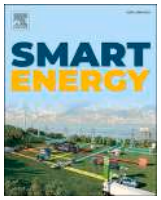
Smart Energy 13 (2024) 100131

Contents lists available at ScienceDirect

Smart Energy

journal homepage: www.journals.elsevier.com/smart-energy



Enhancing HVAC control systems through transfer learning with deep reinforcement learning agents

Kevlyn Kadamala *, Des Chambers, Enda Barrett

School of Computer Science, University of Galway, University Road, Galway, H91 TK33, Galway, Ireland

ARTICLE INFO

Keywords:
Transfer learning
Reinforcement learning
Continuous HVAC control

ABSTRACT

Traditionally, building control systems for heating, ventilation, and air conditioning (HVAC) relied on rule-based scheduler systems. Deep reinforcement learning techniques have the ability to learn optimal control policies from data without the need for explicit programming or domain-specific knowledge. However, these data-driven methods require considerable time and data to learn effective policies without prior knowledge. Performing transfer learning using pre-trained models avoids the need to learn the underlying data from scratch, thus saving time and resources. In this work, we evaluate reinforcement learning as a method of pretraining and fine-tuning neural networks for HVAC control. First, we train an RL agent in a building simulation environment to obtain a foundation model. We then fine-tune this model on two separate simulation environments such that one environment simulates the same building under different weather conditions while the other environment simulates a different building under the same weather conditions. We perform these experiments with two different reward functions to evaluate their effect on transfer learning. The results indicate that transfer learning agents outperform the rule-based controller and show improvements in the range of 1% to 4% when compared to agents trained from scratch.

1. Introduction

Heating, ventilation and air conditioning (HVAC) play a significant role in ensuring occupant comfort. However, if implemented incorrectly, it could lead to large amounts of energy consumption, driving up costs while reducing occupant satisfaction. Reinforcement learning (RL) has proven to be a viable alternative [1–3] of adaptive systems that operate without the need to capture the physical building model, thus offering a more flexible approach alongside rule-based and manual HVAC systems which continue to serve valuable purposes. These RL systems learn an optimal policy by interacting with the environment using actions learnt through their previous experience. However, the drawback of RL is that it is sample inefficient, requiring a considerable amount of time and training resources to reach a desirable level of performance. Thus, this paper aims to see whether a transfer learning approach can help address this challenge. More specifically, we re-use network weights trained on different building control tasks and evaluate their performance relative to sample efficiency. The work aims to enhance smart energy systems research by evaluating whether knowledge from a pre-trained agent can be leveraged to efficiently reduce comfort violations and power consumption over time.

In domains like natural language processing (NLP) and computer vision (CV), most solutions for a particular task begin with using some pre-trained model. This pre-trained model is usually trained on a related task and then fine-tuned on the new task [4–7]. As a result, complex problems can be tackled without the need for extensive computation resources. There are various model repositories available online [8], HuggingFace Models [1] Tensorflow Model Hub [2] that enables researchers to download and load any NLP or CV model and use it on their task directly or perform fine-tuning on a new domain-specific task. Such practice, however, has not been widely adopted in RL for HVAC systems.[3] Our work addresses this gap by providing model weights that can be later used for fine-tuning building control systems.

HVAC control systems vary depending on the location, building size, structure, and material and work under different weather conditions. Imagine we have an RL agent that was already trained on a (source)

* Corresponding author.
E-mail address: k.kadamala1@universityofgalway.ie (K. Kadamala).

[1] https://huggingface.co/models.

[2] https://tfhub.dev/.

[3] HuggingFace Model Hub for RL - https://huggingface.co/models?pipeline_tag=reinforcement-learning.

https://doi.org/10.1016/j.segy.2024.100131
Received 15 September 2023; Received in revised form 17 January 2024; Accepted 17 January 2024
Available online 26 January 2024

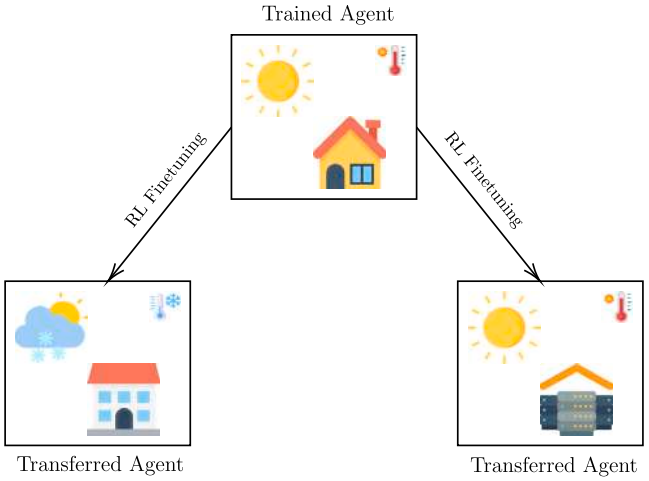


Fig. 1. Transfer Learning Methodology Scenario.

building. We ask the question - can we adapt it to a different (target) building that may differ with respect to its characteristics or climate? Thus, in this work, we use building environments provided by Sinergym [9] to train an RL agent on a source building and then adapt it to a target building that may be of a similar structure but in a different environment or a different structure in a similar environment. However, the source building and the target building may differ with respect to the data available for control and the amount of equipment to be controlled. Thus, to address this problem, we only initialise the input and output layers of the agent based on the target building while using the pre-trained weights of the hidden layers trained on the source building. We then fine-tune this new agent on the target building. Fig. 1 shows the scenario where a trained agent is fine-tuned to different buildings in different weather condition.[4] We evaluate the fine-tuned agent's performance against an agent trained from scratch. Finally, we evaluate both RL agents against a rule-based controller to provide an overall idea of their performance. Thus, to summarise, the main contributions of our work are:

- Evaluate two reinforcement learning algorithms as a pre-training and fine-tuning methodology that can adapt to different buildings irrespective of their characteristics and environments.
- Analyse their performance with and without transfer learning with respect to a rule-based controller.
- Release model weights trained on different buildings to the research community to enable further study and evaluation of transfer learning for building control tasks.

2. Background and related work

2.1. Markov decision processes and reinforcement learning

Reinforcement Learning (RL) is a computational learning method which learns through interactions with the environment [10]. An RL algorithm attempts to solve a Markov Decision Process (MDP). An MDP is based on the Markov Property, which states that the future state is entirely dependent upon its current state and is independent of the past. An MDP consists of four components:

- A set of states S.
- A set of actions A.
- A probability distribution of state transitions T.
- A reward function R.

[4] *Icons for Fig. 1 were obtained from https://www.flaticon.com/.*

The MDP task for building control systems is discretised into timesteps. At each timestep, the RL agent observes the building state s_t and then chooses to perform an action a_t from the possible actions. The executed action would then transition the RL agent into a new state s_{t+1} with reward r. Thus, the RL agent will then attempt to learn a policy π that would maximise its cumulative reward. Reinforcement learning algorithms can be on-policy or off-policy. The difference is based on how they use the training data. As the name implies, on-policy algorithms only utilise data sampled from the current policy π to learn. On the other hand, off-policy algorithms can use any data (from any policy) collected during training. This makes it more sample efficient; however, storing data may require more memory. The authors in [11] evaluated different model-free RL algorithms for continuous HVAC control. We take inspiration from this work and assess our transfer learning methodology using the Proximal Policy Optimisation (PPO) and Soft Actor-Critic (SAC) algorithms.

2.1.1. PPO

Policy gradient algorithms attempt to directly optimise a parameterised policy based on the gradients of the expected return with respect to the policy parameters using gradient ascent. However, policy gradient algorithms are prone to performance collapse, wherein an agent suddenly starts to perform poorly. Proximal Policy Optimisation [12] is an on-policy, policy gradient algorithm that addresses this issue. It updates policies via the objective function:

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

where, $r_t(\theta)$ is the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t,s_t)}$. PPO can be implemented as an extension of Actor-Critic [13] where the actor has to maximise $L^{CLIP}(\theta)$, while the critic has to minimise the squared error between the estimated value function and the target value $L_t^{VF}(\theta) = (V_\theta(s_t) - V_t^{targ})^2$. To encourage exploration, PPO adds an entropy bonus given as S. Thus, the overall objective function to be maximised becomes:

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)] \tag{2}$$

where, c_1 and c_2 are coefficients.

2.1.2. SAC

Like the PPO, the Soft Actor-Critic [14] is an actor-critic policy gradient algorithm; however, it is an off-policy algorithm, based on the maximum entropy framework:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\ \rho_\pi}[r(s_t,a_t) + \alpha \text{H}(\pi(\cdot|s_t))] \tag{3}$$

The advantage of this objective is that along with encouraging optimal behaviour, it incentivises exploration to avoid local optima convergence. Using this framework, the soft policy iteration algorithm learns optimal maximum entropy policies, alternating between policy evaluation and policy improvement. However, this is not practical for large continuous tasks. To address this issue, SAC utilises function approximators for two soft Q-functions and the policy. The loss for the soft Q-functions with the entropy regularised, and the soft Bellman update target is given as:

$$J(\theta_i^Q) = \mathbb{E}_{(s,a,r,s')\sim D}[Q_{\theta_i}(s,a) - r(s,a) + \gamma \min_{\theta_{1,2}} Q_{\theta_i}(s',a') - \alpha log(\pi_\phi(\cdot|s'))] \tag{4}$$

Here, D is the replay buffer that stores transitions during training, and $a' \sim \pi(\cdot|s), log(\pi_\phi(\cdot|s))$ approximates the entropy of the policy. Additionally, the objective for the policy network is given as:

$$J_\pi(\phi) = \mathbb{E}_{s\sim D}[\min_{i=1,2} Q_{\theta_i}(s,a) - \alpha log(\pi_\theta(a,s))] \tag{5}$$

The action a is sampled using the reparameterisation trick where $\tilde{a_\phi}(s,\xi) = tanh(\mu_\phi(s) + \sigma_\phi \odot \xi)$ with $\xi \sim N(0,1)$. Finally, Haarnoja et al. [15] suggest that fixing the value of entropy is a bad choice as the policy should be encouraged to explore regions where the optimal action is unknown (high entropy) while encouraging it to be more deterministic when the near-optimal policy has already been learnt. For this, as the policy is being trained, they dynamically adjust α as:

$$\alpha_t^* = argmin_{\alpha_t} \mathbb{E}_{a_t \sim \pi_t^*}[-\alpha_t log(\pi_t^*(a_t|s_t;\alpha_t)) - \alpha_t \bar{\mathcal{H}}] \quad (6)$$

where $\bar{\mathcal{H}}$ is the target entropy set to the dimension of the action space of the task.

Additionally, we experiment with the "learning starts" hyperparameter of the SAC algorithm to evaluate the effect that the pre-trained policy has on exploration. The "learning starts" hyperparameter is the number of steps before the agent starts to update the policy and value networks, during which random actions are performed. We would be directly leveraging the pretrained policy by setting this to zero.

2.2. Reinforcement learning in HVAC control

Reinforcement learning (RL) has emerged as a promising technique for optimising the control of Heating, Ventilation, and Air Conditioning (HVAC) systems in buildings. The use of RL in HVAC control has attracted considerable attention due to its ability to learn optimal control policies in complex and dynamic environments, such as buildings. Barrett and Linder [16] proposed a tabular Q-learning with an occupancy prediction approach where their approach optimised user comfort and reduced energy costs compared to methods like "always on" and "programmable". Lissa et al. [3] applied deep Q-learning to control space heating and domestic hot water temperature, giving up to 16% in energy savings by optimising the PV consumption. Fang et al. [17] also propose a DQN framework for building HVAC system control. The authors trained a DQN model on a multi-zone office building. Their results showed that the DQN control strategy can reduce the energy consumption of the HVAC system for 11/14 of all fixed temperature setpoint comparison cases, while maintaining acceptable indoor air temperature violations. Dmitrewski et al. [1] propose using RL with data assimilation (DA), a commonly used technique in numerical weather prediction. They showed that an RL control agent with DA maintains the desired temperature range with 15.6% higher frequency than the RL control agent operating without DA. Similarly, Arroyo et al. [2] integrate model predictive control (MPC) with reinforcement learning. In RL-MPC, the goal is to learn an ideal policy while ensuring that it satisfies every constraint. The authors reported that the MPC controller is more effective than a naive RL algorithm, as it has poor constraint satisfaction. However, implementing a state estimator and a one-step-ahead optimiser enables an RL-MPC controller to achieve similar results as the MPC. We use a Python simulator called Sinergym [9] (v2.3.2) to run our experiments. Sinergym is based on the OpenAI gym interface, is compatible with EnergyPlus models, and provides weather variability, customisable reward functions and action spaces.

2.3. Transfer learning with RL in HVAC control

Given a source domain or task, transfer learning attempts to learn an optimal policy for a target domain or task using information learnt from the source domain [18]. The survey paper by Taylor and Stone [19] includes metrics to measure the benefit of transfer learning, along with classifying different techniques based on their approaches and goals. To measure the benefit of transfer learning, in our work, we monitor the time to threshold along with the improvement of total reward accumulation through transfer compared to learning without transfer. We perform experiments where the environments either have the same state and action variables or have different state and action variables with no explicit task mappings.

For building control systems, Lissa et al. [20] explore whether learning can be transferred between HVAC agents with different spatial and geographical characteristics. Using tabular Q-learning, the authors report that with transfer learning, a user experiences temperatures out of their comfort range for only 3% of the day, compared to 7% to 36% without transfer learning. They also report that transfer learning helps achieve faster optimal convergence when there are geographical changes. Thus, by extending the work done by them [20,3], we evaluate transfer learning in a deep learning setup. Zhang et al. [21] apply transfer learning to different homes where they have the same types and the number of appliances to control but differ with respect to other parameters or user preferences. They show that applying transfer learning can effectively reduce the training time of a new policy if the two homes are similar but suggest that the advantages of transfer learning would diminish otherwise. A novel approach for transfer learning proposed by Xu et al. [22] suggests that dividing the design of the neural network controller into a transferable front-end network and a building-specific back-end network can learn the controls required for a target building with minimum effort and better performance. The intuition behind this is that the front-end network captures building-agnostic behaviour while the back-end network would be efficiently trained for each specific building. They compare their work to prior work [23] and with an ON-OFF controller, showing that their proposed approach successfully transfers the Deep RL controller from a source building to a target building. Apart from building HVAC systems, transfer learning has also been applied to microgrids. Lissa et al. [24] have applied transfer learning to Deep RL on a microgrid of five houses, where they control a heat pump for domestic hot water usage. They experiment with three approaches. They use independent learners for each house, which acts as the baseline, independent learners with transfer learning where knowledge is shared amongst the agents, and a global agent that controls all heat pumps in the microgrid. They reported that their transfer learning approach reduced the time to learn near-optimal policies by more than a factor of five when considering the entire cluster of houses. Fang et al. [25] propose a methodology where they transfer the network weights from a pre-trained model onto a target DQN model after which they perform finetuning. They perform experiments that involve transferring the first layer, transferring the first two layers and transferring the first three layers. Their proposed methodology can improve the training efficiency by about 3%–29% compared to that of the DRL models trained from scratch.

Having detailed the literature in the area, no prior work has examined the transferability of a previously trained single deep neural agent onto building HVAC control systems that vary not only geographically but also structurally.

3. Methodology

Transfer learning in other domains, like computer vision and natural language processing, usually involves adapting the output layer and then fine-tuning on domain-specific data. However, in the context of the building HVAC control problem, different buildings will differ with respect to their observation and action spaces. For example, a data centre will have more input variables to consider than a house; thus, its observation space would be larger than the house. Hence, a Deep RL agent would need to adapt its input layer and the output layer according to the dimensionality of the building control problem. This involves randomly initialising the weights of the new network for each new building. However, instead of completely creating a new neural network, we simply reinitialise the input and output layers of the network while keeping the hidden layers (Core) constant.

In order to perform this experiment, we need to train (or pre-train) an agent on the source building to evaluate the transfer process on the target building. Consider Fig. 2, Building A is our source building that has an observation space $O_A = \{o_1, o_2, ...o_n\}$ and an action space $A_A = \{a_1, a_2, ...a_n\}$. We train (pre-train) a Deep RL agent on this build-

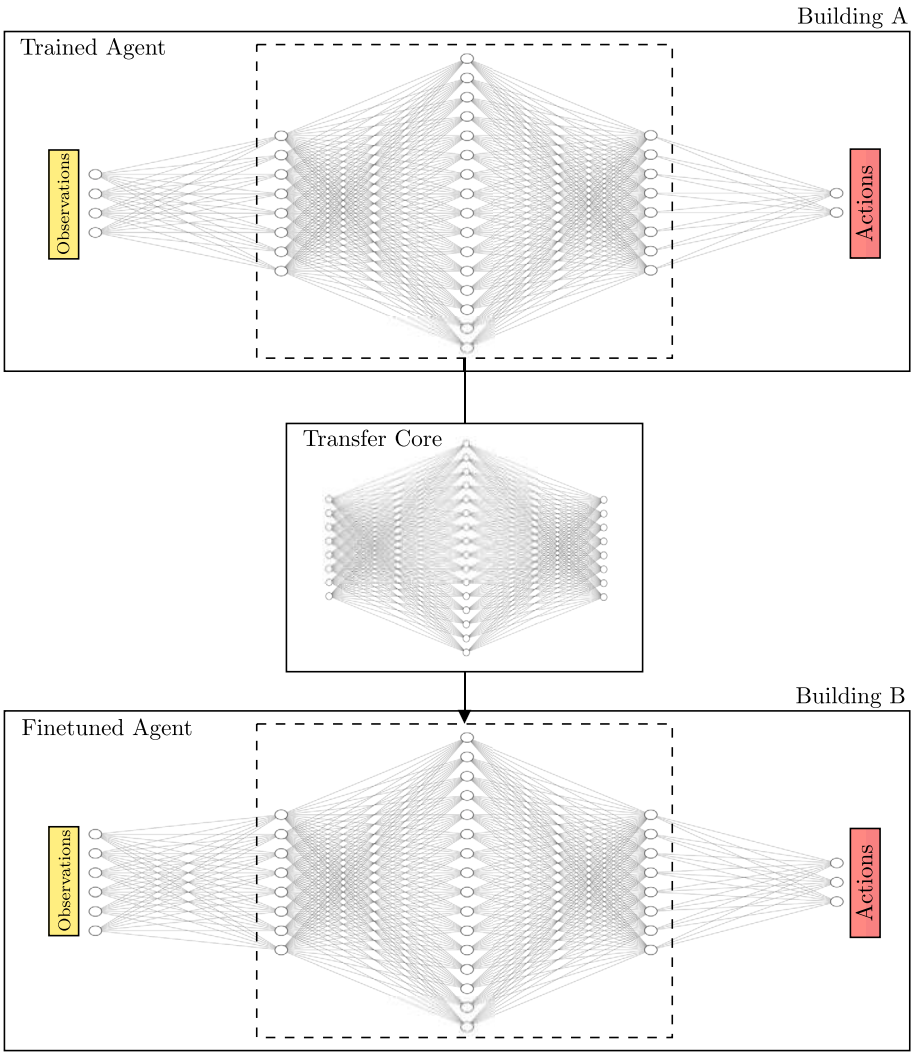


Fig. 2. Overview of the Transfer Learning Methodology.

ing using one of the two algorithms described in Section 2.1. During training, we save the neural network weights for the actor and the critic, which are later used for the transfer process. Now consider a target building (Building B), where the observation space is defined as $O_B = \{o_1, o_2, ...o_m\}$ and an action space $A_B = \{o_1, o_2, ...o_m\}$ such that $n \neq m$. Here, the input and the output dimensionalities of the neural networks will not match. Hence, a direct transfer would not be possible. To solve this issue, we initialise new input and output layers to match the dimensionality of the new observation and action spaces. We then transfer the Core with the pre-trained weights onto the new network to create a combined architecture. The combined architecture is then further trained (fine-tuned) using the same algorithm. The source code[5] and model weights[6] from our implementation is available online.

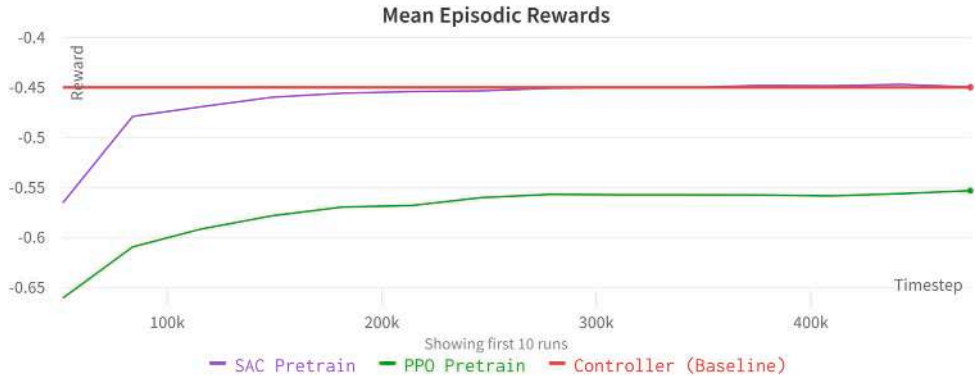
4. Experimental setup

4.1. Environment

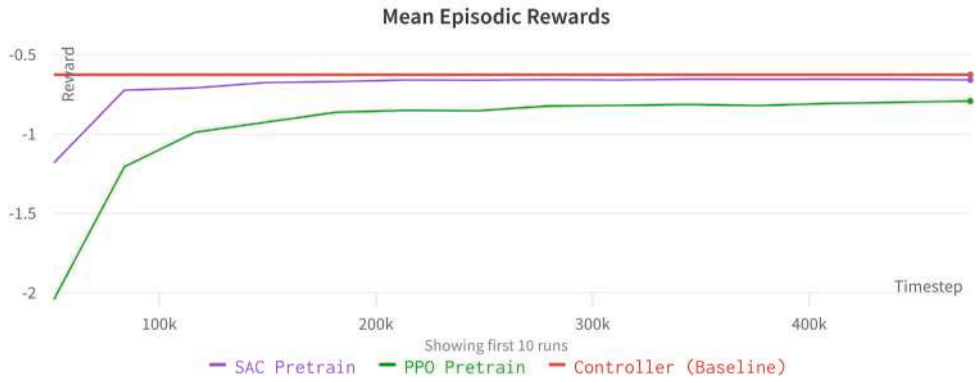
To simulate the building environments, we make use of a Python library[7] called Sinergym [9] (v2.3.2). In particular, we use two building environments to evaluate our methodology. The first is a 5-Zone single-storey building divided into one indoor and four outdoor zones, while the second is a two-zone data centre where the main heat source is the hosted servers.[8] Based on prior literature [11,26,27], the `5ZoneAutoDXVAV.idf`[9] and `2ZoneDataCenterHVAC_wEconomizer.idf`[10] are commonly used EnergyPlus input data files. Hence, for accessibility reasons, we chose these environments as they are supported by Sinergym and EnergyPlus. The state spaces are given as:

- 5-Zone: 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

[5] https://github.com/kad99kev/EHCSTLDRL.
[6] https://drive.google.com/drive/folders/1NAqbOnqyBS1vOKxMv4B76mZ0cDBTLjSt?usp=drive_link.
[7] https://ugr-sail.github.io/sinergym/compilation/v2.3.2/index.html.
[8] https://ugr-sail.github.io/sinergym/compilation/v2.3.2/pages/buildings.html.
[9] https://github.com/ugr-sail/sinergym/blob/v2.3.2/sinergym/data/buildings/5ZoneAutoDXVAV.idf.
[10] https://github.com/ugr-sail/sinergym/blob/v2.3.2/sinergym/data/buildings/2ZoneDataCenterHVAC_wEconomizer.idf.



(a) Linear Reward



(b) Exponential Reward

Fig. 3. Performance on 5-Zone Hot Weather.

total HVAC electricity demand rate, current hour, current day, current month and year.

- Data Centre: Site outdoor air drybulb 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, facility total HVAC electricity demand rate, current hour, current day, current month and year.
- The data centre is split into two zones, east and west. The following inputs are common for both the zones.
  - 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 and people air temperature.

For both environments, the action spaces consist of the heating and cooling setpoints, which are controlled by the RL agent. Table 1 summarises the environments used, their weather type and their observation and action space shapes. The following describes our experiment scenarios:

- Pre-train - We pre-train our base agent on the 5-Zone Hot weather environment.
- Finetuning - We consider two scenarios for finetuning:
  1. 5-Zone Cool weather environment - Same building different weather.
  2. Data Centre Hot weather environment - Different building same weather.

The EnergyPlus weather data used to simulate hot weather is from Davis-Monthan AFB, Arizona, USA, while the cool weather is from Port Angeles, Washington, USA. Each episode in the simulation lasts for a period of one year. An episode of one year contains 35,040 15-minute timesteps. After each episode, we measure four Key Performance Indicators (KPIs):

Table 1
Summary of the environments.

Environment	Weather	Observation Space Shape	Action Space Shape
5Zone	Hot	20	2
5Zone	Cool	20	2
Data Centre	Hot	29	2

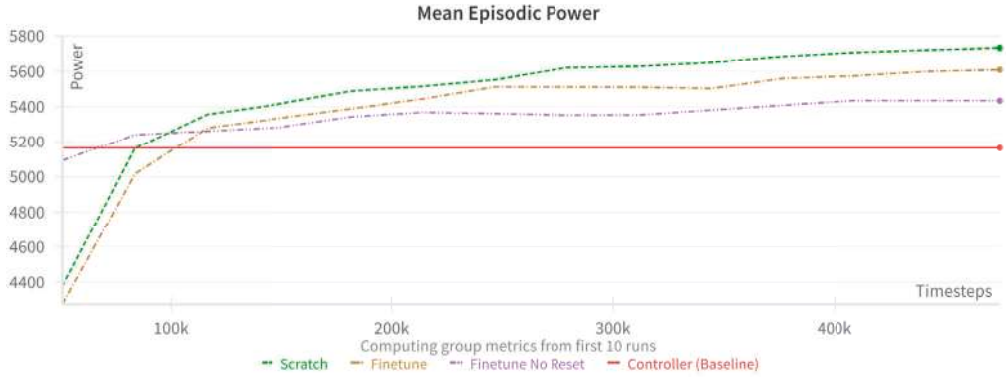
- Mean Reward: Average rewards received per step in the episode.
- Comfort Violation Time (%): Percentage of time the temperature has been beyond the bounds of the set comfort temperature ranges.
- Mean Comfort Penalty: Average of the comfort penalties from the reward component per step in the episode.
- Mean Power: Average power consumption per step in the episode.

4.2. Rewards

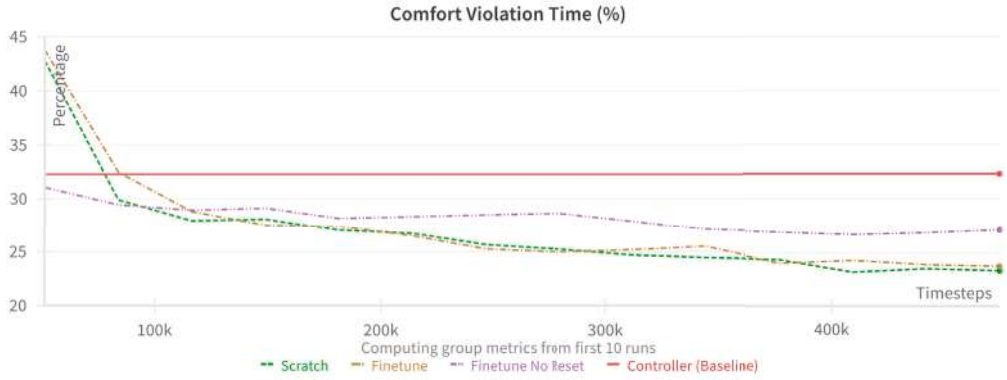
The objective of the Deep RL agent is to minimise energy consumption while ensuring that the temperature lies within the given comfort temperature range. The objective function is a weighted sum of the energy consumption and thermal discomfort after normalising. We consider two types of reward functions:

- Linear Reward - Energy consumption and discomfort are weighted and added.

$$R = -\omega \times \lambda_P \times P_t - (1-\omega) \times \lambda_T \times (|T_t - T_{up}| + |T_t - T_{low}|) \quad (7)$$



(a) Mean Episodic Power



(b) Comfort Violation Time

Fig. 4. KPIs for PPO with Linear Rewards on 5-Zone Cool Weather.

- Exponential Reward - Energy consumption and exponential discomfort are weighted and added.

$$R = -\omega \times \lambda_P \times P_t - (1-\omega) \times \lambda_T \times \exp(|T_t - T_{up}| + |T_t - T_{low}|) \quad (8)$$

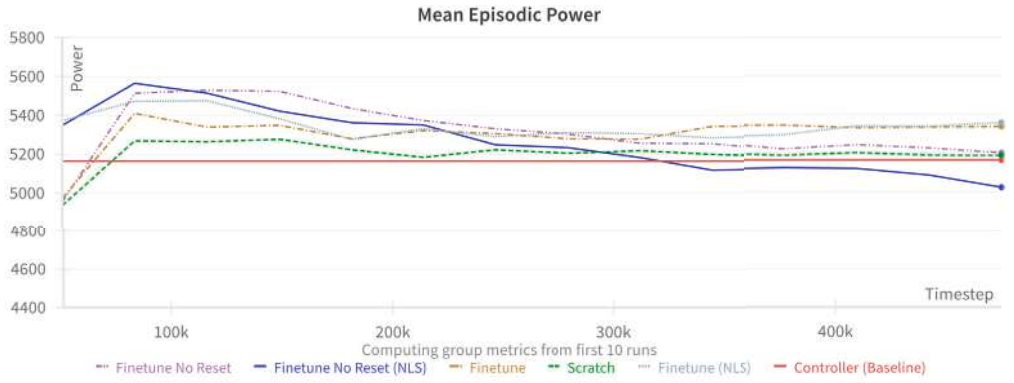
where, P_t is the power consumption, T_t is the current indoor temperature, ω is weight assigned to power consumption and λ_P and λ_T are scaling constants for power and temperature respectively. Discomfort is calculated as the absolute difference between the current temperature and comfort range (T_{up} and T_{low}). If the temperature is inside that range, the discomfort is 0. In our setup, energy and comfort penalties are equally weighted. We include the exponential reward function to evaluate whether changing the discomfort calculation from a linear difference to an exponential difference would increase the task difficulty. Additionally, we believe an exponential reward could enforce policies that reduce comfort violations while increasing power consumed. Hence, to monitor such behaviour, we track the differences in performance with respect to other KPIs like comfort violation time and power consumption.

4.3. Training setup

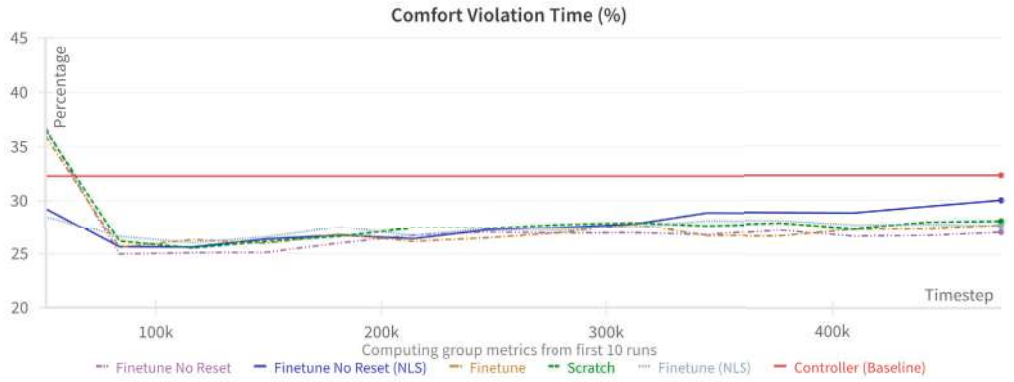
The Deep RL algorithms are implemented with the help of CleanRL [28]. The default hyperparameters included with CleanRL were used for the PPO and SAC algorithms, and they are given in Table 2. The Core architecture that is pre-trained and fine-tuned consists of three linear layers of size 64, 128 and 64. The input and output layers will depend upon the dimensionality of the observation and action spaces as described previously in Section 3. The ReLU activation function is used after every layer, barring the output layer. An RL agent is trained on the 5-Zone Hot environment using this setup. We then fine-tune the model on the 5-Zone Cool and Data Centre Hot environments. This procedure is carried out for both environments. The reward functions used for

Table 2
PPO and SAC Hyperparameters.

Algorithm	Hyperparameter Name	Value
PPO	Total Timesteps	500,000
	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
SAC	Total Timesteps	500,000
	Buffer Size	1e6
	Gamma	0.99
	Target Smoothing Coefficient	0.005
	Batch Size	256
	Learning Starts	5e3
	Policy Learning Rate	3e-4
	Q-network Learning Rate	1e-3
	Policy Update Frequency	2
	Target Update Frequency	2
	Entropy Regularisation Coefficient	0.2
	Autotune Entropy Coefficient	True
	Exploration Noise	0.1
	Noise Clip	0.5



(a) Mean Episodic Power



(b) Comfort Violation Time

Fig. 5. KPIs for SAC with Linear Rewards on 5-Zone Cool Weather.

training are given in Section 4.2. The learning curves for both the PPO and SAC algorithms are compared to the Rule-Based Controllers (RBC) provided by Sinergym.[11] The rules for the RBC for both the 5Zone and data centre are defined in Algorithms 1 and 2, respectively.

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
```

5. Results

5.1. Pre-training

We first begin by evaluating the performance of the algorithms on the 5-Zone Hot Weather simulation. The Core network from the trained agents in this environment will be used for transfer to the 5-Zone Cool and Data Centre Hot environments. The performance of the algorithms can be assessed from the learning curves depicted in Fig. 3. To be specific, Fig. 3a shows the mean reward per episode when using the linear reward function, while Fig. 3b shows the mean reward per episode when using the exponential reward function.

Algorithm 2 Rule for RB Controller for Data Centre.

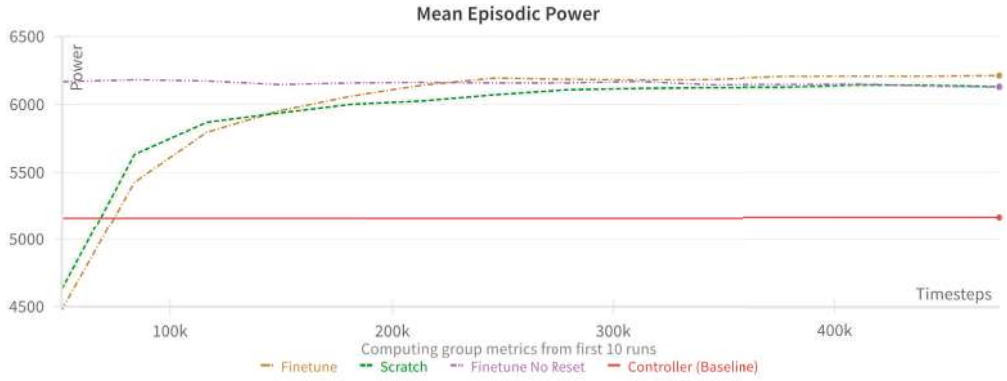
```
lower_limit ← 18
upper_limit ← 27
for each step in environment do
    mean_temp ← mean(west_zone_air_temp, east_zone_air_temp)
    if mean_temp < lower_limit then
        heat_setpoint ← heat_setpoint + 1
        cool_setpoint ← cool_setpoint + 1
    else if mean_temp > upper_limit then
        heat_setpoint ← heat_setpoint − 1
        cool_setpoint ← cool_setpoint − 1
    end if
end for
```

5.2. Fine-tuning on 5-zone cool

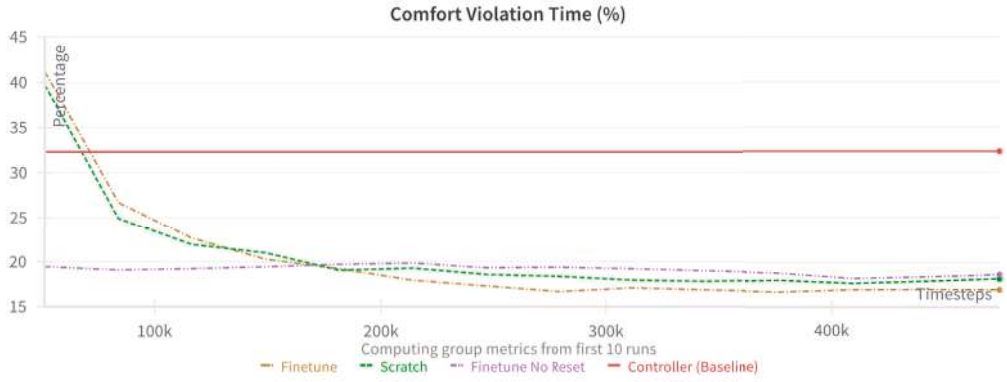
After we pre-train an agent in the Hot environment, we fine-tune it on a 5-Zone building in a Cool environment. Here, the two buildings have the same observation and action shapes. Hence, it is not necessary to re-initialise (reset) the input and output layers for transfer. But for comparison, we carry out three types of experiments:

1. Directly transfer the pre-trained agent without the input and output layer reset.
2. Transfer the Core of the pre-trained agent with the input and output layer reset.
3. Directly train an agent on the environment from scratch.

[11] https://github.com/ugr-sail/sinergym/blob/v2.3.2/sinergym/utils/controllers.py.



(a) Mean Episodic Power



(b) Comfort Violation Time

Fig. 6. KPIs for PPO with Exponential Rewards on 5-Zone Cool Weather.

Table 3
Performance Difference (in %) with Linear Rewards on 5-Zone Cool Weather.

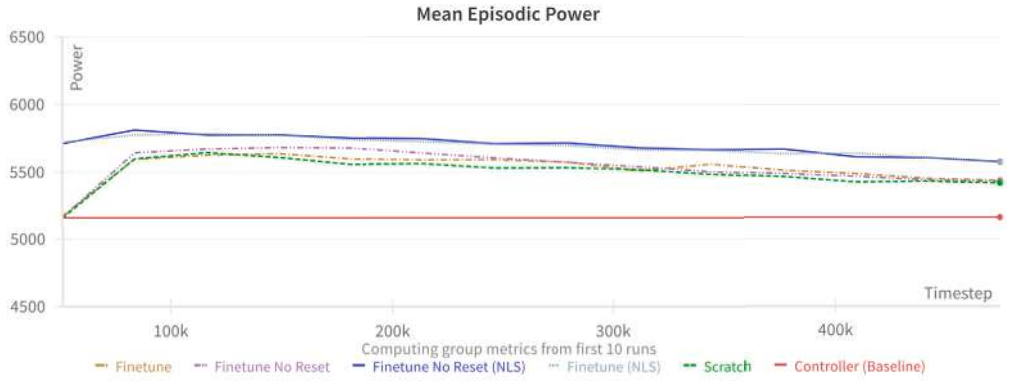
Algorithm	Methodology	Scratch		RB Controller	
		≈ 250k steps	≈ 500k steps	≈ 250k steps	≈ 500k steps
PPO	Scratch	—	—	-4.4	-2.4
	No Reset	-6.5	-5.4	-11.2	-8.0
	Reset	-0.3	-0.2	-4.8	-2.6
SAC	Scratch	—	—	10.2	7.9
	No Reset	-1.2	2.6	9.0	10.3
	No Reset with No Learning Starts	0.06	1.1	10.2	8.9
	Reset	0.4	0.4	10.6	8.3
	Reset with No Learning Starts	-3.5	-2.1	7.0	5.9

5.2.1. *Linear rewards*

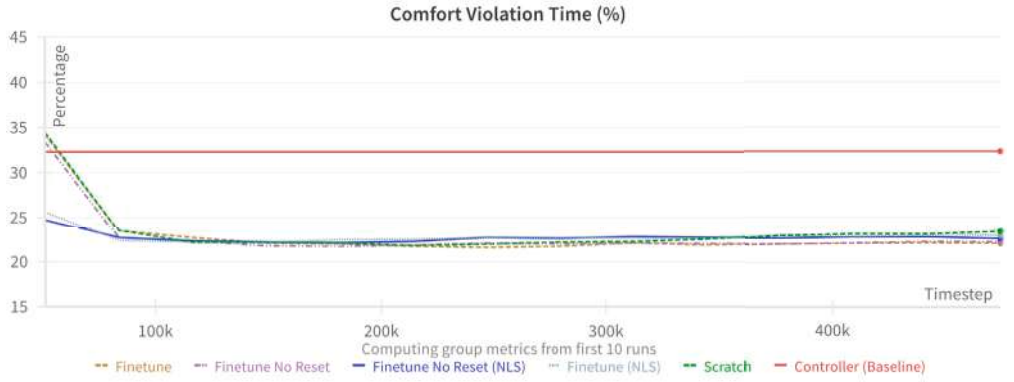
Table 3 shows the performance difference with respect to scratch and the RB Controller at the 250k timestep and 500k timestep mark when using linear rewards (Equation (7)). From the table, we can see that the PPO algorithm is unable to learn a desirable policy. We also see that for the algorithm, transferring network weights damages the learning process, resulting in poorer policies than when trained from scratch. On the other hand, the SAC algorithm, in general, performs better than PPO. It also outperforms the RB Controller by around 7% to 10%. However, when comparing the different weight-transferring methodologies with scratch, we see that not resetting the input and output layer weights has better performance. Figs. 4 and 5 show that the two algorithms have completely different priorities for maximising their rewards. Over the training period, the PPO algorithm aims to minimise comfort violations, which, in contrast, results in utilising more power.

5.2.2. *Exponential rewards*

Table 4, shows the performance difference with respect to scratch and the RB Controller at the 250k timestep and 500k timestep mark when using exponential rewards. Unlike the results we saw in Section 5.2.1, the PPO significantly outperforms the RB Controller. Resetting the input and output layer weights also outperforms training from scratch. From Fig. 6, we can see that the amount of power consumed and the comfort violations stabilise over the training period. However, since the exponential reward function focuses on occupant comfort when comparing Figs. 4 and 6, we can see that it results in



(a) Mean Episodic Power



(b) Comfort Violation Time

Fig. 7. KPIs for SAC with Exponential Rewards on 5-Zone Cool Weather.

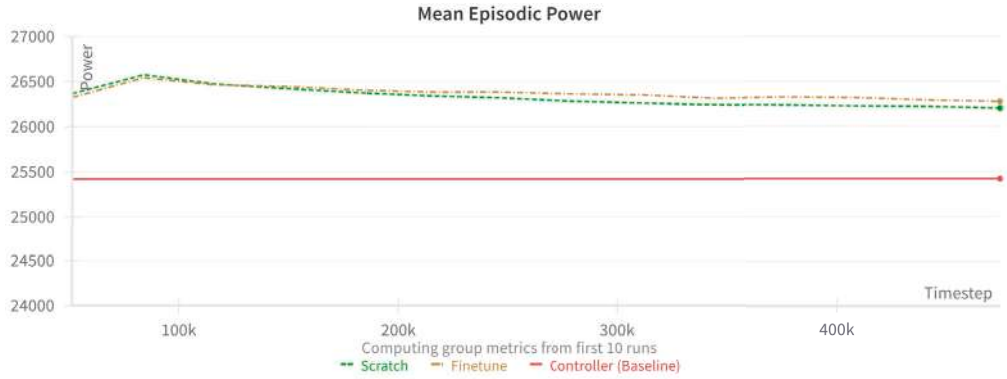
Table 4
Performance Difference (in %) with Exponential Rewards on 5-Zone Cool Weather.

Algorithm	Methodology	Scratch		RB Controller	
		≈ 250k steps	≈ 500k steps	≈ 250k steps	≈ 500k steps
PPO	Scratch	—	—	37.8	40.1
	No Reset	-1.0	-4.5	37.2	37.4
	Reset	4.3	3.5	40.5	42.3
SAC	Scratch	—	—	43.5	42.1
	No Reset	-0.6	3.5	43.2	44.1
	No Reset with No Learning Starts	-3.7	0.1	41.4	42.2
	Reset	0.7	4.1	44.0	44.5
	Reset with No Learning Starts	-3.5	0.3	41.5	42.3

more power usage while reducing occupant discomfort. Similarly, the SAC algorithm outperforms the RB Controller as well. While there is no significant improvement in performance around the halfway mark compared to scratch, all the weight-transferring methodologies improve their performance over the training period, eventually outperforming scratch. Fig. 7 shows the KPIs for the SAC algorithm, which, on the other hand, manages to learn a policy that reduces power usage (Figs. 5a and 7a) while stabilising occupant comfort (Figs. 5b and 7b). As a result, it is able to outperform the PPO algorithm easily. When we aggregate the results for each experimental variant of the PPO algorithm and change the reward from linear to exponential, we see around a 10% increase in mean power for around a 28% decrease in mean comfort violation time, while for the SAC algorithm, we observe an increase of around 1.63% and a decrease of around 18% in the mean power consumed and the comfort violation times, respectively.

5.3. Fine-tuning on data centre hot

We cannot directly transfer the model weights for the data centre task since the dimensionality of the observation space is different compared to the 5-Zone environments. Hence, for all transfer learning experiments in this task, we must re-initialise (reset) the input and output layers based on the dimensionality. Apart from this, we conduct the same experiments described in Section 5.2.



(a) Mean Episodic Power

(b) Comfort Violation Time

Fig. 8. KPIs for PPO with Linear Rewards on Data Centre Hot Weather.

Table 5
Performance Difference (in %) with Linear Rewards on Data Centre Hot Weather.

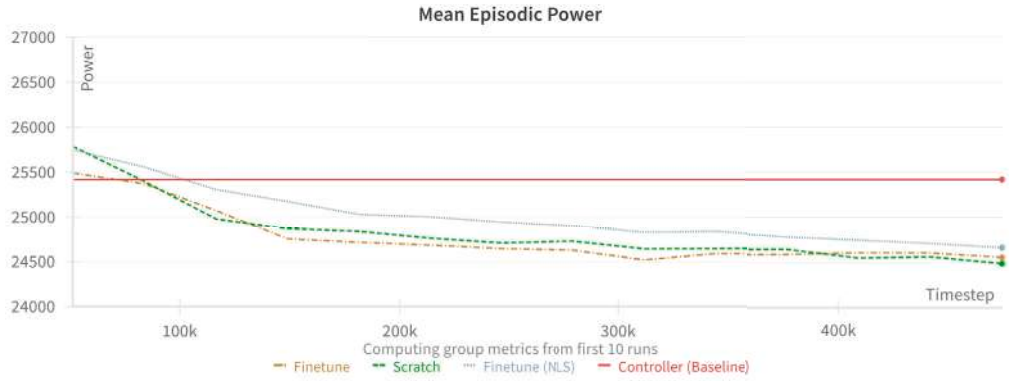
Algorithm	Methodology	Scratch		RB Controller	
		≈ 250k steps	≈ 500k steps	≈ 250k steps	≈ 500k steps
PPO	Scratch	—	—	-3.6	-3.0
	Reset	-0.3	-0.3	-3.9	-3.4
SAC	Scratch	—	—	0.5	1.0
	Reset	0.2	0.3	0.7	1.3
	Reset with No Learning Starts	-0.7	-0.7	-0.1	0.3

5.3.1. Linear rewards

The performance of the PPO algorithm in this environment is similar to its performance in Section 5.2.1. Resetting the input and output layer weights does not outperform training from scratch. Additionally, the PPO algorithm fails to improve upon the RB Controller. However, the learning curves (see Fig. 8) show that the PPO algorithm is able to reduce comfort violation time to almost zero while reducing the amount of power required to achieve this. On the other hand, resetting the input and output layer weights works for the SAC algorithm, outperforming scratch and the RB Controller by a slight margin. Conversely to PPO, however, the SAC algorithm reduces power consumption, which greatly affects comfort violation, increasing it to around 12% per episode for the different training methodologies (see Table 5).

5.3.2. Exponential rewards

Resetting the input and output layer weights gives a minor improvement over training from scratch for the PPO algorithm. From Table 6, we see that despite this improvement, it does not do better than the RB Controller. From Figs. 8 and 10, we can infer that there is no major difference between the two reward functions. While resetting the input and output layer weights for the SAC algorithm performs the worst at the halfway mark, it eventually outperforms scratch and the RB Controller. On the other hand, the reset with no learning starts methodology consistently outperforms scratch, ultimately outperforming the RB Controller as well. Unlike the PPO, the SAC algorithm cannot reduce the comfort violation time to zero. However, it follows a similar trend (seen in Figs. 5, 7, 9 and 11) where mean episodic power reduces over time while comfort violation time increases when using linear rewards, but using exponential rewards reduces the mean episodic power over the training period while the comfort violation time remains stable. When we aggregate the results for each experimental variant of the PPO algorithm and change the reward from linear to exponential, we see around a 0.1% decrease in mean power for around a 93.9% decrease in mean comfort violation time, while for the SAC algorithm, we observe an increase of around 2.1% and a decrease of around 89.1% in the mean power consumed and the comfort violation times, respectively.



(a) Mean Episodic Power

(b) Comfort Violation Time

Fig. 9. KPIs for SAC with Linear Rewards on Data Centre Hot Weather.

Table 6
Performance Difference (in %) with Exponential Rewards on Data Centre Hot Weather.

Algorithm	Methodology	Scratch		RB Controller	
		≈ 250k steps	≈ 500k steps	≈ 250k steps	≈ 500k steps
PPO	Scratch	—	—	-3.5	-3.1
	Reset	0.1	0.07	-3.3	-3.0
SAC	Scratch	—	—	-0.5	0.07
	Reset	-0.2	0.07	-0.7	0.1
	Reset with No Learning Starts	0.3	0.07	-0.1	0.1

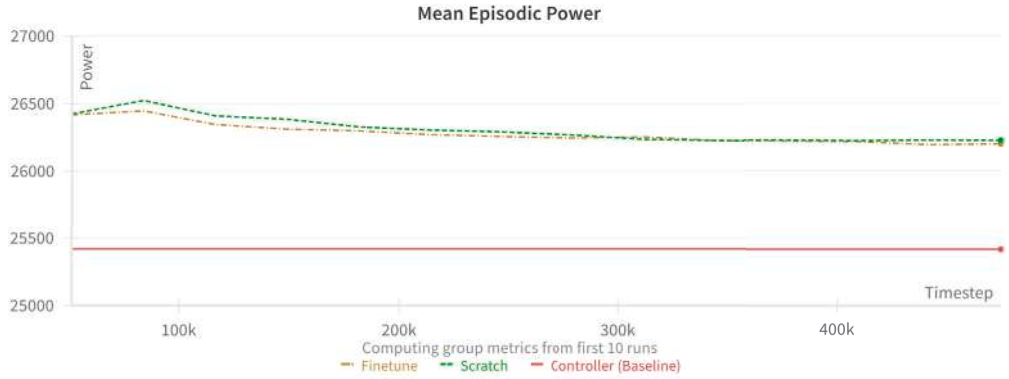
6. Discussion

In our experiments, we evaluated two RL algorithms for transferring agents across buildings with varying weather conditions and schematics. The initial phase involved pre-training a model on the 5-Zone in Hot Weather using both algorithms to establish a foundational model, which was subsequently fine-tuned for specific tasks, namely the 5-Zone in Cool Weather and the Data Centre in Hot Weather. These experiments incorporated two different reward functions - linear and exponential.

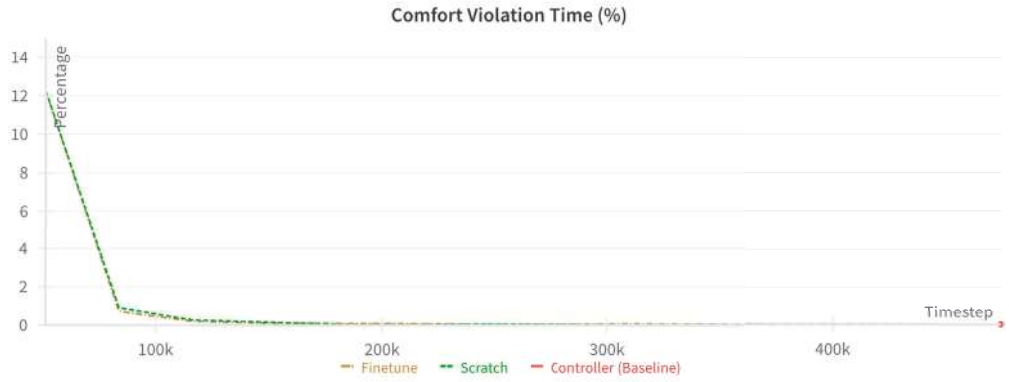
For the 5-Zone Cool weather environment, we have seen that the PPO algorithm struggles to learn an effective policy, particularly when network weights are transferred, resulting in suboptimal policies compared to training from scratch. Conversely, the SAC algorithm consistently outperforms the PPO algorithm and the RB Controller, demonstrating a performance improvement ranging from 7% to 10%. In the case of exponential rewards, the PPO algorithm notably improves over the RB Controller, in contrast to its performance with linear rewards. The SAC algorithm continues to outperform the RB Controller. Importantly, all weight-transferring methodologies exhibit improved performance over training, eventually surpassing the scratch baseline.

For the Datacenter Hot weather environment, the PPO algorithm shows similar performance seen for the 5-Zone Cool weather environment. When resetting the input and output layer weights, PPO fails to surpass training from scratch and does not outperform the RB Controller. However, the algorithm reduces comfort violation time to nearly zero while lowering power consumption. Conversely, resetting input and output layer weights for the SAC algorithm shows slight performance improvement over scratch and the RB Controller. SAC consistently reduces power consumption, impacting comfort violation. For both algorithms, resetting input and output layer weights offers marginal enhancement for PPO and a consistent outperformance trend for SAC. Linear rewards reduce mean episodic power for SAC and increase comfort violation time. Exponential rewards result in a reduction of mean power, while comfort violation time remains constant.

Thus, our findings indicate that the SAC algorithm consistently outperforms the PPO algorithm across both tasks and reward functions,



(a) Mean Episodic Power



(b) Comfort Violation Time

Fig. 10. KPIs for PPO with Exponential Rewards on Data Centre Hot Weather.

particularly when compared to the RB Controller. Biemann et al. [11] showed similar results where they trained different on-policy and off-policy algorithms and found that the SAC algorithm outperforms other RL algorithms. We have shown that this is also true when performing transfer learning, where the input and output layers are adjusted for the change in the observation and action spaces. Experience replay and entropy regularisation may explain why an off-policy algorithm like SAC outperforms an on-policy algorithm like PPO. Compared to the PPO algorithm, the SAC algorithm prefers reducing energy at a small cost of increasing comfort violations. However, when changing the reward function to exponential, we notice the same trend for power consumption while reducing and stabilising the comfort violations when using the SAC algorithm.

In conclusion, resetting the input and output layers and using exponential rewards for the SAC algorithm seems more effective. It consistently outperforms PPO across various tasks and reward functions. However, the choice between linear and exponential rewards may depend on specific considerations, such as the trade-off between power consumption and comfort violation time.

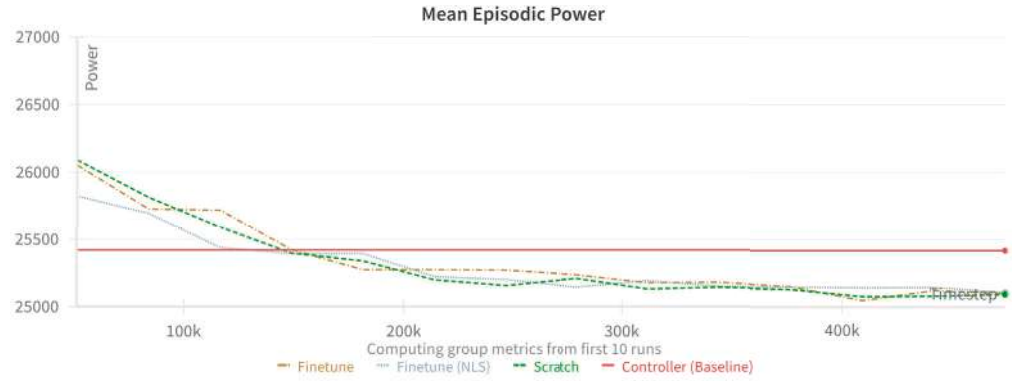
7. Conclusion and future work

In this paper, we evaluated a methodology using two different RL algorithms to transfer agents across buildings under different weather conditions and characteristics. We first pre-trained a model using the two algorithms to obtain a foundation model. This model was then fine-tuned on two different tasks - the 5-Zone in Cool Weather and the Data Centre in Hot Weather. We performed these experiments using two different reward functions - Linear and Exponential.

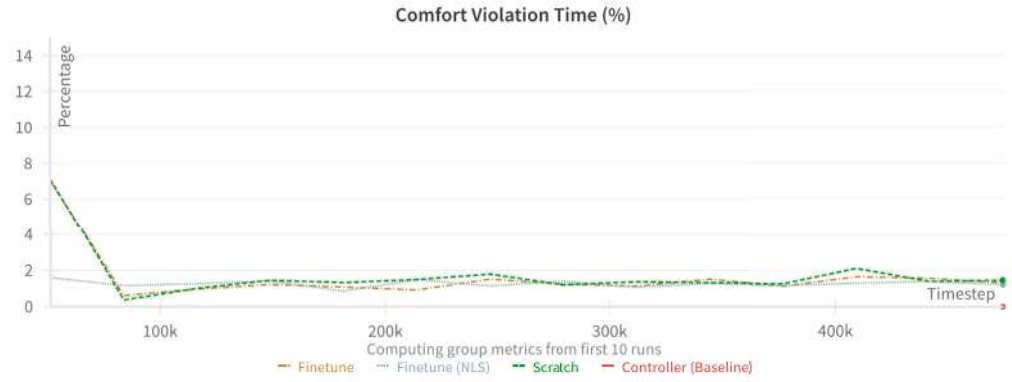
We showed that the SAC algorithm outperforms the PPO algorithm on both tasks using both reward functions when comparing it with respect to the RB Controller. When comparing the results with respect to training from scratch, the PPO algorithm does not provide an improvement. When we evaluate the SAC algorithm, we see that around the halfway mark (250k steps), it provides inconsistent improvement over training from scratch. However, over the entire training period, we see that the different transfer learning variants of the algorithm outperform training from scratch in general. This is true for both the simulation environments with both reward functions. In general, we see around 1% to 4% improvement in rewards.

Apart from the rewards, we also tracked other KPIs, like the mean power consumed per episode and the percentage of time spent outside the comfort temperature zone. From the evaluation for linear rewards, we see that the PPO algorithm, in general, tends to reduce comfort violation time while increasing the average power consumed, while the SAC algorithm shows opposite trends where it reduces the average power consumed while increasing the mean comfort violations. However, when we use the exponential reward function, we see that these curves stabilise for the PPO algorithm, while we observe a similar decreasing trend for the mean power consumed but with stabilised mean comfort violations for the SAC algorithm. Thus, the work furthers ongoing research in smart energy systems, where a pre-trained agent, over time, develops deep knowledge and understanding of the building under their control, effectively being able to reduce comfort violations and power consumed.

For future work, a methodology that can avoid any need for re-initialisation of the input or output layers will be considered. This could lead to useful information from different buildings and weathers being learnt by the agent, potentially developing smarter agents with little training required. Exploring imitation learning as a pre-training methodology could be useful when finetuning with reinforcement learning. This approach can potentially minimise the required



(a) Mean Episodic Power



(b) Comfort Violation Time

Fig. 11. KPIs for SAC with Exponential Rewards on Data Centre Hot Weather.

training and enhance overall performance. Additionally, the behaviour of an expert system like a rule-based controller can be utilised as training data to provide a performance boost, which could result in better and more efficient agents. State information data could also be used to train autoencoders, reducing the dimensionality of the observation space, enabling efficient representation learning and, thus, contributing to improved policy generalisation. Algorithmic performance could also be enhanced with hyperparameter tuning that could be performed with the help of hyperparameter sweeps or metaheuristics like genetic algorithms. There is also an opportunity to experiment with various configurations by adjusting reward weights associated with power consumption and comfort, thus highlighting the potential implications of different weightings. Finally, there is potential to enhance this approach in a multi-objective setting by separately optimising energy and comfort rewards, allowing for personalised weighting by occupants. With this, practicality and sustainability can be improved while exploring dynamic energy costs, varying with time or load magnitude, and integrating considerations like time-varying carbon intensity for total carbon emissions.

CRediT authorship contribution statement

Kevlyn Kadamala: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Des Chambers:** Writing – review & editing, Supervision, Resources, Project administration, Investigation. **Enda Barrett:** Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Investigation.

Declaration of competing interest

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

Data availability

The links to the repository and simulation library are provided in the paper.

Acknowledgement

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.

References

[1] Dmitrewski A, Molina-Solana M, Arcucci R. Cntrlda: a building energy management control system with real-time adjustments. Application to indoor temperature. Build Environ 2022;215:108938.
[2] Arroyo J, Manna C, Spiessens F, Helsen L. Reinforced model predictive control (rl-mpc) for building energy management. Appl Energy 2022;309:118346.
[3] Lissa P, Deane C, Schukat M, Seri F, Keane M, Barrett E. Deep reinforcement learning for home energy management system control. Energy AI 2021;3:100043.
[4] Devlin J, Chang M-W, Lee K, Toutanova K. Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
[5] Tan M, Le Q. Efficientnet: rethinking model scaling for convolutional neural networks. In: International conference on machine learning. PMLR; 2019. p. 6105–14.

[6] Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. J Mach Learn Res 2020;21:5485–551.
[7] Yuan L, Chen D, Chen Y-L, Codella N, Dai X, Gao J, et al. Florence: a new foundation model for computer vision. arXiv preprint arXiv:2111.11432, 2021.
[8] Wightman R. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. https://doi.org/10.5281/zenodo.4414861.
[9] 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; 2021. p. 319–23.
[10] Sutton RS, Barto AG. Reinforcement learning: an introduction. MIT Press; 2018.
[11] Biemann M, Scheller F, Liu X, Huang L. Experimental evaluation of model-free reinforcement learning algorithms for continuous hvac control. Appl Energy 2021;298:117164.
[12] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.
[13] Konda V, Tsitsiklis J. Actor-critic algorithms. Adv Neural Inf Process Syst 1999;12.
[14] Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning. PMLR; 2018. p. 1861–70.
[15] Haarnoja T, Zhou A, Hartikainen K, Tucker G, Ha S, Tan J, et al. Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905, 2018.
[16] 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. Springer; 2015. p. 3–19.
[17] Fang X, Gong G, Li G, Chun L, Peng P, Li W, et al. Deep reinforcement learning optimal control strategy for temperature setpoint real-time reset in multi-zone building hvac system. Appl Therm Eng 2022;212:118552.
[18] Zhu Z, Lin K, Jain AK, Zhou J. Transfer learning in deep reinforcement learning: a survey. IEEE Trans Pattern Anal Mach Intell 2023.
[19] Taylor ME, Stone P. Transfer learning for reinforcement learning domains: a survey. J Mach Learn Res 2009;10.
[20] Lissa P, Schukat M, Barrett E. Transfer learning applied to reinforcement learning-based hvac control. SN Comput Sci 2020;1:1–12.
[21] Zhang X, Jin X, Tripp C, Biagioni DJ, Graf P, Jiang H. Transferable reinforcement learning for smart homes. In: Proceedings of the 1st international workshop on reinforcement learning for energy management in buildings & cities; 2020. p. 43–7.
[22] Xu S, Wang Y, Wang Y, O'Neill Z, Zhu Q. One for many: transfer learning for building hvac control. In: Proceedings of the 7th ACM international conference on systems for energy-efficient buildings, cities, and transportation; 2020. p. 230–9.
[23] Wei T, Wang Y, Zhu Q. Deep reinforcement learning for building hvac control. In: Proceedings of the 54th annual design automation conference 2017; 2017. p. 1–6.
[24] Lissa P, Schukat M, Keane M, Barrett E. Transfer learning applied to drl-based heat pump control to leverage microgrid energy efficiency. Smart Energy 2021;3:100044.
[25] Fang X, Gong G, Li G, Chun L, Peng P, Li W, et al. Cross temporal-spatial transferability investigation of deep reinforcement learning control strategy in the building hvac system level. Energy 2023;263:125679.
[26] An Z, Ding X, Rathee A, Du W. Clue: safe model-based rl hvac control using epistemic uncertainty estimation. In: Proceedings of the 10th ACM international conference on systems for energy-efficient buildings, cities, and transportation; 2023. p. 149–58.
[27] 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; 2023. p. 242–65.
[28] Huang S, Dossa RFJ, Ye C, Braga J, Chakraborty D, Mehta K, et al. Cleanrl: high-quality single-file implementations of deep reinforcement learning algorithms. J Mach Learn Res 2022;23:1–18. http://jmlr.org/papers/v23/21-1342.html.