Continual Reinforcement Learning for HVAC Systems Control: Integrating Hypernetworks and Transfer Learning

Gautham Udayakumar Bekal
Mitchell, Enlyte
gauthambekal93@gmail.com

Ahmed Ghareeb
Department of Mechanical Engineering
College of Engineering, University of Kirkuk
aghareeb@uokirkuk.edu.iq

Ashish Pujari
Department of Mechanical Engineering
University of North Carolina at Charlotte
apujari1@charlotte.edu

March 2025

Abstract

Buildings with Heating, Ventilation, and Air Conditioning (HVAC) systems play a crucial role in ensuring indoor comfort and efficiency. While traditionally governed by physics-based models, the emergence of big data has enabled data-driven methods like Deep Reinforcement Learning (DRL). However, Reinforcement Learning (RL)-based techniques often suffer from sample inefficiency and limited generalization, especially across varying HVAC systems.

We introduce a model-based reinforcement learning framework that uses a Hypernetwork to continuously learn environment dynamics across tasks with different action spaces. This enables efficient synthetic rollout generation and improved sample usage.

Our approach demonstrates strong backward transfer in a continual learning setting: after training on a second task, minimal fine-tuning on the first task allows rapid convergence within just 5 episodes—outperforming Model Free Reinforcement Learning (MFRL) and effectively mitigating catastrophic forgetting.

These findings have significant implications for reducing energy consumption and operational costs in building management, thus supporting global sustainability goals.

Keywords: Deep Reinforcement Learning; HVAC Systems Control; Hypernetworks; Transfer and Continual Learning; Catastrophic Forgetting

Nomenclature

HVAC Heating, Ventilation, and Air Conditioning

DRL Deep Reinforcement Learning

RL Reinforcement Learning

TL Transfer Learning

CL Continual Learning

MFRL Model-Free Reinforcement Learning

MBRL Model-Based Reinforcement Learning

SAC Soft Actor Critic

MPC Model Predictive Control

BOPTEST Building Optimization Testing Framework

BHHP BOPTEST Hydronic Heat Pump

ML Machine Learning

NN Neural Network

RL Agent Reinforcement Learning Agent (Actor-Critic or Q-learning based)

Hypernet Hypernetwork (generates weights for a target network)

Catastrophic Forgetting Rapid loss of previously learned knowledge when adapting to new tasks

Dyna Integrated architecture combining learning, planning, and reacting

Symbols and Notation

γ Discount factor for future rewards, $0 < \gamma \leq 1$

α Learning rate or regularization coefficient (context-dependent)

β Regularization term coefficient used in hypernetwork training

s Environment state (for example, zone temperature or time)

a Action selected by the RL agent (such as a discretized control input)

r Reward returned by the environment for a given s-a pair

s' Next state resulting from the transition (s, a)

π_θ Policy network with trainable parameters θ

Q_ϕ State-action value (critic) network with parameters ϕ

H_φ Hypernetwork with parameters φ

T_δ Target network (e.g., a dynamics or reward model) generated by H_φ, with parameters δ

$\mathcal{M}_\alpha$ Real data memory buffer

$\mathcal{M}_\beta$ Synthetic data buffer (model-generated samples)

$\mathcal{M}_\gamma$ Hypernetwork training data buffer

K_i Number of training episodes or iterations for task i

task-id One-hot identifier specifying the current task

layer-id Layer index used by H_φ to generate target network parameters

$P(s' \mid s, a)$ Transition dynamics (approximated by the hypernetwork)

η Learning rate for the SAC or hypernetwork optimizer

1 Introduction

Heating, Ventilation, and Air Conditioning (HVAC) systems have a major impact on energy use and consumption and are essential to maintaining indoor comfort, air quality, and overall building function. With increased efficiency, they have the potential to save 20–40% of the total energy consumed in commercial buildings and up to 48% in residential buildings. In a particular environment, HVAC systems can account for 60–70% of a building energy demand, resulting in increased operating costs and greenhouse gas emissions. In addition to lowering costs and having a positive environmental impact, increased HVAC efficiency also increases occupant productivity and comfort. Innovative control strategies, such as transfer learning (TL) and reinforcement learning (RL), present viable options to effectively and sustainably maximize HVAC performance and energy efficiency.

Since the advent of deep learning a variety of RL based approaches have been developed to solve HVAC control-related issues. However, RL-based approaches primarily have two challenges, sample inefficiency and generalizability. One of the ways to handle sample inefficiency is using model-based reinforcement learning [1] which involves explicitly creating an approximate model of the environment and letting the RL system learn from both the actual and the surrogate environment. The second problem of generalization is usually dealt with using the transfer learning approach, which has been widely used in natural language processing and computer vision [2]. In the case of RL for HVAC systems, this involves training an agent in a specific HVAC simulator and then varying the building parameters during the fine-tuning phase. Unfortunately, such approaches have a severe shortcoming in the Machine Learning (ML) and

RL domains, since the parameters of the neural network are optimized to minimize the loss on the latest task at hand, which means the model will perform extremely poorly in the older tasks. This problem is well documented in machine learning literature and is called the Catastrophic Forgetting problem [3]

In order to mitigate both of the above issues, we combine Soft Actor Critic (SAC) based RL algorithm with Hypernetworks. Crucially, the SAC utilizes vanilla neural nets, and the model of the environment is learnt using Hypernetworks. We carry out continual transfer learning using various action spaces across the tasks and the Hypernets continually learn these different configurations of the environment. The hypernet then augments the data set of real samples with synthetic data samples to speed up the training of RL agent under Dyna-style [4] model-based RL [5].

2 Related Work

DRL has been proven to be a powerful technique in traditional ML experiments such as games [6]. Its applications have now been impacted in other domains such as tools for optimizing control systems, particularly in HVAC applications [7]. Traditional control methods, such as Model Predictive Control (MPC) [8] and other classical approaches, often require extensive manual tuning and may not adapt easily to changing conditions or uncertainties. It usually struggles with efficiency and adaptability, prompting researchers to explore the potential of RL to enhance energy conservation and indoor comfort. In this context, Gao and Wang [9] investigate HVAC system RL techniques that are model-based and model-free. They compare the computing needs and efficiency of both methods using the Building Optimization Testing (BOPTEST) framework. Their implementation combines model-based techniques and sophisticated RL methods such as Dueling Deep Q-Networks and Soft Actor-Critic. Comparing RL controllers to conventional approaches, they find that all of the approaches improve interior temperature regulation and save operating costs. Remarkably, despite initial model flaws, model-based reinforcement learning outperforms model-free RL with shorter training times.

More recently, Transfer Learning (TL) has been extensively studied across various domains but has yet to be thoroughly investigated in the HVAC systems sector. For instance, a unique TL framework is proposed by Coraci et. al [10] in order to enhance the scalability of DRL controllers in buildings that include integrated energy systems. The authors introduced a heterogeneous TL technique to overcome the scalability and generalization limitations of DRL. This approach is tested in Energyplus simulations with a range of target buildings, concentrating on a DRL policy for controlling a chiller in conjunction with thermal energy storage. Model slicing, imitation learning, and fine-tuning are used in the TL technique to address a variety of building situations and characteristics. The proposed method effectiveness and applicability in the test environments demonstrated that it lowers costs of electricity and temperature violation rates while simultaneously improving self-sufficiency and self-consumption [11].

In prior work, researchers explored techniques to accelerate DRL for energy management by leveraging learned hidden layers from a pretraining phase for fine-tuning. Since states and actions can vary across environments, some re-engineering is required.

While the presented studies show the effectiveness of TL in enhancing HVAC control, they lack extensive investigation into Continual Learning (CL) setup where the system needs to perform well on both forward and backward transfer. Additionally, the literature often presents transfer learning across changing state space, whereas here we present transfer learning across action space. We also observe that despite presenting valuable RL frameworks, crucial parts are sometimes missing, making it difficult to reproduce, validate, or build upon the work.

This study fills a significant gap by presenting a novel approach to continual learning RL through the use of an environment model, which improves generalization. It provides a comprehensive framework for transferring knowledge between environment configurations differing in action space and thus, addressing limitations in existing literature, and enhancing the effectiveness of the RL technique.

Kadamala et. al [12] proposed a modular neural net architecture where the parameters for the hidden layers are transferred from the pre-training environment to the fine-tuning environment. The assumption is that the RL agent trained on the first environment should have information useful for the second environment, which is stored in the parameters. However, they were only able to get a marginal improvement of 1 to 4%. This confirms our analysis that transfer learning of direct RL agents leads to minimal improvement.

Xu et. al [13] used a different neural net architecture to carry out transfer learning across similar environments. However, akin to the aforementioned study, we see that transfer learning is at the RL agent level instead of the environment level.

Gao and Wang [9] analyzed model-based and model-free RL using actor-critic and Q-learning approaches. Here, they show the sample efficiency problem being mitigated compared to the vanilla version of the RL algorithm. However, the authors carry out only a single experiment. We extend their approach by carrying out detailed experiments for two environments and also analyze the backward transfer performance by setting up continual learning setup.

Our approach is quite generic and even though we have used SAC in our experiments it can be modified to handle any standard RL algorithm.

Catastrophic forgetting has been extensively studied in ML literature and we point the interested reader to go through them [14]. Crucially, the problem of Catastrophic forgetting is a tradeoff between the performance of forward task and backward task in a continual learning setup.

The following points highlight our contribution to the literature:

- By using Hypernets for learning the environment models, we create a world model, that acts as a substitute for the real environment and learns the dynamics of environment configurations differing in action spaces.

- We carry out CL at the environment model level and thus utilize MBRL to improve sample efficiency across tasks.
- Using the regularization term in the training of Hyperent, we significantly reduce the problem of catastrophic forgetting.

3 Preliminaries

In this section, we review the key concepts that form the foundation of our work. Each concept is introduced in a way that motivates the need for the next, creating a logical progression toward the core ideas of our approach.

3.1 BOPTEST Simulator

BOPTEST [15] is a standardized simulation framework for evaluating advanced control algorithms in building energy systems. It provides a set of virtual building models with realistic HVAC dynamics, weather data, internal gains, and occupant behavior. BOPTEST has several use cases to choose from, and for this work, we choose Boptest Hydronic Heat Pump (BHHP) since this has been utilized in the existing literature and thus is easier to compare to. For our work, we choose the time, current room temperature, and dry bulb temperature forecast, as input states to SAC. We also add cooling and heating setpoints as inputs to the Hypernetwork and discard time to improve reward prediction. Adding setpoints to SAC had minimal impact hence discarded from SAC. We change the action space as per the task at hand we are training the model on. We define the collection of action 1 as the heat pump modulating signal, action 2 as the heat pump evaporator fan, and action 3 as the emission circuit pump as the action space. Here, actions used to control the environment temperature are discrete, and hence actions generated by the RL agent need to be discretized. The reward is generated by the BOPTEST environment and it involves reducing the thermal discomfort of the occupants in the room. You can refer to the tables 1, 2 and 3 for more detals.

3.2 Soft Actor-Critic

Soft actor-critic is a state-of-the-art RL which directly optimizes the policy to maximize long-term discounted rewards [16] The actor takes the state as input and generates action to maximize reward, and there is a trade-off between exploration and exploitation. The critic uses bootstrapping to estimate the state-action value Q(s,a) which the actor takes as input. Concretely to train SAC we need tuples of (state, action, next-state, reward).

3.3 Model Based Reinforcement Learning

The SAC algorithm described above relies on sampling tuples of (s, a, s', r) for training. However, if we had a statistical model such as a neural net that has learnt to predict s' and r given s, a as inputs then the RL algorithm can query the environment to obtain these generated tuples and train the model. However, the core issue is the predicted s' and r should be accurate and unbiased else it can destabilize the RL algorithm. Hence, in Model Based Reinforcement Learning (MBRL) we train both an RL agent and a model of the environment simultaneously which greatly improves sample efficiency. In our use case we use Hypernets [17] to learn the dynamics of the environment and predict the next state and reward. We follow the Dyna-style model-based RL framework here.

3.4 Transfer Learning

TL involves two stages of training, where in stage 1 we train on task 1 and in stage 2 we train on task 2, using the pre-trained weights after task 1 as the initial weights for the model. This helps in speeding up the training process for task 2. TL has been utilized successfully in both Natural Language Processing (NLP) and Computer Vision (CV).

3.5 Continual Learning

Unlike in TL, in CL [18] we train on multiple tasks in a sequential fashion. Crucially, the difference is in CL we expect the learnt model to perform well on all the tasks and not just on the latest fine-tuned task as in TL. This makes CL much more difficult since both forward and backward transfer performance needs to be satisfactory. Thus a major theme in CL is to mitigate catastrophic forgetting in older tasks, and thus begs the need of sophisticated techniques such as [19].

3.6 Hypernetworks

A Hypernetwork [17] is a neural network that generates the weights of another neural network. Formally, given a Hypernetwork H_ϕ parameterized by ϕ, and a target network f_θ, the Hypernetwork produces $\theta = H_\phi(z)$ where z is an input (e.g., a task embedding). This allows parameter generation conditioned on task-specific or context-specific information. In our experiments, we generate task-specific embedding as one hot encoded value and add Gaussian noise to generate an ensemble of target weights. This acts similarly to Bayesian neural networks and reduces systemic bias which can severely impact the performance of model-based RL.

4 Methodology

We train our model based RL in Dyna-style fashion. Here, we simultaneously train both the SAC-based RL agent along with Hypernets at each time step. Here the SAC-based RL agent learns to maximize the rewards, whereas the hypernet is learning the dynamics of the environment. The RL agent uses both real and synthetic data for training. Initially, the hypernet predictions are poor but after some time steps, the model starts generating high-quality predictions which the RL agent consumes.

The following section provides a detailed explanation of the hypernetwork training procedure.

4.1 Task Setup

In our continual learning setup, we have 3 tasks. Task 1 uses only a single action (Heat pump modulating signal) as the control input to the BHHP environment. Here, the other two actions (heat pump evaporator, emission circuit pump) to control the room temperature are in default setting generated by BHHP itself.

In Task 2, the actor generates all 3 actions output to have a more fine-grained control of the room temperature.

In Task 3 the actor is in a similar setup as in Task 1, generating only a Heat pump modulating signal, for controlling the environment and the remaining two actions are the default values from BHHP.

The Hypernetwork always receives all three actions along with the state as input and generates the next state and reward as output. The specific combination of actions—i.e., whether two actions are produced by the actor policy or are default values from the BOPTEST environment—varies depending on the task. This variation in action configuration across tasks establishes a transfer learning setup, where the Hypernetwork must generalize across different control schemes.

4.2 Training Soft Actor Critic

For all three stages, we train soft actor-critic from scratch. Readers can go through [16] for more details. The difference is, in MBRL we train the SAC algorithm using both real and synthetic samples of (s, a, s', r).

4.3 Training Hypernetworks

The Hypernet takes in task-id and layer-id as input and generates parameters for the target network. The forward pass of the target network is same as that of a regular neural net and takes in current state and action as input and generates next sate or reward as the output. Specifically, in our experiments we have two pairs of Hypernet-target-network setups. The first one takes the state-action pair as input and generates the next state as the output. The other one takes in the same state action as the input and generates a reward as the output. Gaussian noise is added to the task-id before feeding to Hypernet, so that Hypernet generates a distribution of target parameters instead of a single parameter, thus reducing bias which may cripple the RL agent. The Hypernet generates the parameters of the target network on a layer-by-layer basis which reduces the parameters needed to train the Hypernet. It is important to remember that only the Hypernet is trained using backpropagation. The target network is only used to generate the predictions at the output.

4.4 Mitigating Catastrophic Forgetting in Hypernets

Among various methodologies to mitigate catastrophic forgetting in neural nets, Hypernets [20] offer superior solutions in terms of performance and memory consumption. In our scenario, Hypernets will dynamically generate parameters conditioned on task-id and layer-id. This means that Hypernets are trained on task 2, it will rewrite the weights of task 1 due to catastrophic forgetting, which leads to performance degradation of task 1. To prevent this, Hypernet is trained with mean square error loss along with a regularization term. Concretely, the Hypernet generates the parameters of all the previous tasks along with the parameters of the current task, which means that the model balances in improving the performance on the current task (forward transfer) as well as not degrading the performance of previous tasks (backward transfer)

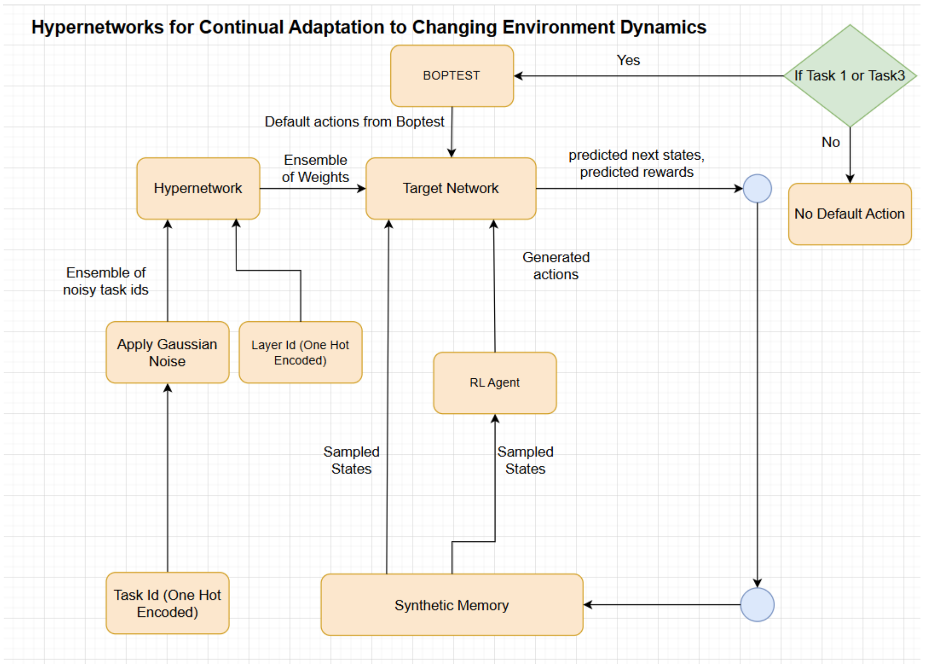

Hypernetworks for Continual Adaptation to Changing Environment Dynamics
BOPTEST
Yes
If Task 1 or Task3
Default actions from Boptest
No
Hypernetwork
Ensemble of Weights
Target Network
predicted next states, predicted rewards
No Default Action
Generated actions
Ensemble of noisy task ids
Apply Gaussian Noise
Layer Id (One Hot Encoded)
RL Agent
Sampled States
Sampled States
Task Id (One Hot Encoded)
Synthetic Memory

Figure 1: Hypernetwork architecture

4.5 Algorithm

Algorithm 1 Continual Model-Based RL with Hypernetwork for Environment Modeling

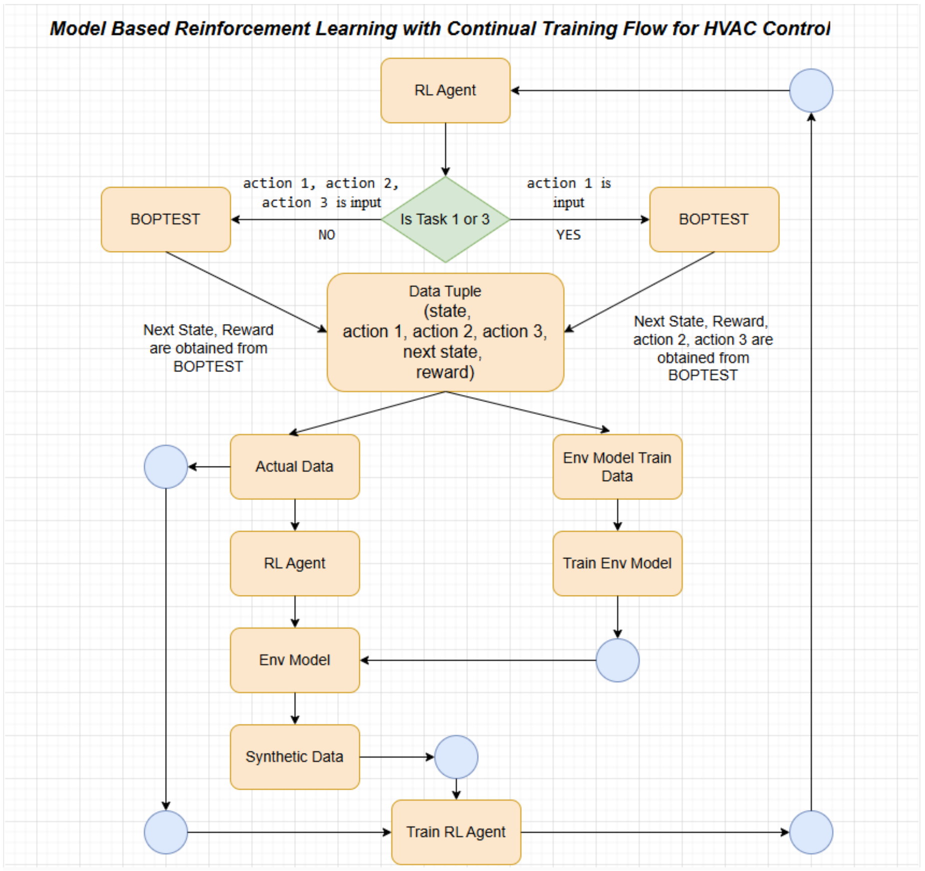
```
1:  Initialize policy $\pi_\theta$
2:  Initialize hypernetwork $H_\varphi$, target network $T_\delta$
3:  Initialize real data memory $\mathcal{M}_\alpha$, synthetic data memory $\mathcal{M}_\beta$
4:  Initialize hypernet train memory $\mathcal{M}_\gamma$
5:  Initialize regularization constant $\alpha$
6:  for each task $T_i$ in sequence do
7:      Reinitialize policy $\pi_\theta$
8:      Set training iterations $K_i$
9:      for $k = 1$ to $K_i$ do
10:         Interact with real environment $T_i$: collect $(s, a, r, s')$ using $\pi_\theta$ and store in $\mathcal{M}_\alpha, \mathcal{M}_\gamma$
11:         Train $H_\varphi$ using samples from $\mathcal{M}_\gamma$:
12:            $T_\delta \leftarrow H_\varphi(\texttt{task\_id}, \texttt{layer\_id})$
13:            sample $(s, a, s', r)$ from $\mathcal{M}_\gamma$
14:            predict $s', r \leftarrow T_\delta(s, a)$
15:            mse_loss $\leftarrow$ MSE(predictions, actual)
16:            regularization $\leftarrow$ MSE$(T_{\delta,\texttt{old}}, T_\delta)$
17:            loss $\leftarrow$ mse_loss $+ \beta \cdot$ regularization
18:            Update $H_\varphi$ by minimizing loss
19:         Generate one-step rollout using $H_\varphi$ and $T_\delta$:
20:            sample actual state $s$ from $\mathcal{M}_\alpha$
21:            take random action $a$ using $\pi_\theta$
22:            use $(s, a)$ to generate synthetic $(s', r)$ using $H_\varphi$ and $T_\delta$
23:            store tuple $(s, a, s', r)$ in $\mathcal{M}_\beta$
24:         Update policy $\pi_\theta$ using data from $\mathcal{M}_\alpha, \mathcal{M}_\beta$
25:     end for
26:     Optionally store $H_\varphi$ state or evaluate across tasks
27: end for
```

Implementation Details

All experiments were conducted on a system with the following specifications: **Processor:** 13th Gen Intel(R) Core(TM) i9-13900 @ 2.00 GHz **Installed RAM:** 32.0 GB **System Type:** 64-bit operating system, x64-based processor

Our model is implemented in Python using PyTorch. The complete codebase is available at: `https://github.com/gauthambekal93/hvac_continual_rl`

Complete Flowchart



Model Based Reinforcement Learning with Continual Training Flow for HVAC Control
RL Agent
action 1, action 2, action 3 is input
NO
Is Task 1 or 3
action 1 is input
YES
BOPTEST
BOPTEST
Data Tuple (state, action 1, action 2, action 3, next state, reward)
Next State, Reward are obtained from BOPTEST
Next State, Reward, action 2, action 3 are obtained from BOPTEST
Actual Data
Env Model Train Data
RL Agent
Train Env Model
Env Model
Synthetic Data
Train RL Agent

Figure 2: Training Loop

4.6 Input Output Mappings

Model I/O Tables

Table 1: Hypernetwork Inputs and Outputs

Component	**Description**
Inputs	One-hot encoded Task ID, One-hot encoded Layer ID
Outputs	Parameters for Target Model

As shown in Table 1, the hypernet takes one hot encoded values as input for Task ID and Layer ID. For example, task 1 can be represented as [1, 0, 0] respectively. The outputs are continuous real values, since they represent the parameters of the Target Network.

Table 2: Target Network Inputs and Outputs

Component	**Description**
Inputs	Real-time Zone Temperature, Dry Bulb Temperature Forecast, Cooling Setpoint, Heating Setpoint , Action 1, Action 2, Action 3
Outputs	Next Real-time Zone Temperature, Reward

The Target Network predicts the next zone temperature and reward using the input states and actions. The actions are obtained from RL_agent and BOPTEST environment depending on task-id.

Table 3: RL Agent Inputs and Outputs

Component	**Description**
Inputs	Time, Real-time Zone Temperature, Dry Bulb Temperature Forecast
Outputs	**Task 1 or Task 3:** Action 1 – Heat pump modulating signal **Task 2:** Action 1 – Heat pump modulating signal Action 2 – Heat pump evaporator fan Action 3 – Emission circuit pump

SAC model will generate 1 or 3 actions at output based on the task-id.

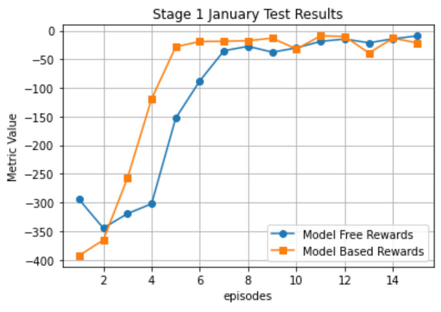
4.7 Parameters used

Table 4 lists a 0.99 discount factor and small learning rates for stability, with large buffers and 100-model ensembles to enhance diversity and reduce bias.

Table 4: Hyperparameter used in experiments

Hyperparameter	**Value**
Discount factor	0.99
Actor learning rate	0.00005
Critic learning rate	0.0002
Discount factor (γ)	0.99
batch size	1024
Real data buffer size	35000
Synthetic data buffer size	35000
Hypernet learning rate	0.0001
Hypernet training data buffer size	4000
Policy update frequency	Every 2 steps
Regularization coefficient (β)	0.1
No. of time steps per episode (K_i)	1344
No. of synthetic samples per time step	10
No. of models generated by Hypernet (for bias reduction)	100

5 Experimental Results



W

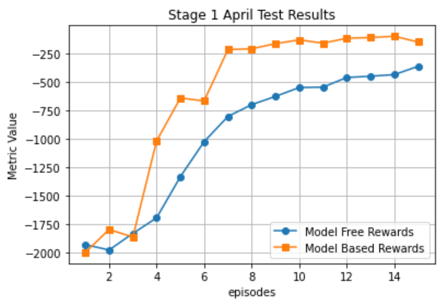
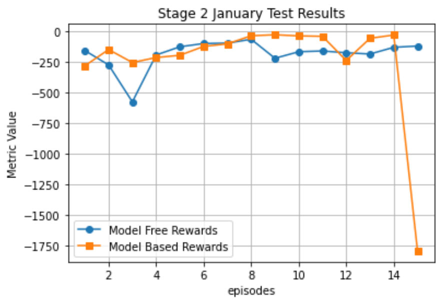


Figure 3: Stage 1 January test results

Figure 4: Stage 1 April test results

To compare the performance of the system, under different seasonal conditions we depicted Fig. 3 and Fig. 4 which shows the results for January and February respectively. For task 1, the actor generates only a heat pump modulating signal as output. The heat pump evaporator fan and emission circuit pump are set to default values and directly obtained from the BOPTEST environment. All three actions are given as input to Hypernet. The left shows results from January testing, and the right shows April performance, illustrating the RL agent control performance. This is task 1, which is the pretraining phase. The Hypernet is trained from scratch along with SAC models which are trained from scratch. As seen in the literature, MBRL performs significantly better than MFRL for both the January and April test periods.



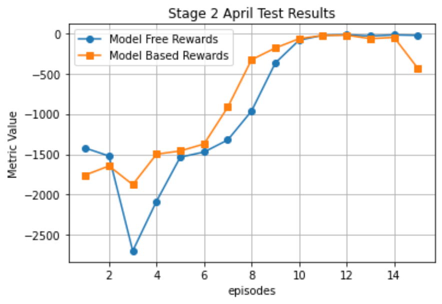


Figure 5: Stage 2 January test results

Figure 6: Stage 2 April test results

In task 2, Fig. 5 and Fig. 6 shows the finetuning of the Hypernets after initial training in task 1. For task 2, the actor generates three actions, heat pump modulating signal, heat pump evaporator fan and emission circuit pump as output, which are given as input to Hypernet. The SAC agent is again trained from scratch for both MFRL and MBRL. However, we utilize transfer learning on the Hypernet which was already trained on task 1. We see that for the January and April test periods, the MBRL tends to perform better from early on, however, caution should be taken when training Hypernets since they can have a significant drop in performance due to instabilities.

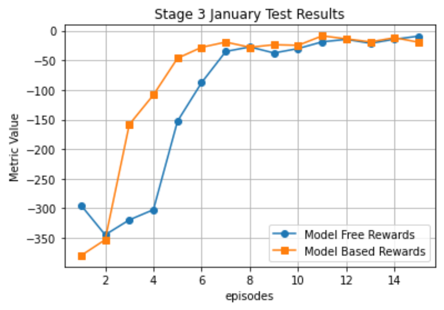


Figure 7: Stage 3 January test results

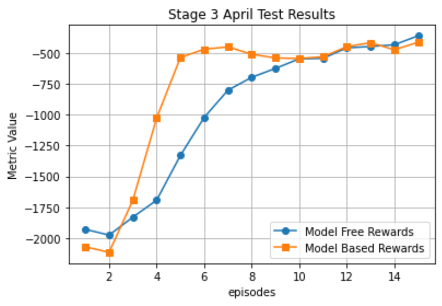


Figure 8: Stage 3 April test results

Fig. 7 and Fig. 8 plots show the results after sequential training of the Hypernet for Task 1 and Task 2. The configuration of the action space is the same as in Task 1. The SAC agent is again trained from scratch for both MFRL and MBRL. Even with minimal retraining of the Hypernet on Task 1, we observe significantly quicker convergence compared to vanilla model-free RL. This demonstrates that the Hypernet retained its ability to accurately model environment dynamics after being trained on Task 2, effectively mitigating catastrophic forgetting with minimal retraining.

As can be seen from the three sequential tasks it can be seen that, MBRL using Hypernetworks can help in faster convergence than MFRL. From tasks 1 and 2, it can be observed that Hypernetwork is able to retain the environment dynamics, even after the intermediate training for task 2. However, training Hypernetworks can be challenging, as evidenced by the results in Task 2, where a sudden drop in performance was observed despite previously stable behavior.

6 Conclusion and Future work

In this work, we proposed a model-based reinforcement learning framework for HVAC control that leverages a Hypernetwork to learn the dynamics of the environment in tasks with varying action spaces. Our approach enables transfer learning by conditioning dynamics generation on task and layer identifiers, allowing the model to adapt to different control configurations. Experimental results in multiple BOPTEST scenarios demonstrate that our method outperforms model-free RL in terms of sample efficiency. Additionally, the Hypernetwork shows strong backward transfer capabilities, retaining performance on earlier tasks even after training on new ones, thereby mitigating catastrophic forgetting with minimal retraining.

While our results are promising, several directions remain for future investigation: While we performed our experiments on 3 tasks, it is possible to extend to several sequential tasks and is a part of our ongoing research. Also, we can extend the method to handle multi-zone environments with more complex dynamics and interactions. Finally, validating the approach in real or emulated buildings beyond BOPTEST to assess practical applicability and safety constraints will be a promising direction.

References

[1] M. Janner, J. Fu, M. Zhang, and S. Levine, “When to trust your model: Model-based policy optimization,” *arXiv*, 2021. `https://doi.org/10.48550/arXiv.1906.08253`

[2] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, “A comprehensive survey on transfer learning,” *arXiv*, 2020. `https://doi.org/10.48550/arXiv.1911.02685`

[3] Y. Luo, Z. Yang, F. Meng, Y. Li, J. Zhou, and Y. Zhang, “An empirical study of catastrophic forgetting in large language models during continual fine-tuning,” *arxiv*, 2025. `https://doi.org/10.48550/arXiv.2308.08747`

[4] R. S. Sutton, “Dyna, an integrated architecture for learning, planning, and reacting,” *ACM*, 1991. `https://doi.org/10.1145/122344.122377`

[5] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker, “Model-based reinforcement learning: A survey,” *arxiv*, 2022. `https://doi.org/10.48550/arXiv.2006.16712`

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing Atari with deep reinforcement learning,” *arxiv*, 2013. `https://doi.org/10.48550/arXiv.1312.5602`

[7] K. A. Sayeda, A. Boodia, R. S. Broujenyc, and K. Beddiard, “Reinforcement learning for HVAC control in intelligent buildings: A technical and conceptual review,” *Journal of Building Engineering*, 2024. `https://doi.org/10.1016/j.jobe.2024.110085`

[8] S. Taheri, P. Hosseini, and A. Razban, “Model predictive control of heating, ventilation, and air conditioning (HVAC) systems: A state-of-the-art review,” *arxiv*, 2022. `https://doi.org/10.1016/j.jobe.2022.105067`

[9] C. Gao and D. Wang, “Comparative study of model-based and model-free reinforcement learning control performance in HVAC systems,” *Elsevier*, 2023. `https://doi.org/10.1016/j.jobe.2023.106852`

[10] D. Coraci, S. Brandi, T. Hong, and A. Capozzoli, “An innovative heterogeneous transfer learning framework to enhance the scalability of deep reinforcement learning controllers in buildings with integrated energy systems,” *Springer*, 2024. `https://doi.org/10.1007/s12273-024-1109-6`

[11] L. Maier, J. Brillert, E. Zanetti, and D. Müller, "Approximating model predictive control strategies for heat pump systems applied to the building optimization testing framework (BOPTEST)," *Taylor & Francis*, 2023. `https://doi.org/10.1080/19401493.2023.2280577`

[12] K. Kadamala, D. Chambers, and E. Barrett, "Enhancing HVAC control systems through transfer learning with deep reinforcement learning agents," *ELSEVIER*, 2024. `https://doi.org/10.1016/j.segy.2024.100131`

[13] S. Xu, Y. Wang, Y. Wang, Z. O'Neill, and Q. Zhu, "One for many: Transfer learning for building HVAC control," *BuildSys*, 2020. `https://doi.org/10.1145/3408308.3427617`

[14] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "PathNet: Evolution channels gradient descent in super neural networks," *arxiv*, 2017. `https://doi.org/10.48550/arXiv.1701.08734`

[15] D. Blum, J. Arroyo, S. Huang, J. Drgoňa, F. Jorissen, H. T. Walnum, Y. Chen, K. Benne, D. Vrabie, M. Wetter, and L. Helsen, "Building optimization testing framework (BOPTEST) for simulation-based benchmarking of control strategies in buildings," *Taylor & Francis*, 2021. `https://doi.org/10.1080/19401493.2023.2280577`

[16] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arxiv*, 2018. `https://arxiv.org/pdf/1801.01290`

[17] V. K. Chauhan, J. Zhou, P. Lu, S. Molaei, and D. A. Clifton, "A brief review of hypernetworks in deep learning," *arxiv*, 2024. `https://doi.org/10.1007/s10462-024-10862-8`

[18] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: Theory, method and application," *arxiv*, 2024. `https://doi.org/10.48550/arXiv.2302.00487`

[19] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience replay for continual learning," *arxiv*, 2018. `https://doi.org/10.48550/arXiv.1811.11682`

[20] J. von Oswald, C. Henning, B. F. Grewe, and J. Sacramento, "Continual learning with hypernetworks," *arxiv*, 2019. `https://doi.org/10.48550/arXiv.1906.00695`