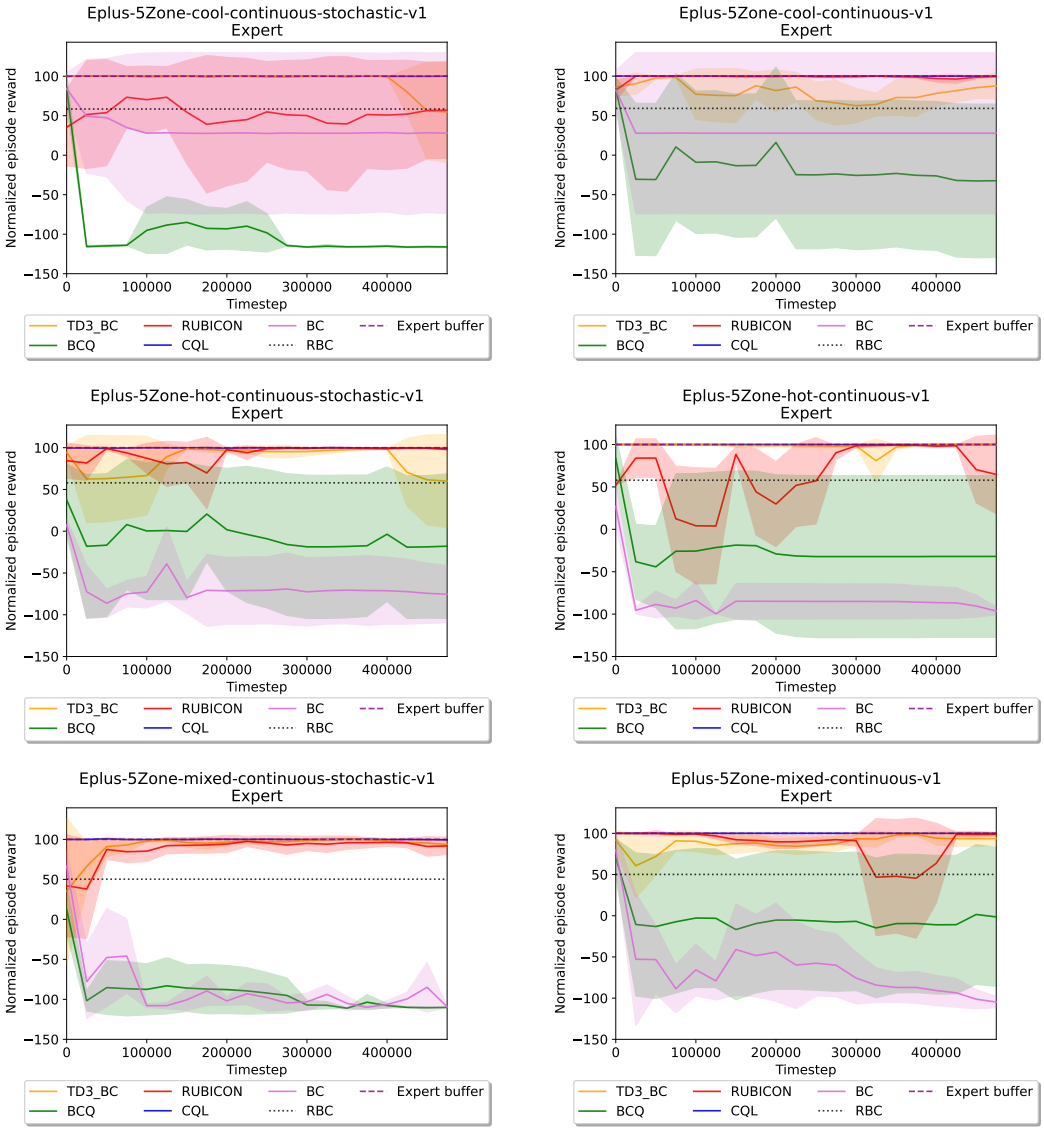

Figure 5: Learning curves of BRL models learn from expert buffers.

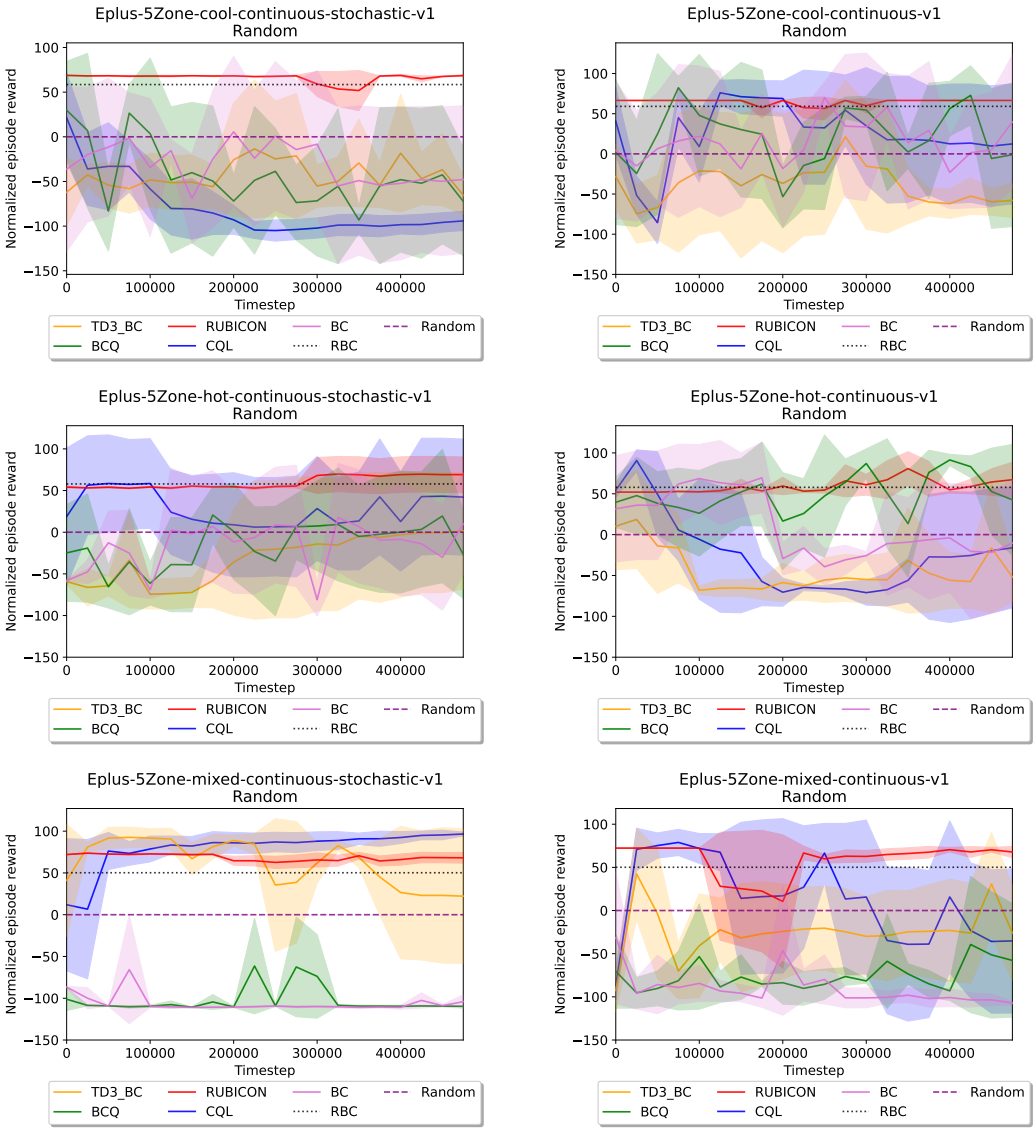

Figure 6: Learning curves of BRL models learn from random buffers

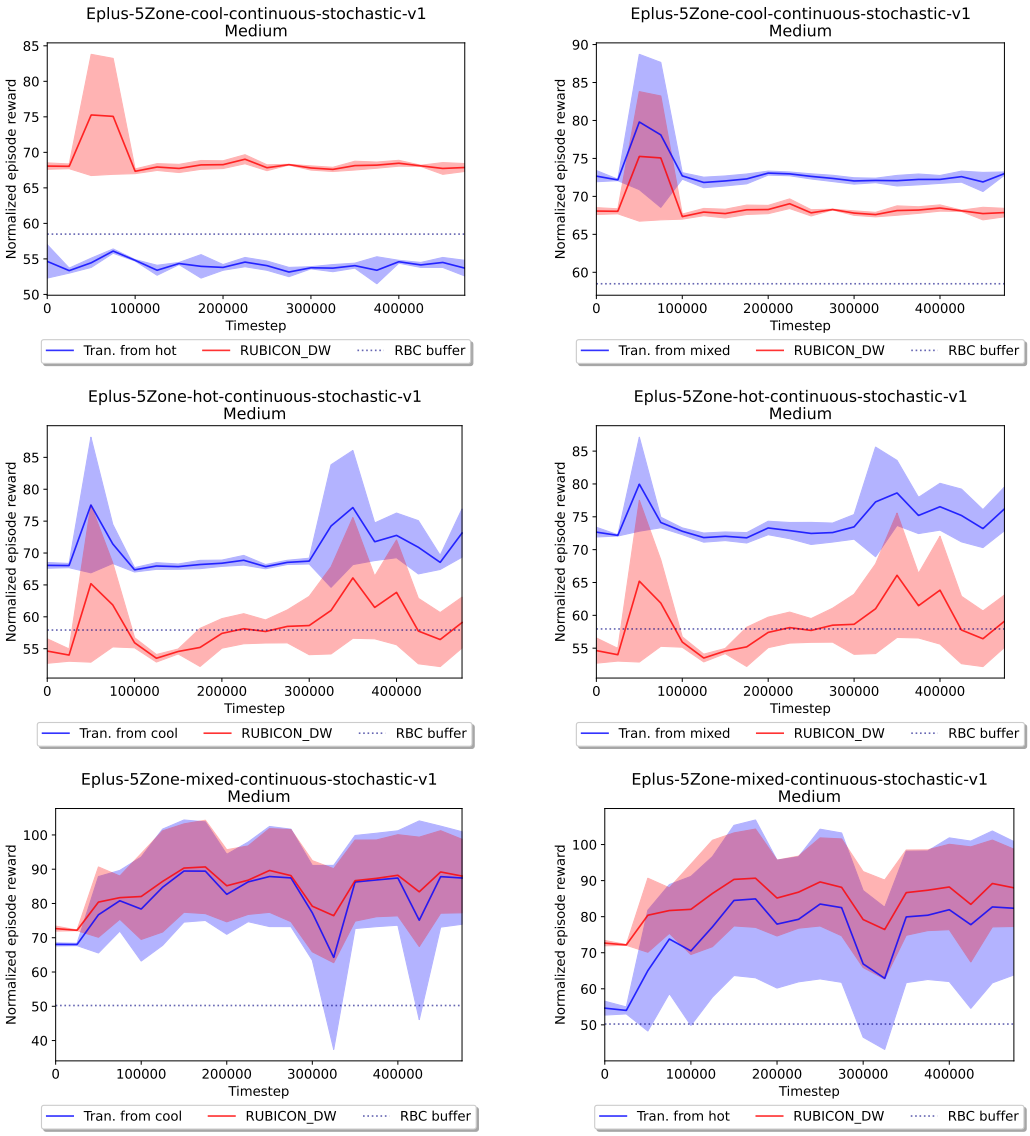

Figure 7: Learning curves of BRL models transferred from other weather types

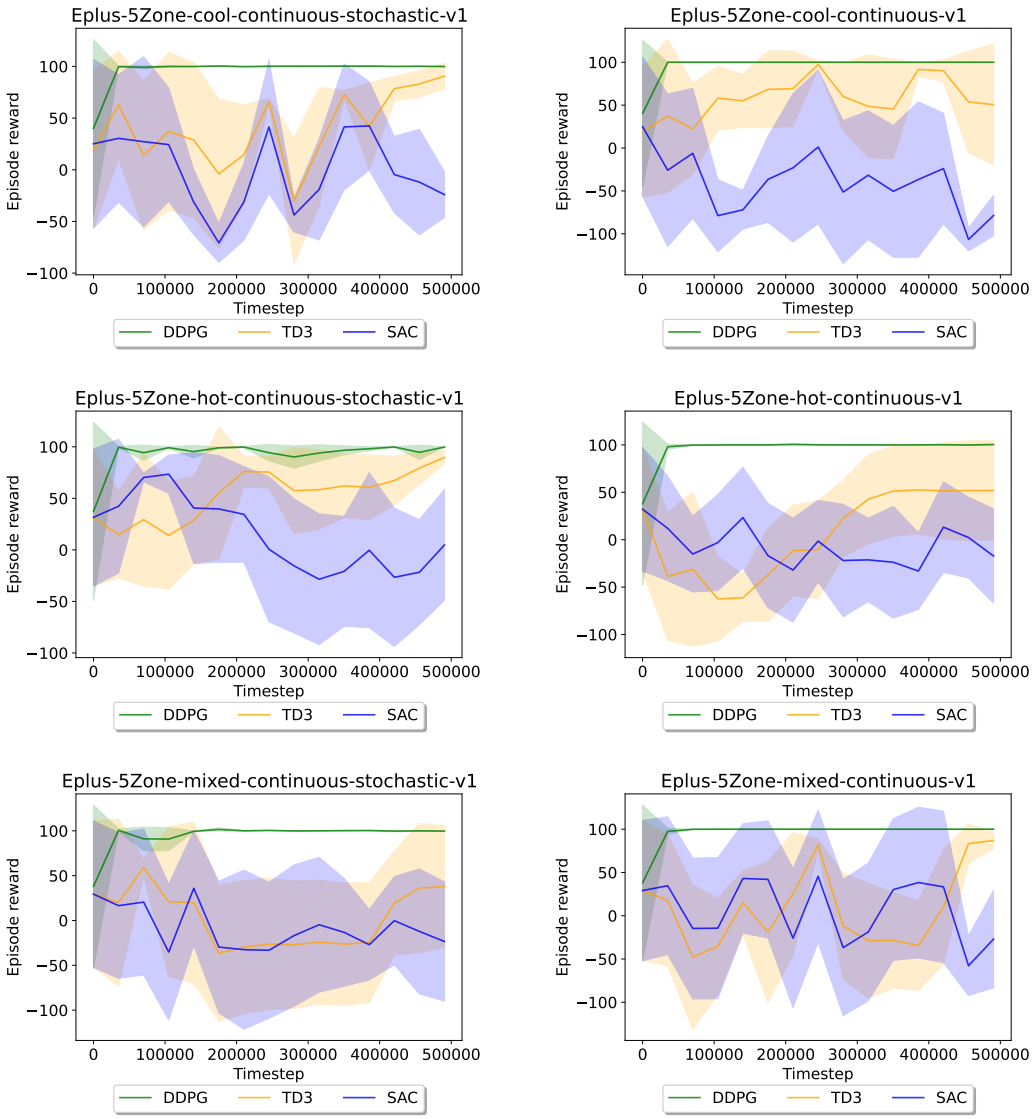

Figure 8: Learning curves of behavioral model training, behavioral models are trained with 500K time steps before generating buffers.

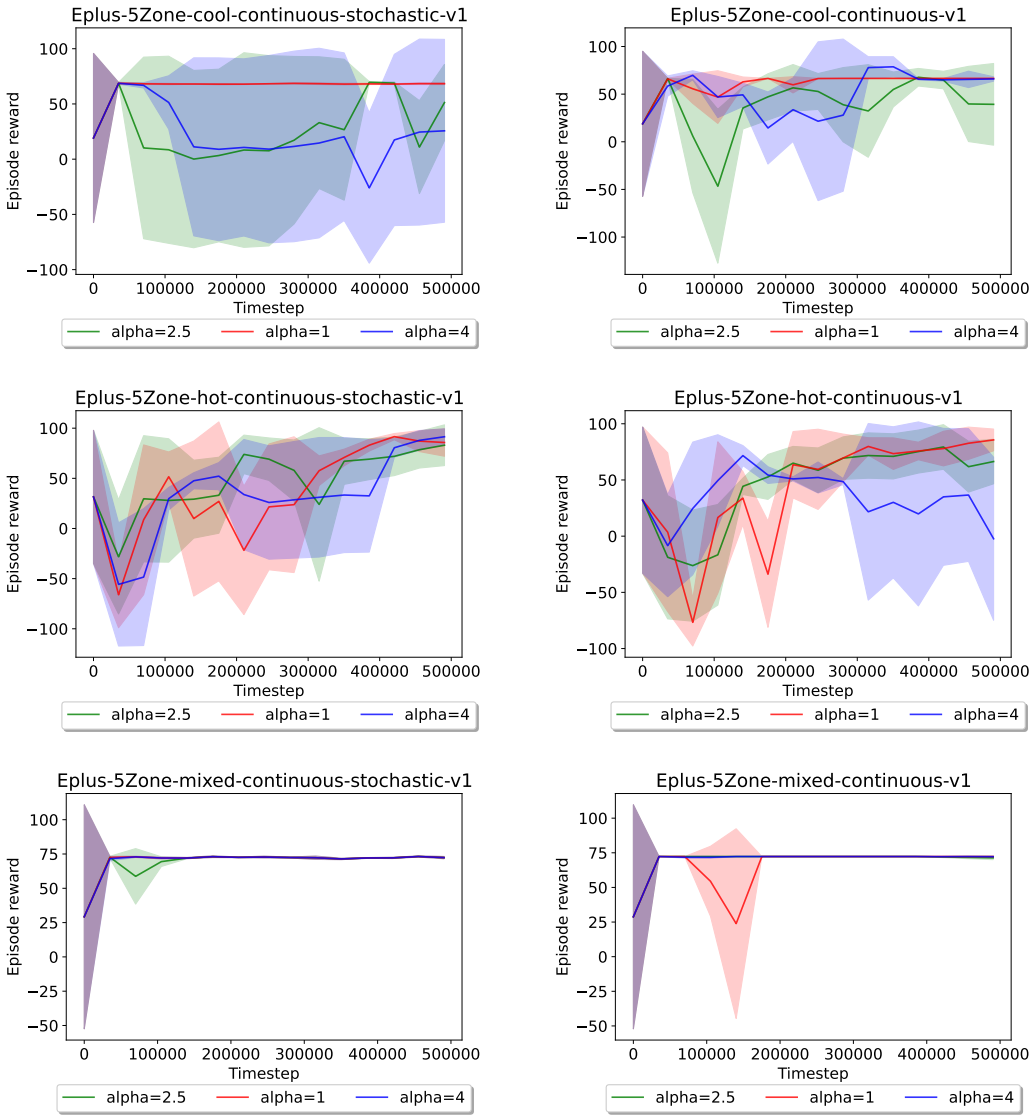

Figure 9: Learning curves of online RUBICON hyperparameter optimization.

D MODEL PARAMETERS

We list the hyperparameters used in this paper for reproducibility. Unless mentioned otherwise, we keep the original hyperparameters setups as the implementations listed in Sec. A since DRL methods are sensitive to hyperparameter tuning (Henderson et al., 2018) (see Table 7, 8, 9, and 10).

	Hyperparameter	Value
Algorithm hyperparameters	Optimizer	Adam (Kingma & Ba, 2014)
	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
	TD3+BC α	2.5
	RUBICON online α	1
	RUBICON offline α	2.5
	RUBICON online ξ	0 if $\bar{Q}_{\mathcal{B}}(s, \pi_b(s)) \geq \bar{Q}_{\mathcal{B}}(s, \pi_{rbc}(s))$ else 1
	RUBICON offline ξ	1
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

Table 7: TD3, TD3+BC, and RUBICON hyperparameters

	Hyperparameter	Value
Algorithm hyperparameters	Optimizer	Adam
	Critic learning rate	$1e^{-3}$
	Actor learning rate	$3e^{-4}$/$1e^{-4}$
	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
	SAC entropy auto-tuning	True
	CQL α threshold	10
	CQL conservative weight	5.0
	CQL number of sampled actions	10
Network architecture	Critic hidden dimension	256
	Critic hidden layers	3
	Critic activation function	ReLU
	Actor hidden dimension	256
	Actor hidden layers	3
	Actor activation function	ReLU

Table 8: SAC/CQL hyperparameters

	Hyperparameter	Value
Algorithm hyperparameters	Optimizer	Adam
	Critic learning rate	$1e^{-3}$
	Actor learning rate	$1e^{-4}$
	Mini-batch size	64
	Discount factor	0.99
	Target update rate	$1e^{-3}$
	Policy noise	$\mathcal{N}(0, 0.1)$
	Policy noise clipping	(-0.5, 0.5)
	Policy update frequency	1
Network architecture	Critic hidden dimension	400/300
	Critic hidden layers	2
	Critic activation function	ReLU
	Actor hidden dimension	400/300
	Actor hidden layers	2
	Actor activation function	ReLU

Table 9: DDPG hyperparameters

	Hyperparameter	Value
Algorithm hyperparameters	Optimizer	Adam
	Critic learning rate	$1e^{-3}$
	Actor learning rate	$1e^{-4}$
	Mini-batch size	100
	Discount factor	0.99
	Target update rate	$5e^{-3}$
	Minimum weighting	0.75
	Max perturbation	0.05
Network architecture	Critic hidden dimension	400/300
	Critic hidden layers	2
	Critic activation function	ReLU
	Actor hidden dimension	400/300
	Actor hidden layers	2
	Actor activation function	ReLU
	VAE hidden dimension	750
	VAE latent vector clipping	(-0.5, 0.5)

Table 10: BCQ/BC hyperparameters