Comparative Study of Deep Reinforcement Learning Algorithms for Residential HVAC Control

Casey Oliver Dettlaff
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164
casey.dettlaff@wsu.edu

Kunal Shankar
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164
kunal.shankar@wsu.edu

Abstract

Buildings account for 40% of U.S. electricity consumption, with HVAC systems representing the largest end use load. In systems with dynamic electricity price, buildings' thermal inertia can be exploited for cost minimization. We present a framework for reinforcement learning for HVAC control that minimizes cost while maintaining a comfortable indoor temperature via a gymnasium environment and resistance–capacitance thermal building model. We evaluate three algorithms: Deep Q Network, Proximal Policy Optimization, and Soft Actor Critic. Hyperparameter tuning is performed for all algorithms to optimize performance. Results demonstrate that DQN is the best algorithm for learning HVAC control.

1 Introduction

Buildings account for approximately 40% of total electricity consumption in the United States, with HVAC systems representing the largest single end use, consuming 50–60% of building energy [1]. Utilities increasingly rely on time of use tariffs and real time pricing to balance supply and demand [2]. These dynamic price signals create opportunities for demand side flexibility that are not exploited by traditional thermostat controls [1]. This results in unnecessary costs for buildings and missed opportunities for grid level demand response.

1.1 HVAC Operation and Control Challenges

In most residential buildings, air conditioners and heat pumps operate in a duty cycle mode, switching between zero power and rated capacity. Simple bang bang thermostats toggle equipment on when indoor temperature exceeds an upper threshold and off when it falls below a lower threshold. Frequent short cycling can increase mechanical wear, reduce efficiency due to start up transients, and shorten equipment lifetime [3]. Practical controllers must avoid unnecessary switching while still meeting comfort requirements.

Model predictive control can explicitly optimize HVAC duty cycles over a forecast horizon and has demonstrated energy savings in buildings, but requires accurate system identification, significant computational resources, and careful constraint handling [4]. These barriers limit MPC deployment at scale, particularly in small and medium buildings with limited automation infrastructure [5]. This

motivates data driven methods that can learn control policies directly from interaction with a building environment.

1.2 Deep Reinforcement Learning for HVAC

Reinforcement learning provides a framework for learning control policies from interaction with an environment, and deep reinforcement learning extends this using neural networks to approximate value functions and policies [6]. For HVAC control, the environment represents a grid–connected building, whose state includes indoor temperature, outdoor conditions, time features, and price signals. Value based methods such as Deep Q Network are well suited to such discrete action spaces and have shown promising results for rule replacement in building control [7]. Policy based methods such as Proximal Policy Optimization [8] and entropy regularized methods such as Soft Actor Critic [9] can handle both discrete and continuous actions and offer different exploration and stability properties [10]. Prior studies report that algorithm choice and hyperparameter tuning significantly affect cost and comfort outcomes [11].

1.3 Reward Design for Multi Objective Control

A central challenge in applying deep reinforcement learning to HVAC is reward design. The controller must simultaneously minimize electricity cost, maintain thermal comfort, and limit equipment cycling to avoid degradation [3]. Cost only rewards can encourage aggressive load shifting at the expense of large comfort violations, while purely comfort based rewards ignore economic objectives [7]. The formulation must therefore combine comfort penalty, energy, and switching penalty in a weighted sum [1, 12]. Smooth penalty functions and soft comfort constraints have been proposed to improve learning stability and safety margins [13].

1.4 Contributions

This work develops and compares three deep reinforcement learning algorithms for single zone residential HVAC control: Deep Q Network, Proximal Policy Optimization, and Soft Actor Critic. Each controller operates at a five minute interval in a custom Gymnasium environment that couples an RC building model with time–varying electricity prices. Using our custom environment, we provide an empirical study of how algorithm choice and reward design affect cost, comfort, and equipment switching.

2 Modeling

This section introduces the control environment interface and the building thermal model that form the basis of our learning framework.

2.1 Gymnasium Reinforcement Learning Framework

Gymnasium provides a standardized API for developing and evaluating reinforcement learning algorithms [14]. The framework defines a cyclical interaction between an agent (the controller) and an environment (the building system), as illustrated in Figure 1. At each timestep, the agent observes the current state of the environment, selects an action according to its policy, and receives a scalar reward signal that encodes control objectives. The environment then transitions to a new state based on the selected action and underlying system dynamics. This cycle repeats until a terminal condition is reached or a predefined episode length is exceeded.

The standard Gymnasium interaction is implemented through the call

```
observation, reward, terminated, truncated, info = env.step(action)
```

where `terminated` indicates whether the episode has ended naturally, `truncated` indicates whether a time limit has been reached, and `info` provides auxiliary diagnostic information.

Four elements formally characterize an HVAC control environment within the Gymnasium framework.

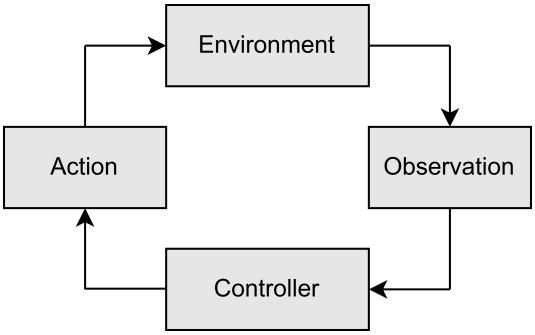

Figure 1: Gymnasium reinforcement learning interaction loop.

Observation space Specifies the information available to the agent at each timestep. For HVAC control, this typically includes indoor temperature, outdoor temperature, time of day, and electricity price, represented as continuous or discrete variables. The observation space defines the input to the neural network policy or value function used by the deep reinforcement learning algorithm.

Action space Specifies the set of admissible control actions. A discrete action space can capture duty cycle operation through binary on off HVAC commands, while a continuous space can represent modulating setpoints or variable power levels. The choice of action space determines which deep reinforcement learning algorithms are applicable, with value based methods such as Deep Q Network requiring discrete actions and policy gradient methods such as Proximal Policy Optimization and Soft Actor Critic supporting both discrete and continuous actions.

Reward function Encodes the control objectives into a scalar signal provided to the agent after each action. In the HVAC setting, the reward typically reflects a trade off between electricity cost and deviation of indoor temperature from a comfort range, and may also penalize excessive equipment switching to discourage short cycling that accelerates wear.

Step function Advances the environment dynamics by one timestep. Given the current state and a selected action, the step function updates the building thermal state and returns the next observation and reward. In our setting, this update is computed using a low order building thermal model driven by outdoor conditions and the chosen HVAC power. The Gymnasium API abstracts this interaction, allowing different deep reinforcement learning algorithms to be applied without changing the environment implementation.

2.2 Building Thermal Models

Efficient reinforcement learning for HVAC control requires thermal models that are both physically meaningful and computationally tractable. High fidelity building simulators such as EnergyPlus can represent detailed multi zone dynamics but are often too slow for large scale RL training [1]. Purely data driven black box models can fit measured data but may lack interpretability and extrapolation capability under new weather or pricing conditions [15]. Grey box models based on resistance capacitance networks strike a balance by capturing dominant heat transfer mechanisms with a small number of parameters [16].

Figure 2 illustrates a multi zone resistance capacitance (RC) thermal network for a residential building. The structure comprises multiple thermal masses representing different building components: attic and roof, interior walls, exterior walls, windows, and the indoor air volume. Each thermal mass has an associated capacitance that stores thermal energy, and resistances model heat transfer between masses and to the outdoor environment. Heat flows into the system include solar radiation (Q_{solar}), internal gains from occupants and equipment (Q_{int}), and HVAC cooling (Q_{AC}). This network can be reduced to simpler representations depending on the fidelity required for control applications.

For the single zone setting considered in this work, the building is represented as a single thermal mass with indoor temperature T_{in} and effective thermal capacitance C. Heat transfer through the building envelope is governed by an equivalent thermal resistance R_{win} to the outdoor temperature T_{out}. The indoor thermal mass receives heat from multiple sources: HVAC cooling or heating (Q_{HVAC}), internal gains from occupants and appliances (Q_{int}), solar heat gains through windows and

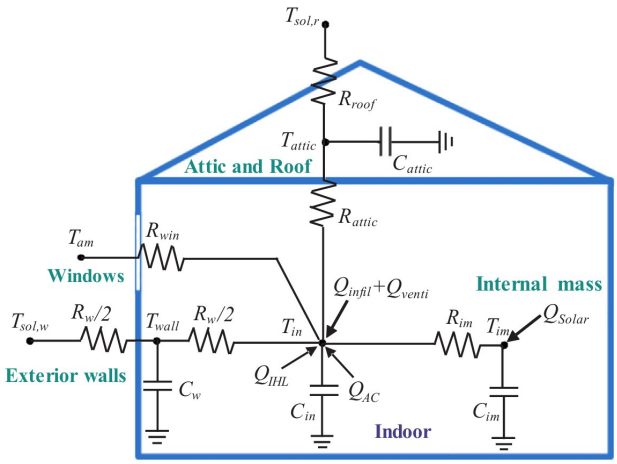

Figure 2: RC modelling of a residential building

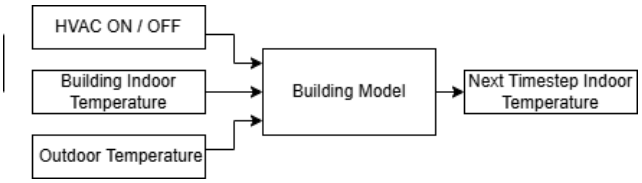

Figure 3: Simplified single zone building thermal model structure. Outdoor temperature and HVAC on/off commands serve as inputs to the building model, which computes the next indoor temperature state at each timestep.

roof (Q_{solar}), ventilation heat exchange (Q_{vent}), and infiltration through building envelope leakage (Q_{inf}). The resulting energy balance for the indoor air node can be written as

$$C\frac{dT_{\text{in}}}{dt} = \frac{T_{\text{out}} - T_{\text{in}}}{R_{\text{win}}} + Q_{\text{HVAC}} + Q_{\text{int}} + Q_{\text{solar}} + Q_{\text{vent}} + Q_{\text{inf}},$$

where each heat flow term contributes to the rate of change of indoor temperature. This continuous time ordinary differential equation can be discretized at the control sampling interval using zero order hold or other numerical integration methods and embedded directly into the Gymnasium step function.

Figure 3 shows the simplified structure used in the Gymnasium environment. The building model function takes three inputs: the current indoor temperature, the outdoor temperature, and the binary HVAC on/off command. At each five minute timestep, the model integrates the thermal dynamics and returns the updated indoor temperature, which is then included in the observation provided to the reinforcement learning agent. This modular design decouples the thermal physics from the control algorithm, enabling the same building model to be used across different deep reinforcement learning methods.

3 System Setup

This section introduces the data generation and preprocessing required to create the control environment. The datasets used include the electricity price profile, the outdoor temperature profile, and the non-HVAC building load profile. Each profile is produced at a 5-minute resolution, resulting in 288 timesteps per day, and is saved as a CSV file in the environment's `data/` directory.

3.1 Electricity Price Profile

The script first constructs an hourly price curve that reflects realistic dynamic pricing patterns, including low overnight prices, morning increases, midday dips, and evening peaks. This hourly signal is then interpolated to 5-minute intervals and perturbed with small random noise to introduce realistic variability. The resulting price trajectory is clipped to remain within acceptable bounds and saved as `electricity_price.csv`.

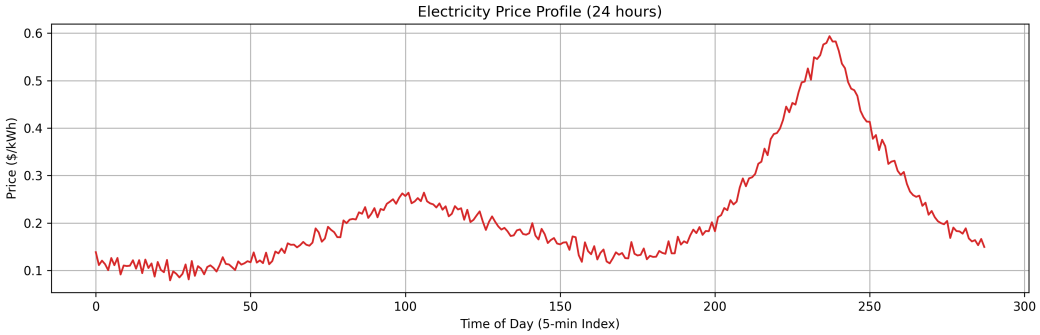

Figure 4: Electricity Price

3.2 Outdoor Temperature Profile

A representative hourly outdoor temperature pattern is defined to model a typical day in a warm climate. The script interpolates this profile to 5-minute resolution and adds small Gaussian noise to mimic natural temperature fluctuations. The values are then clipped to physical limits (−20°C to 50°C) before being stored in `outdoor_temperature.csv`.

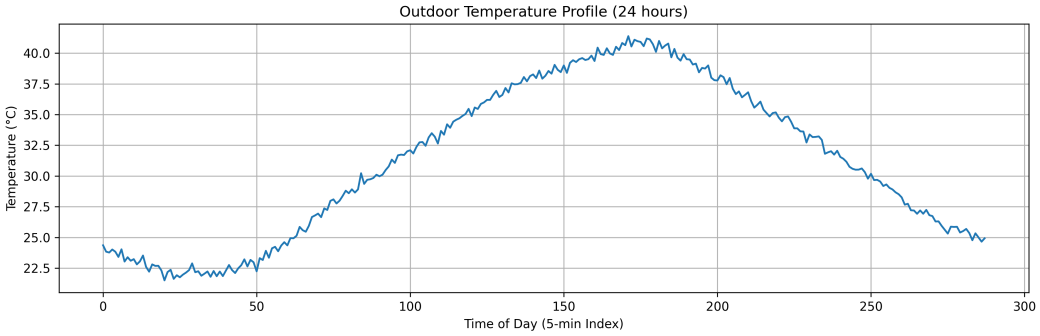

Figure 5: Outdoor Temperature Profile

3.3 Non-HVAC Load Profile

To model internal building usage, the script generates a non-HVAC load profile representing plug loads, lighting, and appliance usage throughout the day. A typical hourly load shape is interpolated to 5-minute intervals and scaled with random noise to reflect realistic variability in household consumption. The load values are clipped to reasonable operating ranges and written to `non_hvac_load.csv`.

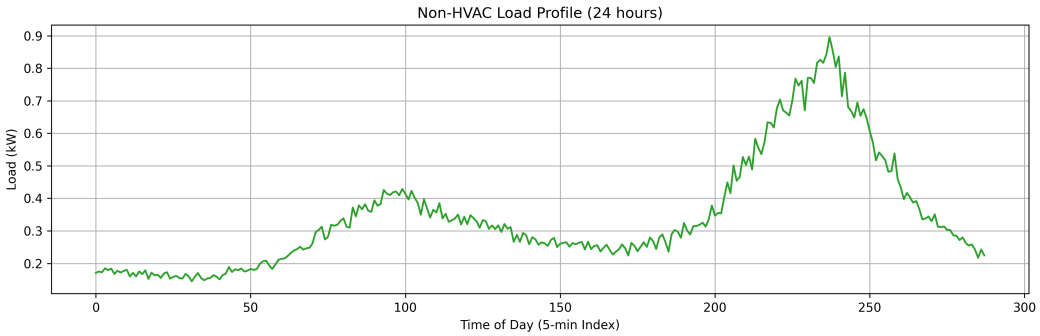

Figure 6: Non HVAC Load

4 Environment Design

Having established the Gymnasium framework for reinforcement learning and reviewed deep RL algorithms for HVAC control, we now detail the formal design of our simulation environment. Building on the thermal modeling foundations discussed in Section 2, we integrate a calibrated first-order RC model with a Gymnasium-compliant interface, enabling DQN, PPO, and SAC agents to learn cost-optimal control policies under realistic weather and pricing scenarios. This section presents the mathematical formulation, system assumptions, and notation used throughout this work.

4.1 System Architecture and Assumptions

Our environment models a building whose indoor temperature is affected by the outdoor climate and the HVAC power supplied to it, while also experiencing time-varying electricity prices. The building dynamics follow a first-order RC thermal model that updates indoor conditions based on HVAC energy use and outdoor temperature. We adopt the following assumptions:

- **Single-zone building**: The building is modeled as one thermal mass with a uniform indoor temperature. This simplification is suitable for small residential buildings and enables faster simulation during RL training.
- **Fixed RC parameters**: The thermal resistance and capacitance are held constant. These parameters were estimated using measured residential building data in earlier work. We assume these parameters capture the building dynamics, and remain unchanged throughout the day.
- **Fixed Heats**: The RC building model must also account for heat gains from internal sources such as occupants and appliances, as well as heat gains or losses due to solar radiation and ventilation. We assume all non-HVAC heat gain and loss contributions remain constant.
- **Constant HVAC performance**: The cooling system operates with a fixed cooling capacity and a constant coefficient of performance (COP). The cooling delivered per kW of electrical input does not vary.
- **Binary control**: The HVAC system has only two states: ON or OFF. When turned ON, the unit draws its full rated electrical power.

4.2 Markov Decision Process Formulation

We formulate the HVAC control problem as a finite-horizon Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and γ is the discount factor Each episode spans 24 hours at 5-minute resolution, after which the environment resets to initial conditions.

4.2.1 Observation Space

At timestep t, the RL agent observes a dictionary-based observation vector implemented as a Gymnasium `Dict` observation space. The features were chosen based on which state variables were most likely to help the HVAC controller make optimal decisions. Each feature is represented as a single-element numpy array of dtype `float32`.

$$\begin{aligned}\mathbf{s}_t = \{&\text{Indoor Temperature} : T_{\text{in},t},\\ &\text{Outdoor Temperature} : T_{\text{out},t},\\ &\text{Electricity Price} : \lambda_t,\\ &\text{Time of Day} : h_t\},\end{aligned} \tag{1}$$

4.2.2 Action Space

The agent selects discrete actions $a_t \in \mathcal{A} = \{0, 1\}$ at each timestep corresponding to binary HVAC control:

$$a_t = \begin{cases} 0 & \text{HVAC OFF},\\ 1 & \text{HVAC ON}. \end{cases} \tag{2}$$

4.2.3 Reward Function

The reward consists of three parts: a comfort penalty, a cost penalty, and a switching penalty. The comfort penalty p_t encourages the agent to keep the indoor temperature within a defined comfort band (we use a range of 20.0°C–21.67°C). Let T_t represent the indoor temperature, with comfort limits $T_{\min}$ and $T_{\max}$. The midpoint and half-width are:

$$T_{\text{mid}} = \frac{T_{\min} + T_{\max}}{2}, \quad w = \frac{T_{\max} - T_{\min}}{2}. \tag{3}$$

$$p_t = -\left(\frac{T_t - T_{\text{mid}}}{w}\right)^2 \tag{4}$$

This provides a smooth, differentiable penalty that equals zero at the midpoint and grows quadratically as the temperature deviates in either direction. This formulation encourages the agent to maintain temperatures near the center of the comfort zone while avoiding hard constraint violations that can lead to unstable learning.

The cost penalty p_t^{price} motivates the agent to consume power when electricity prices are low (ranging from 0.1 to 0.2 $ per kWh).

$$p_t^{\text{price}} = \frac{\text{price}_t - 0.1}{0.1}.$$

- penalty $= 0$ when $\text{price}_t = 0.1$,
- penalty $= 1$ when $\text{price}_t = 0.2$.

The switching penalty p_t^{switch} discourages the agent from rapidly switching the HVAC on and off to prevent equipment weardown.

$$p_t^{\text{switch}} = \begin{cases} 1 & \text{HVAC}_t \neq \text{HVAC}_{t-1}\\ 0 & \text{otherwise}. \end{cases}$$

The total reward is a weighted sum of the penalties, where c_1, c_2, and c_3 are constants that can be tuned to influence the agent's behavior.

$$r_t = -\left(c_1\, p_t^{\text{comfort}} + c_2\, p_t^{\text{price}} + c_3\, p_t^{\text{switch}}\right),$$

4.2.4 Initialization Function

The constructor accepts pre-loaded data profiles for electricity price, outdoor temperature, and non-HVAC load as pandas DataFrames.

4.2.5 Reset Function

The `reset()` method initializes each episode by setting the timestep counter to zero, resetting indoor temperature to a comfortable baseline value, and loading initial values from the exogenous data profiles. Returns the initial observation dictionary and an empty info dictionary.

4.2.6 Building Model Function

The indoor temperature for the current timestep is then calculated using the RC building model, discretized with zero-order hold (ZOH) integration. The model is expressed in state-space form:

$$\mathbf{x}_{t+1} = \mathbf{A}_d \, \mathbf{x}_t + \mathbf{B}_d \, \mathbf{u}_t, \tag{5}$$

Where:

- $\mathbf{x}_t = [T_{\text{in},t}]$: Indoor Temperature State
- $\mathbf{u}_t$: Input Vector Containing:
  - Outdoor Temperature
  - Internal Heat Gains
  - HVAC Cooling Power
  - Solar Heat Gains
  - Ventilation Heat Gains
  - Infiltration Heat Gains

Real input data: The building simulator is driven by measured data from two established sources. Weather inputs including ambient temperature and solar irradiance are obtained from NREL's National Solar Radiation Database (NSRDB) for a southern U.S. location, covering summer conditions. Electrical load inputs representing non-HVAC household demand are extracted from the Pecan Street Dataport residential smart meter dataset (Austin, Texas), providing measured profiles of appliances, lighting, and plug loads.

4.2.7 Step Function

Step method: The `step(action)` method executes one timestep of environment dynamics:

1. Converts the discrete action to HVAC electrical power
2. Increments the timestep counter and updates exogenous variables from data profiles
3. Calls the RC building model with ZOH integration to compute the next indoor temperature
4. Computes the immediate reward based on the quadratic comfort penalty
5. Stores profiles for post-episode analysis
6. Returns observation dictionary, reward, termination flag, truncation flag, and info dictionary

In the step function, outdoor temperature, electricity price, and non-HVAC load are refreshed from external data sources to reflect the current timestep. The HVAC load is determined by the action (an action of 0 sets HVAC power to 0, while an action of 1 applies the unit's rated 2.5 kW power).

The step function then computes the reward. It also checks whether the current timestep has reached the maximum allowed value to determine if the episode should terminate or truncate. Our simulation ends after one full day (288 timesteps). The step function returns the observation, reward, and the termination signals to the main simulation loop.

5 Reinforcement Learning Algorithms

Having defined the MDP formulation and Gymnasium environment, we now describe the deep reinforcement learning algorithms employed to learn HVAC control policies. This section focuses on the algorithmic foundations and hyperparameter configurations for DQN and PPO, with SAC included for completeness.

5.1 Deep Q-Network (DQN)

Deep Q-Networks (DQN) use a neural network to estimate how good each action is in a given state. To keep learning stable, DQN uses two important ideas. First, it saves past experiences in a replay buffer and randomly samples them during training so the agent does not learn from highly correlated data. Second, it keeps a separate copy of the network, called the target network, which is updated only once in a while. This gives the algorithm steady targets to learn from and prevents the training from becoming unstable.

The agent explores using an ϵ-greedy strategy, meaning it periodicslly chooses random actions. At the start, $\epsilon = 1.0$, so the agent explores more, but it gradually decreases to 0.02 so the agent becomes more confident and selects the best actions more often. During training, the network learns by educing the difference between its predicted Q-values and the target Q-values, which are calculated using the Bellman equation.

We conduct a structured hyperparameter search for DQN using a grid search to find settings that provide a good balance between learning speed, stability, and final performance. The grid search tests all 54 possible combinations of the selected parameters. For each setting, we train a DQN agent for 50,000 timesteps and measure how well it performs over a full 24-hour episode. Performance is scored using the total absolute deviation from the comfort zone midpoint, which captures how well the agent maintains comfortable indoor temperatures throughout the day. We also keep several parameters fixed, including the target network update interval (500 steps) and the final exploration epsilon (0.02). The search space includes:

- **Learning rate**: $\{10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$ Controls the magnitude of network weight updates.
- **Discount factor** γ: $\{0.95, 0.98, 0.995\}$ Determines the importance of future rewards.
- **Replay buffer size**: $\{50{,}000, 100{,}000\}$ Affects the diversity of experience sampling.
- **Exploration fraction**: $\{0.05, 0.1, 0.2\}$ Controls how quickly ϵ decays from 1.0 to 0.02.

The optimized hyperparameters identified by the grid search are:

- **Learning rate**: 0.001
- **Discount factor** γ: 0.98
- **Replay buffer size**: $100{,}000$
- **Exploration fraction**: 0.05

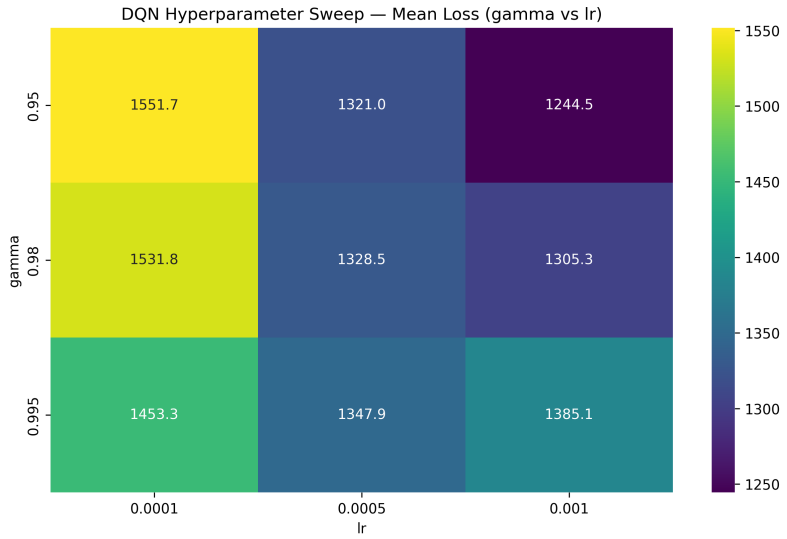

Figure 7: DQN Hyperparameter Heatmap

5.2 Proximal Policy Optimization (PPO)

PPO is a reinforcement learning method that learns a policy $\pi_\theta(a \mid s)$ directly, meaning it tries to improve the actions it chooses by adjusting the policy parameters. Instead of learning a Q-function, PPO updates the policy through gradient ascent, but it does so carefully to avoid making changes that are too large. To achieve this, PPO uses a *clipped objective* that limits how much the policy is allowed to move in a single update.

PPO uses two neural networks: one that outputs the policy (a distribution over the two HVAC actions), and another that estimates the value of each state. The agent collects several steps of experience, then performs multiple passes of training on this data using mini-batches. This process helps PPO learn stable and reliable policies while avoiding overly large updates. PPO is known stability and robustness in continuous control problems. Its clipped objective helps prevent unstable policy shifts, making it well suited for environments where small changes in action can lead to gradual, time-delayed effects.

For PPO hyperparameter optimization, the search space includes:

- **Learning rate**: $\{10^{-5}, 5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4}\}$
- **Discount factor** γ: $\{0.95, 0.98, 0.995\}$
- **Batch size**: $\{64, 128, 256\}$
- **Rollout horizon** n_{steps}: $\{256, 512, 1024\}$

The optimal hyperparameters were:

- **Learning rate**: 5×10^{-5}
- **Rollout horizon** n_{steps}: 256
- **Batch size**: 256
- **Discount factor** γ: 0.95

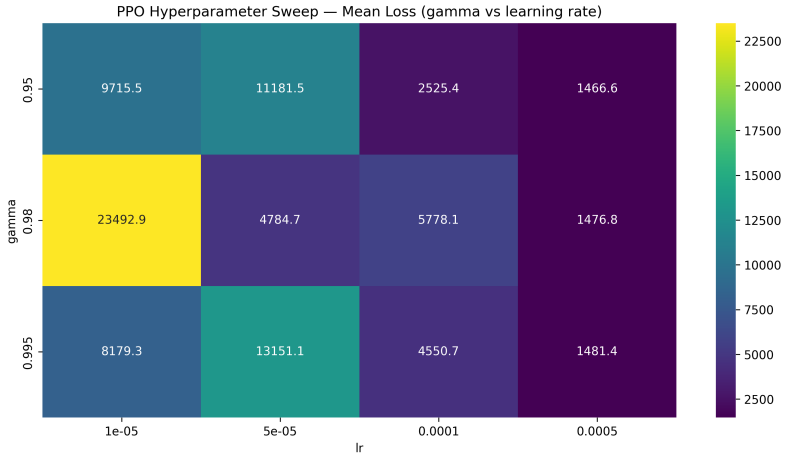

Figure 8: PPO Hyperparameter Heatmap

5.3 Soft Actor-Critic (SAC)

Soft Actor–Critic (SAC) is an actor–critic reinforcement learning method that learns both a policy and a value function. Unlike traditional algorithms that focus only on maximizing rewards, SAC also encourages the agent to stay uncertain by rewarding higher policy entropy. This means the agent is less likely to get stuck repeating the same actions early in training. Although SAC is typically used for continuous action spaces, it can be adapted to discrete control. We considered SAC for this application because HVAC control involves delayed temperature response, meaning both actions (HVAC ON and HVAC OFF) can appear reasonable in the short term.

The hyperparameter space for SAC is:

- **Learning rate**: $\{3 \times 10^{-4}, 10^{-4}, 5 \times 10^{-5}\}$
- **Discount factor** γ: $\{0.95, 0.98, 0.995\}$
- **Batch size**: $\{64, 128, 256\}$
- **Polyak smoothing coefficient** τ: $\{0.005, 0.01\}$
- **Entropy coefficient setting**: $\{\texttt{auto}, 0.001\}$

The optimized parameters are:

- **Learning rate**: 3×10^{-4}
- **Discount factor** γ: 0.98
- **Batch size**: 128
- **Polyak smoothing coefficient** τ: 0.01
- **Entropy coefficient**: `auto`

6 Results

6.1 Baseline Results

We use a simple bang-bang controller as the baseline upon which to compare our results. At each timestep, the controller compares the indoor temperature T_t against the upper comfort bound.

$$\text{action}_t = \begin{cases} 1, & \text{if } T_t > 21.67^\circ\text{C}, \\ 0, & \text{otherwise.} \end{cases}$$

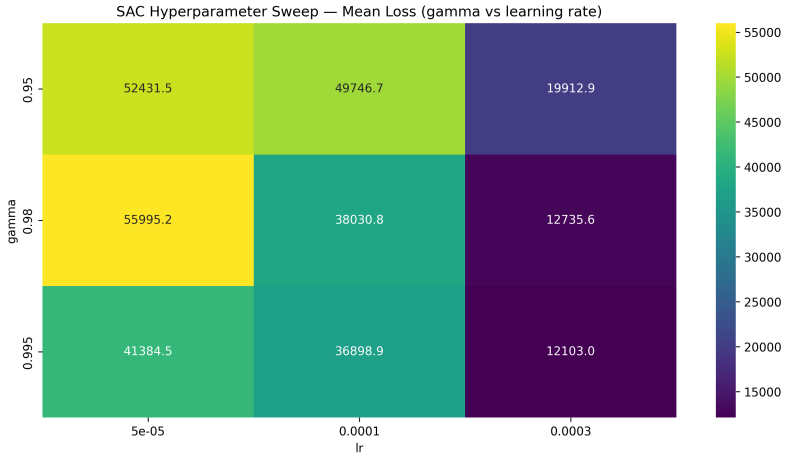

Figure 9: SAC Hyperparameter Heatmap

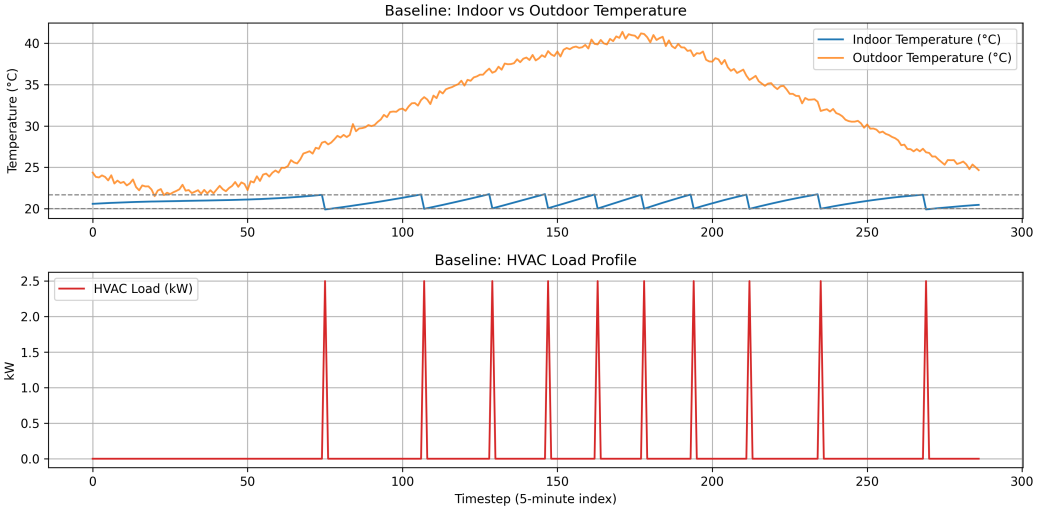

Figure 10: Baseline Controller Results

6.2 Comfort-Only Results

We first attempted to train an RL controller using a reward function that included only the comfort penalty (with all other penalties set to zero). The goal was to produce an RL policy that behaves comparably to the baseline controller, allowing us to validate the RL formulation.

6.2.1 DQN Results

The DQN achieved moderate performance after 500,000 timesteps, equivalent to 1740 training intervals. Although it did not match the baseline controller or consistently maintain indoor temperature within the comfort bounds, it did learn the HVAC's duty-cycle behavior. We believe that with a longer training duration, DQN could serve as a suitable algorithm for this HVAC control task.

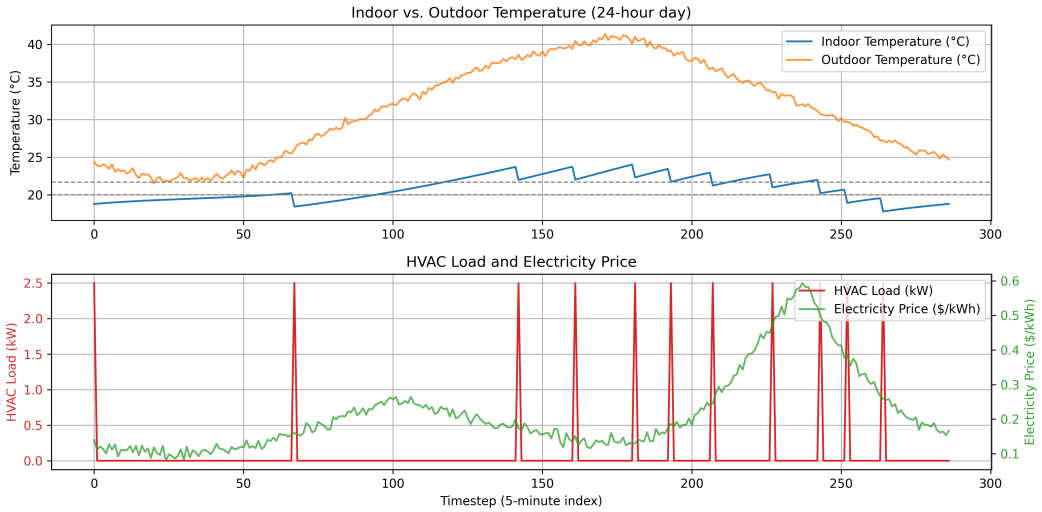

Figure 11: DQN Results

6.2.2 PPO Results

The PPO algorithm performed poorly. Even with tuned parameters, it failed to learn meaningful HVAC control behavior. Instead, it simply switched the HVAC on at the start of the day and did not adapt afterwards.

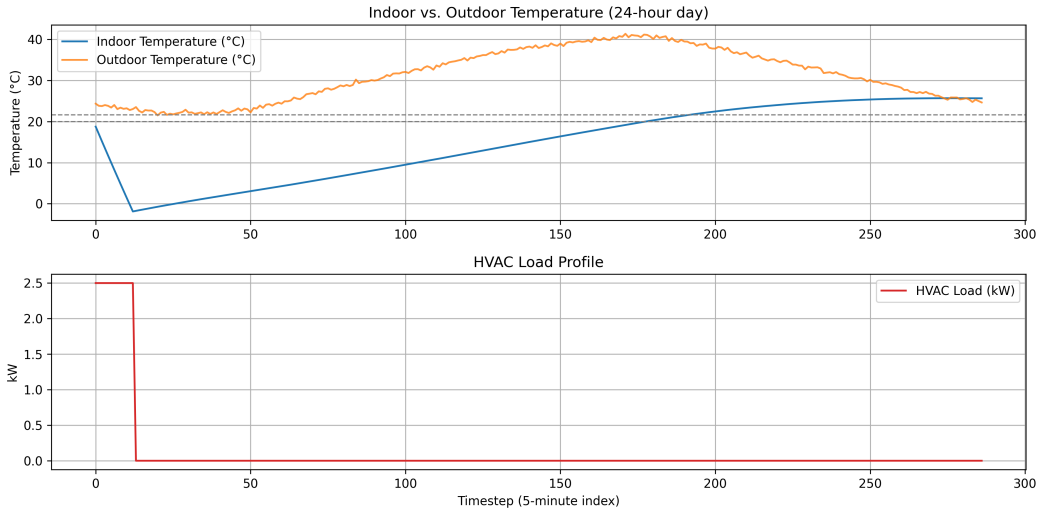

Figure 12: PPO Results

6.2.3 SAC Results

The SAC algorithm did not learn the proper control of the HVAC. It cooled the building too early, after which all actions resulted in a high penalty, and it failed to learn.

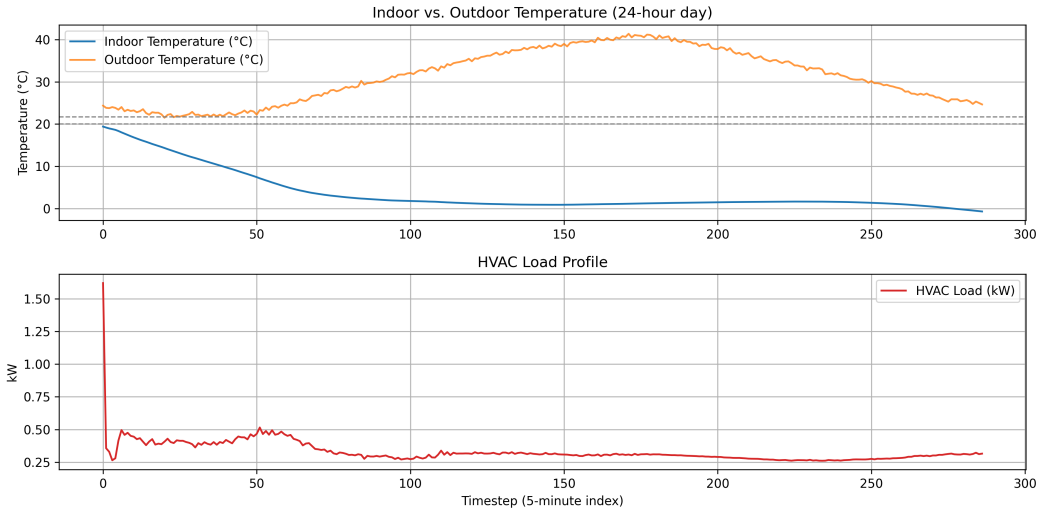

Figure 13: SAC Optimized Results

6.3 Cost Aware Controller Results

For the cost-aware controller, we trained a DQN using the same optimized hyperparameters for 500,000 timesteps. Similar to the comfort-only version, this agent learned a duty-cycle strategy and attempted to maintain temperatures within the comfort bounds. Unlike the comfort-only DQN, however, the cost-aware model began cooling in advance of the price spikes, pre-chilling the building during cheaper periods to reduce energy costs during peak rates.

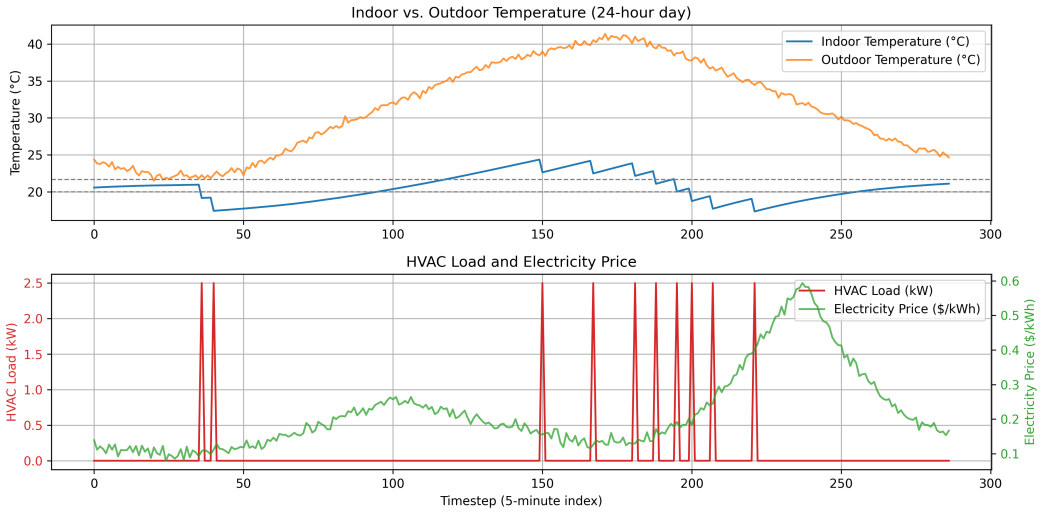

Figure 14: Price Aware HVAC Controller Results

7 Conclusion

DQN successfully learned duty-cycle HVAC behavior when trained for 500,000 timesteps with optimized hyperparameters. Among the three algorithms evaluated, DQN performed the best. Several factors likely explain this outcome. First, DQN naturally matches the discrete ON/OFF action space. Second, it handles low-dimensional problems well, making it a good fit for an environment with only

a few state variables. Additionally, DQN adapts effectively to abrupt transitions, which aligns with the threshold-style control rule (turn ON when hot, OFF otherwise). In contrast, PPO and SAC appear more sensitive to reward scaling and structure, which may have contributed to their weaker results. Overall, DQN was the most stable and promising approach for this task.

We also showed that by including a scalable reward term for electricity price, the DQN agent learned to pre-cool the building before price peaks, maintaining indoor temperature while benefiting from lower energy costs. While the observed DQN behavior does not match the bang–bang controller in strictly holding the temperature within the comfort bounds, this work demonstrates the feasibility of applying DQN to building HVAC control. Future work may involve longer multi-day training, increased environmental variability, and further reward tuning to maintain comfort more reliably.

References

[1] Tianshu Wei, Ying Wang, and Qi Zhu. Deep reinforcement learning for building HVAC control. In *Proceedings of the 54th Annual Design Automation Conference*, pages 1–6, 2017.

[2] Johnson O. Petinrin and Mostafa Shaaban. Application of open distribution system simulator (OpenDSS) in voltage control. *REPCOMSEET Journal*, 5(1), 2016.

[3] Xingbin Li, Biao Sun, Qinglong Zeng, Yang Li, and Yassine Chen. Multi-agent deep reinforcement learning based HVAC control for large office buildings. *Energy and Buildings*, 325:114574, 2025.

[4] Optimizing HVAC systems with model predictive control. *Frontiers in Energy Research*, 2025.

[5] Dongsu Kim et al. Practical challenges of model predictive control (MPC) for grid-interactive efficient buildings. Technical report, Lawrence Berkeley National Laboratory, 2024.

[6] Samuel Brandi et al. Reinforcement learning control for energy efficient buildings. *ETH Zurich Research Collection*, 2024.

[7] Zhengxuan Wang, Jie Ling, Changzhi Shao, Ming Li, Yang Yu, and Ben Xu. Multi-agent deep Q-network for multi-zone residential HVAC control. *Computer Modeling in Engineering & Sciences*, 136(3):2171–2201, 2023.

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

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, ICML '18, pages 1861–1870, 2018.

[10] L. Zhang et al. Comparison of two deep reinforcement learning algorithms towards energy-efficient control. *ETH Zurich Research Collection*, 2023.

[11] Dennis Azuatalam, Wee-Lih Lee, Frits de Nijs, and Ariel Liebman. Application of deep Q-networks for model-free optimal control balancing between frequency stability and economics. *Journal of Building Performance Simulation*, 13(4):389–407, 2019.

[12] Chi Zhang, Sanmukh R. Kuppannagari, Rajgopal Kannan, and Viktor K. Prasanna. Deep Q-network boosted with external knowledge for HVAC control. pages 184–188, 2021.

[13] Efficient and assured reinforcement learning-based building HVAC control. *Scientific Reports*, 15:91326, 2025.

[14] Farama Foundation. Gymnasium: A standard API for reinforcement learning, 2024. Accessed: 2025-12-03.

[15] J. Smith et al. Comparing building thermal dynamics models and estimation methods. *arXiv preprint arXiv:2508.09118*, 2024.

[16] M. Bahrami et al. A resistance–capacitance model for real-time calculation of cooling load in HVAC-designed spaces. *Journal of Thermal Science and Engineering Applications*, 7(4), 2015.