Enhancing HVAC Control Efficiency: A Hybrid Approach Using Imitation and Reinforcement Learning

Kevlyn Kadamala(✉), Des Chambers, and Enda Barrett

University of Galway, Galway, Ireland
{k.kadamala1,des.chambers,enda.barrett}@universityofgalway.ie

Abstract. This paper explores the application of imitation learning (IL) and reinforcement learning (RL) in HVAC control. IL learns to perform tasks by imitating a demonstrator, utilising a dataset of demonstrations. However, the performance of IL is highly dependent on the quality of the expert demonstration data. On the other hand, RL can adapt control policies based on different objectives, but for larger problems, it can be sample inefficient, requiring significant time and resources for training. To overcome the limitations of both RL and IL, we propose a combined methodology where IL is used for pre-training and RL for fine-tuning. We introduce a fine-tuning methodology to HVAC control inspired by a robot navigation task. Using the 5-Zone residential building environment provided by Sinergym, we collect state-action pairs from interactions with the environment using a rule-based policy to create a dataset of expert demonstrations. Our experiments show that this combined methodology improves the efficiency and performance of the RL agent by 1% to 11.35% compared to existing literature. This study contributes to the ongoing discourse on how imitation learning can enhance the performance of reinforcement learning in building control systems.

Keywords: Imitation learning · Reinforcement learning · Continuous HVAC control

1 Introduction

Building operations account for 30% of global final energy consumption[1], with HVAC systems typically accounting for 40% of total building energy consumption[2]. Occupant comfort level satisfaction relies heavily on the effective functioning of heating, ventilation, and air conditioning (HVAC) systems. Incorrect implementation, however, may result in excessive energy consumption, escalating costs, and reduced occupant satisfaction. Traditional building controls primarily

[1] https://www.iea.org/energy-system/buildings.
[2] https://www.environment.gov.au/system/files/energy/files/hvac-factsheet-energy-breakdown.pdf.

A. Bifet et al. (Eds.): ECML PKDD 2024, LNAI 14949, pp. 256–270, 2024.
https://doi.org/10.1007/978-3-031-70378-2_16

rely on rules and heuristics derived from expert experience. These rule-based controllers (RBCs) usually rely on pre-determined set points, which may not be optimal as they are not customised to the building specifics and local weather conditions. Recently, there has been an increase in solutions that leverage smart thermostats, replacing manual control configurations. New methodologies and algorithms like model predictive control (MPC) [1] and reinforcement learning (RL) [15] can adapt their control policies based on different objectives or cost functions. However, both systems have certain drawbacks. MPC systems find it challenging to deal with non-linearity in building dynamics caused by the complex nature of building systems, long-horizon predictions for accurately forecasting building system behaviour over extended periods, uncertainties in occupancy patterns and external factors like weather conditions. On the other hand, the limitation of RL lies in its sample inefficiency, requiring a significant investment in time and training resources to reach a desirable level of performance. In this work, we intend to address the challenges RL faces in HVAC control. More specifically, we aim to see whether an imitation learning approach can enhance the training speed and overall performance of the algorithm.

Imitation learning (IL) involves learning to perform a task by observing and imitating the behaviour of a demonstrator. Instead of relying on explicit programming seen in MPC or reward signals used in RL, this method utilises a dataset of demonstrations that consist of input-output pairs representing actions taken in different states by the demonstrator. These demonstrations are usually collected from interactions between humans or expert systems with the environment. IL has, thus, seen applications in domains like autonomous vehicle driving [12], robotics [18], navigation tasks [19], etc. However, a drawback of IL is that the performance of the trained agent is highly dependent on the quality of the expert demonstration data. If the expert demonstrations are sub-optimal or incomplete, the learned policy may inherit these limitations and fail to generalise well in novel situations. Thus, we propose utilising IL and RL as pre-training and fine-tuning methodologies, respectively. Policies obtained from pre-training with IL will provide a foundation for RL fine-tuning, making optimisation easier than learning from scratch.

In this study, we ask the question - how can imitation learning help improve the performance of reinforcement learning in the application of building control systems? Imitation learning techniques, specifically behavioural cloning (BC), have been used in HVAC control before [7]. However, after pre-training with BC, we obtain a trained actor and an untrained critic for RL fine-tuning. Having such a combination interact during fine-tuning can lead to a drop in performance. Thus, we introduce a fine-tuning methodology to HVAC control that was inspired by a robot navigation task [19]. For our experiments, we use the 5-Zone residential building environment[3] provided by Sinergym [10]. To create a dataset of expert demonstrations, we utilise a rule-based policy and collect the state-action pairs from interactions with the environment. Figure 1 illustrates an overview of our training methodology. Our experiments show that the combined

[3] https://ugr-sail.github.io/sinergym/compilation/main/pages/buildings.html#zone.

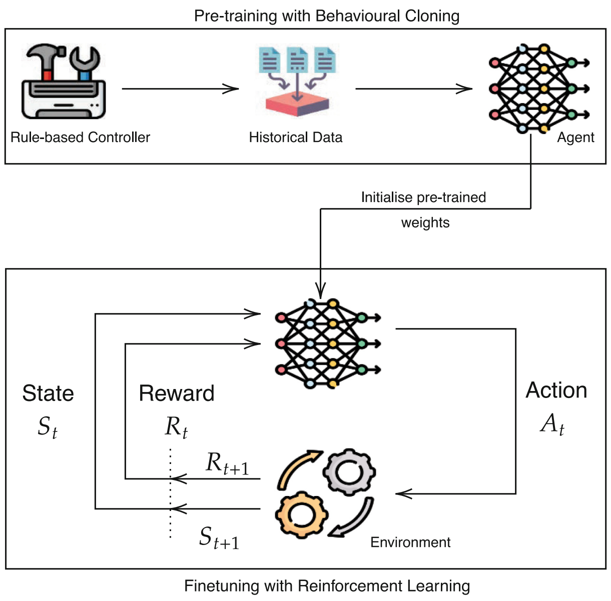

Pre-training with Behavioural Cloning
Rule-based Controller
Historical Data
Agent
Initialise pre-trained weights
State S_t
Reward R_t
R_{t+1}
S_{t+1}
Action A_t
Environment
Finetuning with Reinforcement Learning

Fig. 1. Overview of the pre-training and fine-tuning process.

methodology improves the efficiency and performance of the RL agent by 1% to 11.35% when compared to prior work [7], naive fine-tuning and training from scratch.

2 Previous Work

Reinforcement learning (RL) is gaining traction as a valuable method for Heating, Ventilation, and Air Conditioning (HVAC) systems due to its ability to learn optimal control policies for improving the management of complex and dynamic environments, such as buildings.

Previous research in HVAC systems has often utilised tabular methods in RL [2, 14, 24]. However, progress in the fields of deep learning and RL, has led to methods that blend both, using deep neural networks to improve reinforcement learning (DRL) algorithms. With the help of the Deep Q-Networks (DQN) algorithm, Wei et al. [23] were able to reduce costs by 20% to 70% when compared to scheduling methods. Similarly, when using DQN to control space heating and domestic hot water temperature, Lissa et al. [15] saw up to 16% energy savings. Arroyo et al. [1] combine model predictive control (MPC) with reinforcement learning (RL) to create RL-MPC, which aims to find the best policy while meeting all constraints. They show that RL-MPC outperforms basic RL in constraint satisfaction but can achieve similar results to pure MPC with a state estimator and optimiser. Further analysis of deep learning algorithms has been conducted by Biemann et al. [4], where they evaluated different model-free RL algorithms for continuous HVAC control. However, due to the considerable time and data required for RL to learn effective policies, researchers have incorporated transfer learning techniques to address this challenge. Transfer learning involves taking an established or trained policy from the source domain and adapting it to a target domain by leveraging past knowledge. In the context of building control systems, Lissa et al. [16] transfer HVAC agents with different spatial and geographical characteristics using tabular Q-learning, reporting that with transfer learning, temperature comfort violations were brought down to only 3% of the day, compared to 7% to 36% without transfer learning. Extending this work into a deep learning setup, Kadamala et al. [11] show that their heterogeneous transfer learning methodology adapts to buildings that differ in climate and/or characteristics, showcasing improvements from 1% to 4% compared to agents trained from scratch. Chen et al. [5] created Gnu-RL, incorporating a differentiable MPC. Initially pre-trained with imitation learning on historical data, it refines its policy using the PPO algorithm. Their research demonstrated a 6.6% energy reduction in simulations and a 16.7% decrease in cooling demand in a real-world conference room over three weeks, surpassing the existing controller while effectively managing temperature settings. Liu et al. [17] integrate RL with a rule-based control policy by adding a behavioural cloning loss (Eq. 3.1) to the policy update step to penalise the policy that differs too much from the behavioural policy. The proposed approach demonstrates significant performance improvements in building HVAC control tasks, notably where rule-based control methods are prevalent and robust. Coraci et al. [6] performed online transfer learning (OTL) with the help of imitation learning. Here, the pre-trained agent is transferred to the target controller, but it does not operate during the imitation learning phase; instead, the memory buffer of the OTL agent is initialized with transitions from the rule-based controller. This proved to be effective for enhancing the OTL agent during the initial days of development.

In our work, we perform imitation learning for pre-training, after which we fine-tune the agent with PPO. Similar work has been done by Dey et al. [7], where they generated a large dataset with roughly four years' worth of artificial states and discrete action data from a rule-based controller. They pre-train using imitation learning on this data and then fine-tune with PPO. However, in our work, we take inspiration from Ramrakhya et al. [19]. We perform behavioural cloning as a pre-training strategy and adopt the fine-tune methodology proposed via the critic learning and interactive learning phases, showing that the combined strategy can outperform naive fine-tuning while only requiring a single year's worth of data for pre-training.

3 Methodology

3.1 Behavioural Cloning from Demonstrations

Behavioural cloning (BC) uses supervised learning to learn a policy π from a dataset of state-action pairs $\zeta \in D$. It attempts to minimise the difference between the learned policy and expert demonstrations with respect to a defined metric or cost function L. Thus, the optimisation problem for BC can be defined as:

$$\hat{\pi}^* = \arg\min_{\pi} \sum_{\zeta \in D} \sum_{x \in \zeta} L(\pi(x), \pi^*(x)) \tag{1}$$

where $\hat{\pi}^*$ is the approximated policy and $\pi^*(x)$ is the expert action at state x.

Historical data is required to pre-train an agent using BC. For this, we generated an artificial dataset using the actions from the RBC defined in Sect. 4.3. A dataset amounting to a single year's worth of data was generated. Using this data, we then train the actor using BC. Here, the state consists of observation data from the environment consisting of building and weather information, and actions consist of the setpoint values set by the RBC (see Table 2). In our work, we implement the negative log-likelihood loss function given in Eq. 2

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{t=1}^{N} (\log \pi_\theta(a_t|s_t) + \lambda H(\pi_\theta)) \tag{2}$$

where $\pi_\theta(a_t|s_t)$ is the predicted probability of action a_t given state s_t, $H(\pi_\theta)$ is the entropy of the policy distribution π_θ and λ is the weight given to the entropy term for regularisation. λ is set to 0.001 based on the Imitation library [8]. We perform an 80% - 20% split of the training and testing data to evaluate the performance of the trained actor. For testing, we use mean absolute error (MAE) to measure the absolute difference between predicted and actual setpoint temperatures. Table 1 summarises the training and testing losses for the three environments. Each environment was simulated thrice with different seeds.

Table 1. Training and Testing Losses.

Environment	Training Loss	Testing Loss
Hot	–0.519 ± 0.002	8.624e–5 ± 5.36e–5
Mixed	–0.52 ± 0.003	1.065e–4 ± 6.31e–5
Cool	–0.519 ± 0.003	8.325e–5 ± 3.72e–5

However, BC has a few disadvantages. Firstly, the performance of the policy learnt heavily depends on the quality of samples provided by the expert dataset. Additionally, as the expert policy π^* determines the distribution of the sampled states x, the learnt policy π will perform poorly on unseen states. Thus, BC often learns a policy that generalises poorly. This work uses BC only as pre-training to provide good neural network weight initialisation. The pre-trained agent is then fine-tuned with RL, which helps mitigate BC issues and makes the agent more robust and generalisable.

3.2 Reinforcement Learning Fine-Tuning

Reinforcement learning (RL) involves a set of states (S) and control actions (A), where the system dynamics are defined by a probabilistic transition model, denoted as $p(s_{t+1} = s'|s_t = s, a_t = a)$, representing the likelihood of transitioning from state s to state s' by taking action a at time t. Additionally, RL incorporates a reward function $r_t = R(s_t, a_t)$ that provides a reward r_t at timestep t. The goal of an RL agent is to learn a policy π that maximizes its cumulative reward.

Our work uses the Proximal Policy Optimisation (PPO) algorithm for fine-tuning with RL [20]. PPO is a policy gradient algorithm that optimises a parameterised policy using the gradients of the expected return with gradient ascent. It effectively mitigates performance collapse by introducing a clipped surrogate objective function to control policy updates within a specified range, simplifying the optimization process compared to other more complex algorithms. The objective function of the PPO algorithm is defined as:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta), \hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{3}$$

where, $r_t(\theta)$ is the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t,s_t)}$ and $\hat{A}_t$ is the advantage function. PPO is implemented using the Actor-Critic model [13] where the actor has to maximise $L^{CLIP}(\theta)$ and an entropy bonus given as S, while the critic has to minimise the value function error term $L_t^{VF}(\theta)$. Hence, the overall objective function to be maximised can be defined as:

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)] \tag{4}$$

where, c_1, c_2 are coefficients and $L_t^{VF}(\theta)$ is the squared-error loss between the estimated value function and the target value given as $(V_\theta(s_t) - V_t^{targ})^2$.

To perform fine-tuning with PPO, we initialise the actor with the policy weights that were trained with the help of Behavioural Cloning. However, the critic is initialised with random weights. As a result, we would end up with a trained actor and an untrained critic. Thus, inspired by the methodology described by [19], we first train the critic during RL. To do this, we divide the RL training into two phases. Figure 2 describes the learning rate schedules for the best-performing agent.

Critic Learning Phase. During this phase, we collect interactions from the environment using the trained actor to train the critic. In this phase, the actor can be completely frozen (as described in [19]); however, from experiments, we find that gradually increasing the actor learning rate from 0.0 at the start to 1.5×10^{-5} at the 700,000th timestep provides the best results. During this time, the critic is initialised with a relatively high learning rate, which is decayed as training progresses. In our experiments, we find that initialising the critic with a learning rate of 1.0×10^{-3} gave the best results. We maintain this high learning rate for the first 150,000 timesteps.

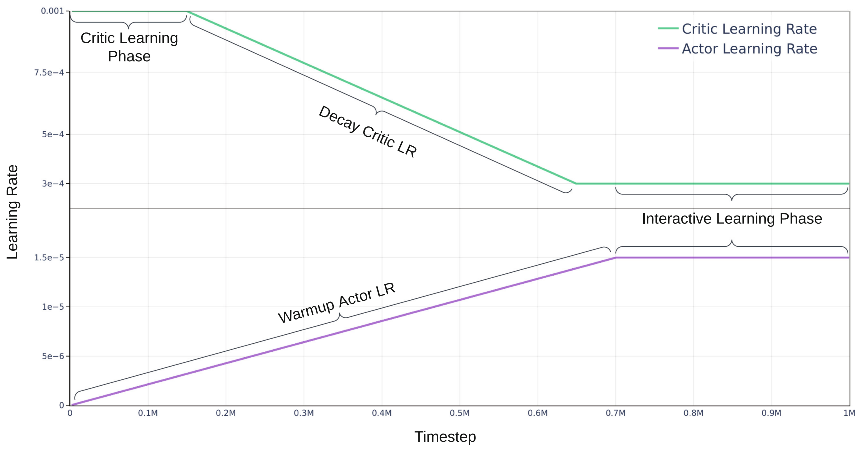


Fig. 2. Actor and Critic learning rate schedules during fine-tuning.

Interactive Learning Phase. After the actor and critic learning rates completely warm up and decay, they are stabilised for the remainder of the training process. We can keep the learning rates during this phase at the same or at different stabilisation values. Through our experiments, we find that keeping the actor and critic learning rates stable at different values yields the best results. For the actor, we stabilise the learning rate at 1.5×10^{-5}, which remains constant from the 700,000th timestep until the end of training. For the critic, we gradually decay the learning rate from 1.0×10^{-3} to 3.0×10^{-4} until the 650,000th timestep, which remains constant until completion.

4 Experimental Setup

4.1 Environment

Our experiments are simulated on the `5ZoneAutoDXVAV` environment provided by the Python library known as Sinergym [10][4] (v3.1.7). The `5ZoneAutoDXVAV` is a single-storey building divided into one indoor and four outdoor zones[5]. The state space and action space for the environment are given in Table 2. We simulate three different weather conditions in our experiments using EnergyPlus. The hot weather data is from Davis-Monthan AFB, Arizona, USA; the mixed weather data is from New York City, New York, USA, and the cool weather data is from Port Angeles, Washington, USA. All environments are initialised with stochasticity in weather. This stochasticity is introduced with the Ornstein-Uhlenbeck [3] process where σ, μ and τ are 1.0, 0.0 and 0.001, respectively. Each simulated episode spans over a duration of one year. Within each episode, 35,040 intervals exist, lasting fifteen minutes each.

[4] https://ugr-sail.github.io/sinergym/compilation/main/index.html.
[5] https://ugr-sail.github.io/sinergym/compilation/main/pages/buildings.html#zone.

Table 2. Environment Summary.

	Variable Names	Number
State Space	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, zone people occupant count, people air temperature, facility total HVAC electricity demand rate, hour, day and month	19
Action Space	Heating setpoint and Cooling setpoint	2

4.2 Rewards

The goal of the Deep RL agent is to reduce energy usage while maintaining a comfortable temperature range. This is achieved through an objective function that combines the weighted sum of energy consumption and thermal discomfort, which are normalised. Equation 5 describes the reward function.

$$R = -\omega \times \lambda_P \times P_t - (1 - \omega) \times \lambda_T \times \exp(|T_t - T_{upper}| + |T_t - T_{lower}|) \quad (5)$$

where, P_t is the power consumption and T_t is the current indoor temperature. ω represents the weight assigned to power consumption and thus $(1 - \omega)$ is the weight assigned to comfort. λ_P and λ_T are scaling constants for power consumption and comfort penalties, respectively. T_{upper} and T_{lower} define the upper and lower limits of the comfort temperature range. Discomfort is determined by calculating the absolute difference between the current temperature and the comfort range. If the temperature falls within the comfort range, the discomfort value is zero. Along with rewards, we also analyse two other Key Performance Indicators (KPIs):

- Comfort Violation Time (%): Percentage of time that the temperature has been beyond the bounds of the user comfort temperature ranges.
- Mean Power: Average power consumption per step in the episode.

4.3 Training Setup

In our work, we implement Behavioural Cloning with the help of the Imitation library [8] and the PPO algorithm with the help of CleanRL [9]. Table 3 represents the default hyperparameters for the PPO algorithm included in CleanRL. Following CleanRL, the actor and the critic networks are built using separate neural networks. Thus, there are no shared weights. The neural network architecture for both the actor and the critic consists of a single hidden layer of size 64.

Table 3. PPO Hyperparameters.

Hyperparameter Name	Value
Total Timesteps	1,000,000
Learning Rate	3.0e–4
Number of Steps per Policy Rollout	2048
Anneal Learning Rate	True
Gamma	0.99
General Advantage Estimation	0.95
Minibatch Size	32
Update Epochs	10
Advantage Normalisation	True
Surrogate Clipping Coefficient	0.2
Clip loss for Value Function	True
Entropy Coefficient	0.0
Value Function Coefficient	0.5
Maximum Norm for Gradient Clipping	0.5

The layer initialisation included in CleanRL was skipped. The Tanh activation function is used after every layer except the output layer. The performance of the learned agents is compared to the Rule-Based Controllers (RBC) provided by Sinergym[6]. The rules for the RBC are defined in Algorithm 1. During the

Algorithm 1 Rule for RB Controller for 5Zone

```
summer_setpoint ← (22.5, 26.0)
winter_setpoint ← (20.0, 23.5)
summer_range ← (1 June, 30 September)
for each step in environment do
    if current_date is in summer_range then
        curr_setpoint ← summer_setpoint
    else
        curr_setpoint ← winter_setpoint
    end if
end for
```

training process, we periodically evaluate the performance of the agents in the same environment but with a different seed. We evaluate its performance for the entire year, i.e. one episode. For testing, we follow a similar procedure to the evaluation, where we test the agent in the same environment but with a different seed and monitor its performance for a total of five years, i.e. five episodes. Our

[6] https://github.com/ugr-sail/sinergym/blob/main/sinergym/utils/controllers.py.

implementation source code can be accessed at GitHub - https://github.com/kad99kev/HVACIRL.

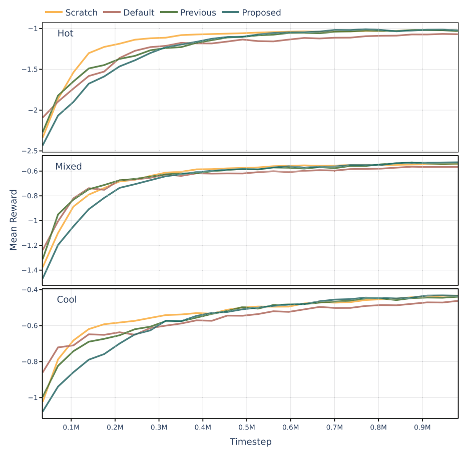
5 Results

In our experiments, we compare the proposed methodology with different agents. The RBC or **controller** agent is the rule-based controller defined in Sect. 4.3 (see Algorithm 1). The **imitation** agent is the behavioural cloning agent without fine-tuning. We compare this agent to highlight the performance of the pre-trained agent before we fine-tune it with RL. The **scratch** agent is an RL agent trained without any pre-training. It is trained with the default hyperparameters and training methodology of PPO (see Table 3). The **default** agent is a pre-trained agent that is fine-tuned naively. For this agent, we follow the same RL training methodology as the scratch agent; however, the weights of the actor are initialised with BC. Thus, fine-tuning is naive, where a pre-trained actor and an untrained critic interact with each other. Finally, we include the agent described in [7] (called **previous**), where the authors utilised a learning rate for the policy network that was one-hundredth of the learning rate of the baseline RL learning strategy for the initial ten episodes and one-tenth for subsequent training episodes. We include these agents to show how naive fine-tuning can affect the performance of an agent and, thus, the need for a different fine-tuning methodology. In the following sections, we refer to the agent trained using our proposed methodology as the **proposed** agent.

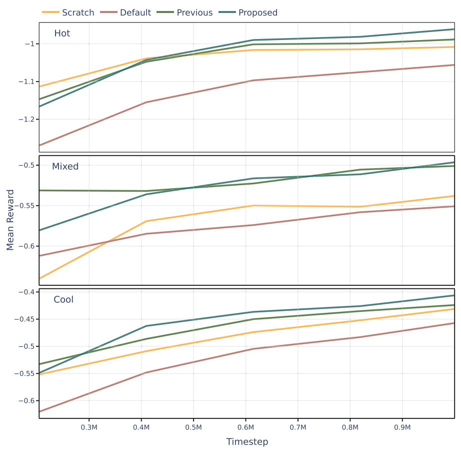
5.1 Performance Analysis

In this section, we compare the training, evaluation and testing performance of the different agents. Figure 3a shows that within 400,000 timesteps, our **proposed** agent begins outperforming all other RL agents. When we compare the performance on evaluation, Fig. 3b, shows that our **proposed** agent learns faster than the **default** agent. Additionally, our **proposed** agent consistently and significantly outperforms the **scratch** agent. This implies that pre-training the agent helps it generalise better. However, Fig. 3b also suggests that the fine-tuning methodology plays an important role. Naively fine-tuning a pre-trained agent with default learning rates can lead to poor policies being learnt, as it is evident that the **default** agent cannot outperform the **scratch** agent. Along with this, our **proposed** agent also outperforms the **previous** agent.

When comparing the rewards and KPIs (see Table 4) of the fine-tuned RL agents with an agent trained solely on the imitation learning dataset and the RBC, our **proposed** agent achieves the highest reward across all three weather environments during testing. In the 5-Zone hot weather environment, the **proposed** agent outperforms the **previous** agent by 2.83%, the **default** agent by 9.17% and the **scratch** agent by 5.04% while significantly outperforming the **controller** agent by 21.87%. While it may not be the best at saving power, the **proposed** agent has the least number of comfort violations among all agents.



(a) Mean reward during training.



(b) Mean reward during evaluation.

Fig. 3. Learning curves of the different agents during training and evaluation on hot, mixed and cool weather.

Table 4. Rewards and KPIs average performance summary during testing.

Environment	Agent	Rewards (higher better)	Power Consumption (lower better)	Comfort Violations (%) (lower better)
5-Zone Hot	Controller	−1.230	5939.19	41.82
	Imitation	−2.936	**5623.01**	50.80
	Scratch	−1.012	6834.64	31.23
	Default	−1.058	6614.23	33.48
	Previous	−0.989	6942.00	30.47
	Proposed	**−0.961**	6836.64	**28.81**
5-Zone Mixed	Controller	−0.619	**5725.47**	45.45
	Imitation	−1.793	5991.53	48.29
	Scratch	−0.538	7259.21	25.73
	Default	−0.550	7131.13	28.06
	Previous	−0.501	7027.80	23.48
	Proposed	**−0.496**	6917.05	**22.72**
5-Zone Cool	Controller	−0.455	**3751.00**	29.24
	Imitation	−1.385	4033.08	43.42
	Scratch	−0.429	5548.59	20.56
	Default	−0.458	5260.54	22.34
	Previous	−0.424	5314.49	20.09
	Proposed	**−0.406**	5021.88	**19.07**

As the table shows, the **imitation** and **controller** agents are the best at saving power; however, they have the highest percentage of comfort violations. The **proposed** agent, however, can balance this by not overly spending power yet significantly reducing comfort violations. Similar behaviour is observed in the 5-Zone mixed weather environment as well. The **proposed** agent demonstrates significant enhancements in overall rewards, with improvements of 1.00%, 9.82%, 7.81%, and 19.87% compared to **previous**, **default**, **scratch**, and **controller** agents, respectively. Again, we observe that the **imitation** and **controller** agents are the best at saving power but are the poorest at maintaining comfort. The **proposed** agent stands out for its efficient power utilisation and minimal comfort violations among all RL agents evaluated. Likewise, the **proposed** agent performs best in the 5-Zone cool weather environment. Compared to **previous**, **default**, **scratch** and **controller** agents, it sees a 4.25%, 11.35%, 5.36% and 10.77% improvement in the overall rewards, respectively. Here, the performance of the **default** agent is comparable to the **controller** agent. The **proposed** agent achieves a 5.51% decrease in power consumption compared to the **previous** agent, resulting in a 1.02% reduction in comfort violations. Similar to the mixed weather environment, the **proposed** agent performs best in both power consumption and comfort violations when compared to the other RL agents.

5.2 Policy Analysis

This section compares the policies adopted by the different agents in our experiments. We analyse their heating and cooling setpoint temperatures and their effect on the indoor temperature. Hourly observed temperatures show the average temperatures at each hour throughout the year, while monthly observed temperatures show the average temperature for each month.

From Fig. 4, we can see that the **proposed** agent and the **previous** agent follow similar policies; however, the **proposed** agent maintains a slightly higher cooling and heating setpoint on average. The **imitation** agent follows the RBC policy very closely. In all three environments, we observe that the **scratch** agent drastically changes its setpoint temperatures as the outdoor temperature becomes warmer during the day. We can also see that the RL agents consistently maintain indoor temperatures within the average hourly user comfort zone, while the **imitation** and **controller** agents periodically violate comfort.

When comparing the monthly setpoint temperatures, from Fig. 5, we see that the **imitation** agent fails to adapt its setpoint temperatures during summer

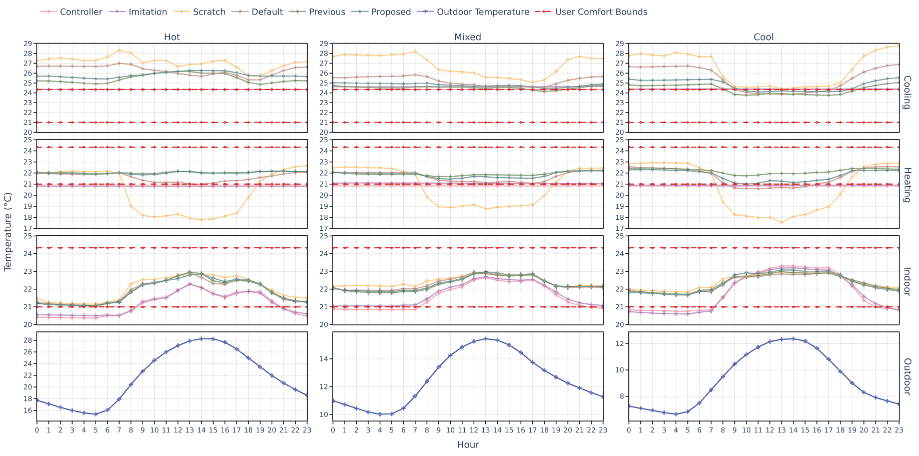


Fig. 4. Setpoint, indoor and outdoor temperatures summarised by the hour.

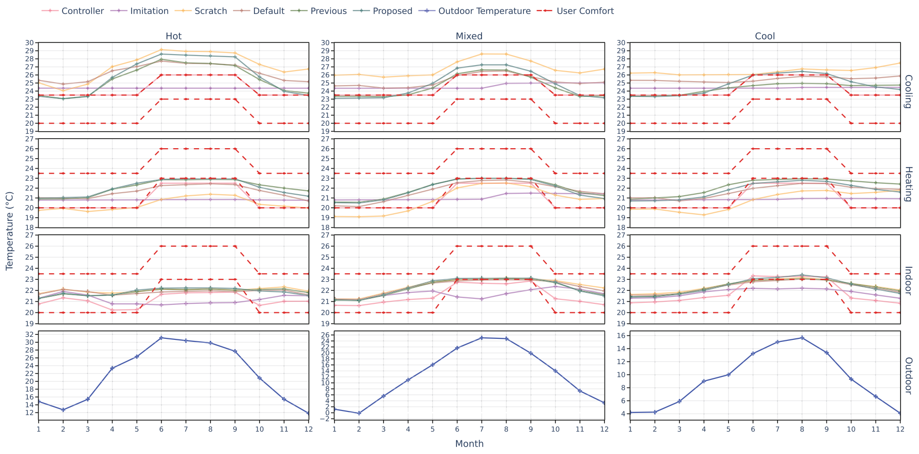


Fig. 5. Setpoint, indoor and outdoor temperatures summarised by the month.

when the temperatures increase. The graphs highlight how RL agents adapt their policies as user comfort and season change. We can see that the cooling setpoints are higher in hot weather, which decreases as we move to mixed weather, with it being the lowest in cool weather. The heating setpoints are relatively the same for hot and mixed weather but are slightly lower for cool weather. All agents struggle to maintain comfort in hot weather; however, the **proposed** agent does the best to maintain indoor temperatures. For other weather conditions, the RL agents are better able to maintain comfortable indoor temperatures. The RL agents are much better at keeping indoor temperatures closer to the user's comfort zone than the **controller** agent, which does well for mixed and cool weather. The **imitation** agent, however, does poorly throughout. This proves the need for RL fine-tuning, as the RL agents are able to adapt their setpoint temperatures well to maintain indoor comfort.

6 Conclusion and Future Work

Imitation learning helps provide a foundation for the RL agent during training. However, naively training the trained actor and untrained critic together can lead to worse policies being learnt than training from scratch. Thus, to avoid this, we can either freeze the actor or initialise it with a very low learning rate while the critic learns from interactions with the environment. From our experiments, we see that following this methodology results in agents outperforming not only the RBC but also naively fine-tuned agents and agents trained from scratch across all three weather environments. We also build upon prior work, showing that a better learning rate tuning strategy is able to outperform their agent.

The policies learnt by the proposed agent can better maintain indoor temperatures within the user's comfort bounds than all other agents. While this comes at the expense of more power, the least power-consuming agents are RBC and imitation. These two agents are the worst at respecting the user's comfort zone. Thus, the proposed agent is efficient at striking a balance between power consumption and user comfort.

For future work, a hyperparameter study could be conducted to analyse the effect of different hyperparameters on this approach. Additionally, historical data could be utilised to train an autoencoder, thus reducing the dimensionality of the observation space for effective representation learning. Finally, a multi-objective approach to this method can be considered. By separating the energy and comfort variables, further analysis can be conducted on different policies learnt to prioritise comfort and/or energy.

Acknowledgments. This work was conducted with the financial support of the Science Foundation Ireland Centre for Research Training in Artificial Intelligence under Grant No. 18/CRT/6223.

Ethical Impact. We do not foresee any ethical implications for our work. Our experiments do not involve personal data; all data and simulations used are publicly available on GitHub – https://github.com/ugr-sail/sinergym.

References

1. Arroyo, J., Manna, C., Spiessens, F., Helsen, L.: Reinforced model predictive control (rl-mpc) for building energy management. Appl. Energy **309**, 118346 (2022)
2. Barrett, E., Linder, S.: Autonomous hvac control, a reinforcement learning approach. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7–11, 2015, Proceedings, Part III 15, pp. 3–19. Springer (2015). https://doi.org/10.1007/978-3-319-23461-8-1
3. Benth, F.E., Šaltytė-Benth, J.: Stochastic modelling of temperature variations with a view towards weather derivatives. Appl. Math. Finance **12**(1), 53–85 (2005)
4. Biemann, M., Scheller, F., Liu, X., Huang, L.: Experimental evaluation of model-free reinforcement learning algorithms for continuous hvac control. Appl. Energy **298**, 117164 (2021)
5. Chen, B., Cai, Z., Bergés, M.: Gnu-rl: a precocial reinforcement learning solution for building hvac control using a differentiable mpc policy. In: Proceedings of the 6th ACM international conference on systems for energy-efficient buildings, cities, and transportation, pp. 316–325 (2019)
6. Coraci, D., Brandi, S., Hong, T., Capozzoli, A.: Online transfer learning strategy for enhancing the scalability and deployment of deep reinforcement learning control in smart buildings. Appl. Energy **333**, 120598 (2023)
7. Dey, S., Marzullo, T., Zhang, X., Henze, G.: Reinforcement learning building control approach harnessing imitation learning. Energy AI **14**, 100255 (2023)
8. Gleave, A., et al.: Imitation: Clean imitation learning implementations. arXiv:2211.11972v1 [cs.LG] (2022). https://arxiv.org/abs/2211.11972
9. Huang, S., et al.: Cleanrl: high-quality single-file implementations of deep reinforcement learning algorithms. J. Mach. Learn. Res. **23**(274), 1–18 (2022). http://jmlr.org/papers/v23/21-1342.html

10. Jiménez-Raboso, J., Campoy-Nieves, A., Manjavacas-Lucas, A., Gómez-Romero, J., Molina-Solana, M.: 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)
11. Kadamala, K., Chambers, D., Barrett, E.: Enhancing hvac control systems through transfer learning with deep reinforcement learning agents. Smart Energy, p. 100131 (2024)
12. Kebria, P.M., Khosravi, A., Salaken, S.M., Nahavandi, S.: Deep imitation learning for autonomous vehicles based on convolutional neural networks. IEEE/CAA J. Automatica Sinica **7**(1), 82–95 (2019)
13. Konda, V., Tsitsiklis, J.: Actor-critic algorithms. Advances in neural information processing systems **12** (1999)
14. Li, B., Xia, L.: A multi-grid reinforcement learning method for energy conservation and comfort of hvac in buildings. In: 2015 IEEE International Conference on Automation Science and Engineering (CASE), pp. 444–449. IEEE (2015)
15. Lissa, P., Deane, C., Schukat, M., Seri, F., Keane, M., Barrett, E.: Deep reinforcement learning for home energy management system control. Energy AI **3**, 100043 (2021)
16. Lissa, P., Schukat, M., Barrett, E.: Transfer learning applied to reinforcement learning-based hvac control. SN Comput. Sci. **1**(3), 1–12 (2020)
17. Liu, H.Y., Balaji, B., Gupta, R., Hong, D.: Rule-based policy regularization for reinforcement learning-based building control. In: Proceedings of the 14th ACM International Conference on Future Energy Systems, pp. 242–265 (2023)
18. Osa, T., Sugita, N., Mitsuishi, M.: Online trajectory planning and force control for automation of surgical tasks. IEEE Trans. Autom. Sci. Eng. **15**(2), 675–691 (2017)
19. Ramrakhya, R., Batra, D., Wijmans, E., Das, A.: Pirlnav: pretraining with imitation and rl finetuning for objectnav. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 17896–17906 (2023)
20. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
21. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: a survey. J. Mach. Learn. Res. **10**(7) (2009)
22. Vázquez-Canteli, J.R., Nagy, Z.: Reinforcement learning for demand response: a review of algorithms and modeling techniques. Appl. Energy **235**, 1072–1089 (2019)
23. Wei, T., Wang, Y., Zhu, Q.: Deep reinforcement learning for building HVAC control. In: Proceedings of the 54th Annual Design Automation Conference 2017, pp. 1–6 (2017)
24. Yu, Z., Dexter, A.: Online tuning of a supervisory fuzzy controller for low-energy building system using reinforcement learning. Control. Eng. Pract. **18**(5), 532–539 (2010)
25. Zhu, Z., Lin, K., Jain, A.K., Zhou, J.: Transfer learning in deep reinforcement learning: a survey. IEEE Trans. Pattern Anal. Mach. Intell. (2023)