POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Politecnico di Torino

Master's Degree Thesis

Practical Evaluation of DDPG, TD3, and SAC for HVAC Control: A Comparative Study of Training Methods and Deployment Strategies

Supervisors
Prof. Lorenzo BOTTACCIOLI
Dott. Pietro RANDO MAZZARINO

Candidate
Filippo BERTOLOTTI

Academic Year 2024-2025

Abstract

Heating, Ventilation, and Air Conditioning (HVAC) systems account for a significant proportion of the world's total energy consumption. In recent years, research in the HVAC area has focused on developing new control systems based on artificial intelligence, particularly reinforcement learning. Reinforcement learning algorithms are well suited to the prior objective of HVAC: reducing energy consumption while maintaining good thermal comfort. However, the commercial application of such technology is still uncommon. This study aims to investigate the most effective training methods for algorithms with a view to future real-world applications. Three different reinforcement learning algorithms (DDPG, TD3, and SAC) were trained in a simulated residential environment using Energym, each in three different ways: online as is, offline, and offline with online fine-tuning. The resulting agents were compared to find the combinations that yielded the best results, such as the convergence period, loss of thermal comfort or energy during training, and, of course, the capability to perform better than a widespread controller, such as PID, during the testing phase. The study concluded that TD3 and DDPG, when trained offline with just two weeks of online fine-tuning, achieve faster convergence to an optimal trade-off between energy reduction and thermal comfort.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the contemporary context of rising energy costs and a growing focus on occupant well-being, ensuring thermal comfort within indoor environments has become imperative for maintaining optimal health and productivity. A survey spanning two decades, as conducted by Graham et al. [1], across more than one thousand buildings worldwide revealed that 39% of building occupants expressed dissatisfaction with the temperature of their workspace. In addition, Heating, Ventilation, and Air Conditioning (HVAC) systems account for approximately 40% of the annual energy consumption in the US, a trend that is reflected in much of the world [2]. Achieving an equilibrium between thermal comfort and energy efficiency constitutes a considerable challenge for the HVAC industry.

The energy consumption of an HVAC system is strongly influenced by its controller, deploying optimal control strategies could lead to significant improvements in energy savings. Currently, Rule-Based Control (RBC) is the most widespread approach, determining parameters such as temperature setpoints, heat pump power levels, and airflow rates based on fixed rules. However, RBC methods are typically static and do not adapt to varying environmental conditions, which limits their efficiency.

An alternative to RBC is represented by Model Predictive Control (MPC), which optimizes a cost function over a prediction horizon based on a thermal model of the building [3]. Although MPC can achieve better performance, it requires an accurate model of the thermal dynamic of the building, which makes its application highly specific and often impractical. In practice, different buildings require different models, and inaccuracies in modeling can lead to suboptimal and unstable control behaviors.

The extensive implementation of Artificial Intelligence (AI) in recent years has facilitated the evolution of pioneering control methodologies, particularly through the use of Deep Reinforcement Learning (DRL). The potential of DRL has been demonstrated by its remarkable performance in complex tasks, including playing

Atari games, mastering the game of Go [4], controlling robotic manipulators [5], and autonomous driving [6]. A significant benefit of DRL is its model-free nature, which enables it to learn optimal control policies directly from interaction with the environment, without requiring explicit physical or thermal models and acting as a black-box system. DRL agents are based on artificial neural networks whose weights are updated during a trial-and-error training phase. However, because of its trial-and-error nature, during the early stage of training phase agents may engage in arbitrary actions, which can result in undesired situations, for example thermal discomfort or elevated energy consumption. This behavior discourages the commercial deployment of such a control system. To address these issues, a potential strategy that has been recommended is to train DRL agents within a simulated environment prior to real-world deployment.

This thesis investigates the application and comparative performance of three reinforcement learning algorithms, Deep Deterministic Policy Gradient (DDPG), Twin Delayed DDPG (TD3), and Soft Actor-Critic (SAC), under three distinct training strategies: online learning, offline learning, and a hybrid offline-to-online fine-tuning approach. The objective of this study is to identify which combination of algorithm and training method is optimally suited to realistic HVAC control scenarios, with particular emphasis on commercial deployment.

Although previous research has explored reinforcement learning for building energy management, systematic comparisons between algorithms and training strategies remain limited. This thesis aims to fill that gap through a structured evaluation using a simulated HVAC environment built with the Energym library.

The main scientific contributions of this work are:

- A systematic comparative study of DDPG, TD3, and SAC across three training strategies (online, offline, offline with online optimization), using consistent environments, metrics, and evaluation procedures.
- An empirical analysis of agent behavior during training in different scenarios, with a focus on convergence speed, stability, and sensitivity to training conditions, offering practical insight into each algorithm's robustness and reliability.
- A quantitative evaluation of the offline with online fine-tuning approach, highlighting its potential as a safe and effective pathway for the deployment of RL agents in real-world HVAC systems by combining historical data with online adaptation.
- A discussion of real-world constraints such as comfort, safety, and computational cost to assess the feasibility of each training method in commercial applications.

- Actionable guidelines for deployment, based on the observed trade-offs between performance, data requirements, and implementation complexity.

By focusing on both performance and practical applicability, this thesis contributes to bridging the gap between reinforcement learning research and its real-world use in intelligent building control systems.

The structure of the thesis is as follows:

- **Chapter 1** – Introduces the topic and research questions addressed.
- **Chapter 2** – Describes the technologies and tools used.
- **Chapter 3** – Reviews the related work on HVAC control and energy optimization.
- **Chapter 4** – Details the adopted methodology.
- **Chapter 5** – Explains the experimental setup.
- **Chapter 6** – Presents and analyzes the results.
- **Chapter 7** – Summarizes the conclusions and outlines the future research directions.

Chapter 2

Enabling Technologies

2.1 Reinforcement learning

Reinforcement Learning (RL) is one of the main branches of machine learning, alongside supervised and unsupervised learning. Its fundamental concept is derived from behavioral psychology, where researchers explored how animals and humans learn from interaction with their environment. In particular, Thorndike's "Law of Effect" [7] states that behaviors followed by favorable consequences are more likely to be repeated, whereas behaviors followed by unfavorable consequences are less likely to recur. In the 1950s and 1960s, early computational models of learning, such as temporal-difference methods and dynamic programming, laid the groundwork for formalizing these ideas in mathematical terms. In the 1980s, algorithms like Q-learning and actor-critic methods marked a turning point in the development of RL as a distinct area within machine learning. The field gained significant momentum in the 2010s with the rise of deep reinforcement learning, especially after DeepMind's success with Deep Q-Networks (DQN) applied to Atari games, which demonstrated the power of combining neural networks with RL.

Reinforcement learning is a machine learning technique designed to solve sequential decision-making problems. Currently, it is employed in a wide range of applications, including optimization, control, monitoring, and maintenance tasks. Examples span across domains such as robotics, advertising, resource management, and web system configuration.

In RL algorithms, an agent learns how to act optimally in an environment through trial-and-error interactions, where each action executed in a specific environment configuration yields a reward which can be positive or negative, providing a quantitative evaluation of its effectiveness within that particular configuration. The interaction between the agent and the environment is typically modeled as a Markov Decision Process (MDP). An MDP is a mathematical framework for

modeling decision-making problems and is characterized by a set of states, a set of actions, a transition probability function, a reward function, and a discount factor.

Here are the main elements defining the environment in a reinforcement learning setting, typically modeled as a Markov decision process:

- **State space:** The agent observes the environment at each time step. The set of all possible configurations of the system is called the state space, denoted by *S*.

- **Action space:** The action space represents the set of possible actions that an agent can take when in a given state. It can be discrete, with a finite number of choices, or continuous. The set of available actions in the state s is denoted by *A(s)*.

- **Transition function:** The transition function $P(s' \mid s, a)$ defines the probability of the system changing into state *s'* from state s after the agent has taken action a.

- **Reward function:** The reward function quantifies the immediate benefit received after taking an action in a specific state. The environment satisfies the Markov property if the reward and the next state depend only on the current state and action, not on the sequence of past states and actions.

Algorithm 1 Basic reinforcement learning algorithm

Inizialize agent:
  Choose an initial policy $\pi(a|s)$
  Initialize value function $Q(s, a)$
Initialize environment state s
for $t = 1,2, ...T$ **do**
  Select action $a_t \leftarrow$ sample from $\pi(a_t|s_t)$
  Obtain new state and reward $s', r \leftarrow$ execute (a, s)
  Update value function
  $Q(s, a) \leftarrow Q(s, a) + \alpha\left[r + \gamma \max(Q(s', a')) - Q(s, a)\right]$
  Update policy $\pi(a|s) \leftarrow \pi(a|s) + \beta \nabla J(\theta)$
  Update current state $s \leftarrow s'$
end for

According to MDP, an RL agent receives a state s_t within a state space S, at

each time step. Subsequently, the agent selects an action a_t from an action space A, based on a policy $\pi(a_t \mid s_t)$, which defines the agent's behavior. Afterward, the agent receives a reward r_t and moves to the next state s_{t+1}, according to the transition function $P(s_{t+1} \mid s_t, a_t)$. This cycle iterates until either a terminal state is reached or a terminal condition is met.

The reinforcement learning agent aims to learn an optimal policy that allows it to select the best possible action in each state. Its objective is to maximize the expected cumulative reward, or minimize it when the reward represents a cost. The typical reinforcement learning cycle is illustrated in Fig. 2.1.

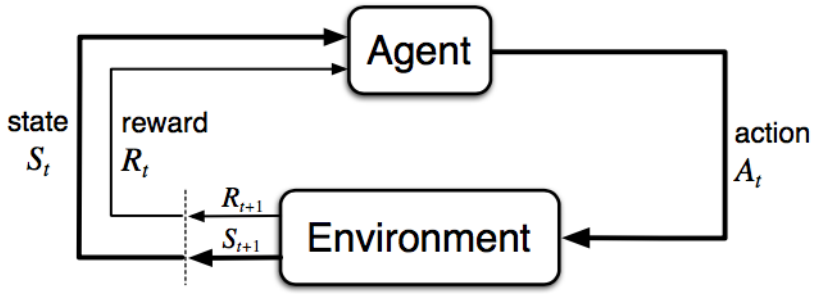

Figure 2.1: RL cycle

Balancing exploration and exploitation is crucial for the development of a sophisticated reinforcement learning agent. These two behaviors represent different strategies an agent can adopt when attempting to identify an optimal solution. Exploitation involves selecting the action that currently appears to be the best choice based on the agent's knowledge, from the agent's perspective this is a "safe" decision. However, this choice may correspond only to a local optimum, leading to suboptimal long-term results. In contrast, exploration encourages the agent to take actions that may initially seem less promising, with the aim of discovering better solutions over time. Through exploration, the agent deliberately takes risks, betting on the possibility that its current knowledge is insufficient to identify the truly optimal policy.

2.1.1 Model-free vs model-based

A key distinction among reinforcement learning algorithms is whether they are model-free or model-based. Model-free algorithms aim to directly optimize the policy or value function based on observed transitions and rewards. They do not attempt to understand the underlying dynamics of the environment, instead taking actions based on the changes and feedbacks of the environment, essentially operating as a black-box system. On the other hand, model-based algorithms

require a model of the dynamics of the environment, allowing the agent to plan ahead by using this knowledge to make informed decisions. In this thesis, we will focus on model-free algorithms.

2.1.2 Policy-based vs value-based

Model-free algorithms can pursue the goal of maximizing the reward through the optimization of either the value function or the policy. The value function predicts the expected future reward and quantifies the "goodness" of a given state or a state-action pair. The agent will then select actions that lead to states with the highest value function. When considering state-action pairs, the value function is represented by the Bellman equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max(Q(s',a')) - Q(s,a) \right] \tag{2.1}$$

These methods are referred to as value-based algorithms, such as Q-learning. In contrast, policy-based algorithms aim to directly identify the optimal policy π^* which dictates the agent's behavior by mapping states to actions. Policy-based methods update the policy without relying on a value function. Fig. 2.2 shows the taxonomy of the RL algorithms.

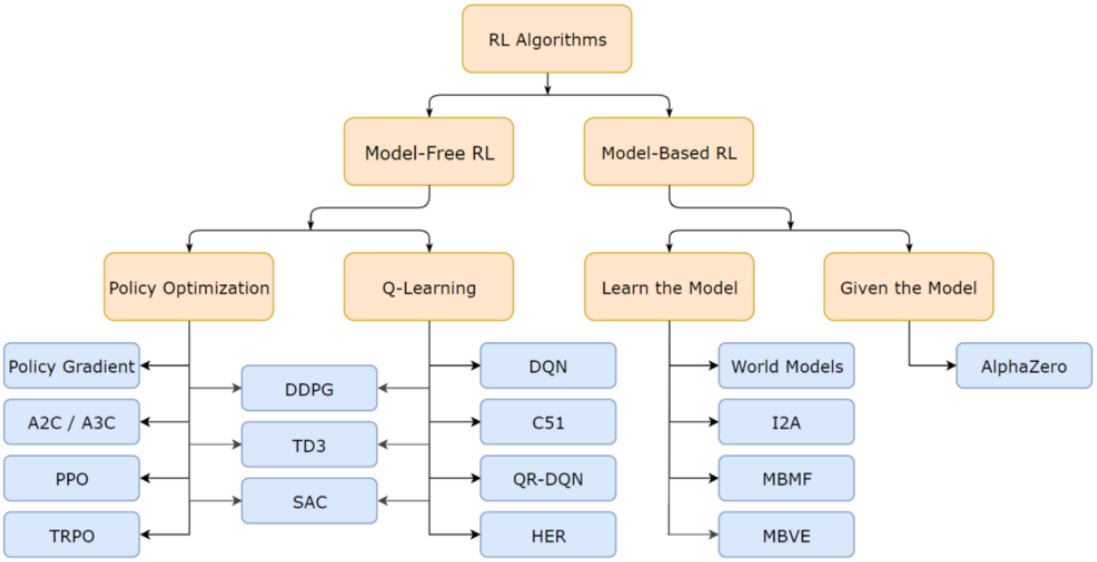

Figure 2.2: RL algorithms classification

2.1.3 Online vs offline

Reinforcement learning algorithms can be trained using two primary approaches: online and offline training.

Online training occurs when the agent interacts directly with the environment, learning through trial and error by observing the outcomes of its actions in real time. This allows the agent to continuously refine its policy based on immediate feedback, but may require significant time and resources and may not always be safe or feasible in real-world applications.

Offline training, on the other hand, involves learning from a pre-collected dataset of experiences. Each entry in the dataset typically contains a state, the action taken in that state, the resulting reward, and the next state. In this setting, the agent is not allowed to interact with the environment during training. This has the advantage of being safer and more practical in domains where live experimentation would be costly or risky (e.g., robotics, healthcare, energy systems). However, a key limitation is that the agent is constrained by the quality and diversity of the data: it cannot explore new policies beyond what is represented in the dataset. In the best cases, it ends up imitating the behavior of the agent that generated the dataset.

A common and effective strategy is to combine both approaches: using offline training to pre-train the agent on historical data, followed by online fine-tuning to adapt and optimize its behavior through exploration. This hybrid approach can significantly reduce training time and improve sample efficiency.

2.2 Deep Reinforcement Learning

The term Deep Reinforcement Learning (DRL) is used to describe a particular type of RL algorithm that employs a Deep Neural Network (DNN) to optimize the value function or the policy. The concept of Neural Network (NN) has its origins in biology; in fact, they were developed as generalizations of mathematical models of biological nervous systems [8].

A neural network is built from networks of neuron-like nodes clustered in layers. There are three types of layers: input, hidden, and output layers. In the context of an MDP problem, the input layer is responsible for describing the environment state, while the output layer corresponds to the actions to be taken in that state. The hidden layers are used to apply mathematical functions to the collected data. The neurons are connected by weighted edges, and each neuron's impulse is computed as the weighted sum of the input signals. The signal typically flows from the input layer to the output layer in a feed-forward direction. The learning capability of an NN is achieved by adjusting the weights θ between the nodes through optimization algorithms such as stochastic gradient descent. Typically, a neural network is

considered deep when it has more than two hidden layers.

The adoption of DNNs in reinforcement learning allows agents to approximate complex functions, enabling them to handle high-dimensional state or action spaces that would be intractable with traditional tabular methods. This synergy between deep learning and reinforcement learning has led to impressive results in fields such as robotics, autonomous vehicles, and game playing. An example of a deep neural network is shown in Fig. 2.3.

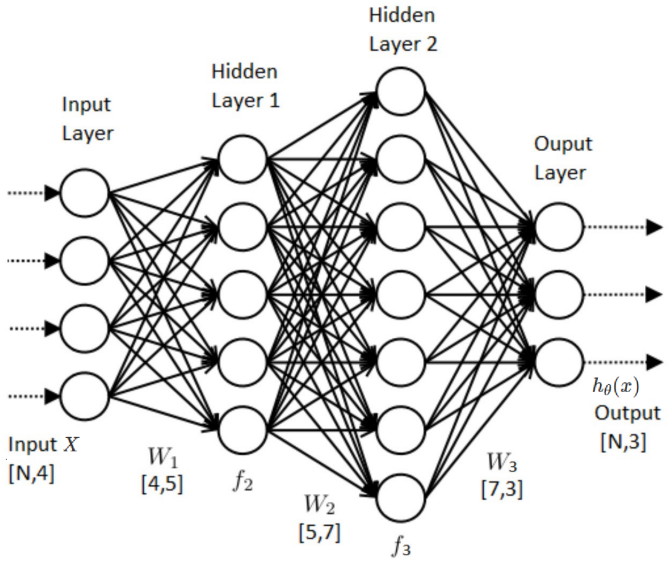

Figure 2.3: Deep neural network structure

Many DRL algorithms have been developed over time, including DQN, PPO, TD3, SAC, and DDPG. While they are classified by the macro-charachteristics described previously, they also differ in their policy for updating the weights between the nodes of neural networks. However, all of them share some important parameters that influence the agent's learning strategy, and these parameters are summarized in Tab. 2.1

2.2.1 DDPG

The first reinforcement learning algorithm examined in this study is the Deep Deterministic Policy Gradient (DDPG). The algorithm was first introduced in 2015 by Lillicrap et al. [9] and is capable of dealing with a continuous action space. DDPG is a hybrid algorithm that combines elements of policy-based and value-based methods. The algorithm employs an actor-critic technique, meaning it is split into two models: the actor and the critic. Each of these comprised two

Parameter	**Description**
Learning rate (λ)	Controls how much the weights of the network are adjusted with respect to the loss gradient.
Discount factor (γ)	Determines the importance of future rewards in the optimization process.
Buffer size	Size of the replay buffer, where experienced data are stored.
Batch size	Size mini-batch data on which gradient loss is calculated.
Soft target update coefficient (τ)	Determines the rate at which the target network is updated towards the online network, improving stability.

Table 2.1: DRL algorithms parameters

distinct neural networks, resulting in a total of four NN. The actor network $\mu(s \mid \theta^{\mu})$ is policy-based and takes actions according to the current state and updates the policy in the direction suggested by the critic. The critic network $Q(s, a \mid \theta^{Q})$ is a value-based network and estimates the value function and evaluates the actor's behavior. DDPG stores experiences in a replay buffer R, from which samples are drawn to update the NN parameters. It is essential that the replay buffer maintains a balance between old and new experiences in order to guarantee stability. At each weights update, the algorithm samples a set of N experiences from the replay buffer, N is equal to the dimension of the mini-batch. The loss function for the critic and the actor network is then calculated. The objective of both models is to minimize the respective loss function. The loss function for the critic and the actor is as follows:

$$J_Q = \frac{1}{N}\sum_{i=1}^{N}\left(r_i + \gamma(1-d)Q_{\text{targ}}(s_i', \mu_{\text{targ}}(s_i')) - Q(s_i, \mu(s_i))\right)^2 \tag{2.2}$$

$$J_\mu = \frac{1}{N}\sum_{i=1}^{N} Q(s_i, \mu(s_i)) \tag{2.3}$$

In equations (2.1) and (2.2), Q_{targ} and μ_{targ} represent the target networks of the

two models. Both the critic target network and the actor target network are similar to the main networks, but they update their weights at a slower rate because they provide a more stable estimate.

Algorithm 2 DDPG

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor network $\mu(\text{s}|\theta^\mu)$ with weights θ^Q and θ^μ
Initialize target networks Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for *episode* = 1,2, ...M **do**
    Receive the initial environment state s_0
    for t = 1,2, ...T **do**
        Select action $a_t = \mu(s_t|\theta^\mu)$ according to the current policy
        Execute action a_t, observe reward r_t and new state s_{t+1}
        Store transitions (s_t, a_t, r_t, s_{t+1}) in R
        Sample a random minibatch of N transitions (s_t, a_t, r_t, s_{t+1}) from R
        Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
        Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
        Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i} \tag{2.4}$$

        Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \tag{2.5}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'} \tag{2.6}$$

    end for
end for

2.2.2 TD3

Twin Delayed DDPG (TD3) is a reinforcement learning algorithm built on DDPG [10] that addresses its biggest limitation, the overestimation of the Q-values. TD3 can only be used for environments with continuous action spaces. The "Twin" in the name refers to the fact that the critic networks are two. TD3 simultaneously

learns two distinct Q-functions Q_{ϕ_1} and Q_{ϕ_2}, the target is then calculated using the smaller Q-value. The actor network in TD3 has the same function as in DDPG: it takes the current state and returns a deterministic action. The two critic networks $Q_{\phi_1}(s,a)$ and $Q_{\phi_2}(s,a)$ are trained independently to estimate the value of the action taken in a specific state. When computing the target value, the algorithm uses the minimum between the two Q-values, mitigating the overestimation bias typically found in actor-critic methods. The target value is calulated as follows:

$$y = r + \gamma(1-d)\min_{j=1,2} Q_{\phi_j}^{\text{targ}}(s', \mu^{\text{targ}}(s') + \epsilon) \tag{2.7}$$

$\epsilon \sim \text{clip}(\mathcal{N}(0,\sigma), -c, c)$ is a clipped random noise added to the target action to implement target policy smoothing. The loss function of the critic is then computed as

$$J_Q = \frac{1}{N}\sum_{i=1}^{N}\Big(y_i - Q_{\phi_j}(s_i,a_i)\Big)^2 \tag{2.8}$$

The actor policy is updated less frequently than the critic (every d steps, with $d > 1$) and aims to maximize the estimated Q-value of the first critic network:

$$J_\mu = -\frac{1}{N}\sum_{i=1}^{N} Q_{\phi_1}(s_i, \mu(s_i)) \tag{2.9}$$

As in DDPG, TD3 uses target networks for both the actor and the critic. These target networks are updated using a soft update rule to ensure training stability. In total, TD3 employs six neural networks: two critics, one actor, and their respective target networks. This architecture, combined with delayed policy updates and target smoothing, leads to more stable and reliable learning in environments with continuous action spaces.

Algorithm 3 TD3

Initialize policy parameters θ, Q-function parameters ϕ_1, ϕ_2, replay buffer $\mathcal{R}$
Set target parameters equal to main parameters: $\theta_{\text{targ}} \leftarrow \theta$, $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$
repeat
    Observe state s and select action $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{\text{Low}}, a_{\text{High}}), \quad \epsilon \sim \mathcal{N}$
    Execute action a, observe next state s' and reward r
    Store (s, a, r, s', d) in replay buffer $\mathcal{R}$
    if it's time to update **then**
        for j in range(however many updates) **do**
            Sample a batch of transitions $B = \{(s, a, r, s')\}$ from $\mathcal{R}$
            Compute target actions:

$$a'(s') = \text{clip}(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{\text{Low}}, a_{\text{High}}), \quad \epsilon \sim \mathcal{N}(0, \sigma) \tag{2.10}$$

            Compute targets:

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', a'(s')) \tag{2.11}$$

            Update Q-functions by one step of gradient descent:

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_{\phi_i}(s, a) - y(r, s', d)\right)^2 \quad \text{for } i = 1, 2 \tag{2.12}$$

            if j mod policy_delay $= 0$ **then**
                Update policy by one step of gradient ascent:

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_{\phi_1}(s, \mu_\theta(s)) \tag{2.13}$$

                Update target networks:

$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho)\phi_i \quad \text{for } i = 1, 2 \tag{2.14}$$

$$\theta_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1 - \rho)\theta \tag{2.15}$$

            end if
        end for
    end if
until convergence

2.2.3 SAC

Soft Actor-Critic (SAC) is a reinforcement learning algorithm designed to achieve high sample efficiency and stability by maximizing the trade-off between expected return and entropy [11]. SAC is specifically intended for environments with continuous action spaces. Unlike deterministic algorithms like DDPG and TD3, SAC employs a stochastic actor, meaning that the policy outputs a probability distribution over actions instead of a single deterministic action.

The "Soft" in the name refers to the inclusion of an entropy term in the objective function, encouraging exploration by penalizing certainty in the action choices. This mechanism allows the agent to balance exploration and exploitation more effectively. SAC maintains two critic networks $Q_{\phi_1}(s, a)$ and $Q_{\phi_2}(s, a)$, similar to TD3, and uses the minimum of the two Q-values to reduce positive bias in critic estimates. The target value is calulated as follows:

$$y = r + \gamma(1 - d)\left(\min_{j=1,2} Q_{\phi_j}^{\text{targ}}(s', a') - \alpha \log \pi(a'|s')\right) \tag{2.16}$$

where $a' \sim \pi(\cdot|s')$ and α is the temperature parameter that controls the importance of the entropy term.

The critic loss function is defined as:

$$J_Q = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q_{\phi_j}(s_i, a_i)\right)^2 \tag{2.17}$$

The actor policy is optimized to maximize not just the expected Q-value but also the entropy of the policy itself:

$$J_\pi = \frac{1}{N}\sum_{i=1}^{N}\left(\alpha \log \pi(a_i|s_i) - Q_{\phi_1}(s_i, a_i)\right) \tag{2.18}$$

SAC also uses target networks for the critics, updated via a soft update mechanism to ensure stability during training. In total, SAC employs five neural networks: two critics, one actor, and two target critics. The combination of entropy maximization, double Q-learning, and soft target updates makes SAC one of the most robust and efficient algorithms for continuous control tasks.

Algorithm 4 SAC

Initialize policy parameters θ, Q-function parameters ϕ_1, ϕ_2, temperature parameter α, replay buffer $\mathcal{R}$
Set target Q-function parameters: $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$
repeat
    Observe state s and sample action $a \sim \pi_\theta(\cdot|s)$
    Execute action a, observe next state s' and reward r
    Store (s, a, r, s', d) in replay buffer $\mathcal{R}$
    if it's time to update **then**
        for j in range(however many updates) **do**
            Sample a batch of transitions $B = \{(s, a, r, s')\}$ from $\mathcal{R}$
            Sample actions and compute log probabilities:

$$a'(s') \sim \pi_\theta(\cdot|s'), \quad \log \pi_\theta(a'(s')|s') \tag{2.19}$$

            Compute target values:

$$y(r, s', d) = r + \gamma(1 - d) \left(\min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', a'(s')) - \alpha \log \pi_\theta(a'(s')|s') \right) \tag{2.20}$$

            Update Q-functions by one step of gradient descent:

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_{\phi_i}(s, a) - y(r, s', d) \right)^2 \quad \text{for } i = 1, 2 \tag{2.21}$$

            Update policy by one step of gradient ascent:

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left(\alpha \log \pi_\theta(a|s) - Q_{\phi_1}(s, a) \right), \quad a \sim \pi_\theta(\cdot|s) \tag{2.22}$$

            Adapt temperature α (optional, if automatic entropy tuning is used):

$$\nabla_\alpha \frac{1}{|B|} \sum_{s \in B} -\alpha \left(\log \pi_\theta(a|s) + \mathcal{H} \right) \tag{2.23}$$

            Update target networks:

$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho) \phi_i \quad \text{for } i = 1, 2 \tag{2.24}$$

        end for
    end if
until convergence

2.3 d3rlpy

In this work, the d3rlpy library (Deep Deterministic Dataset-driven Reinforcement Learning in Python) [12] was adopted as the main framework for implementing, training, and evaluating reinforcement learning algorithms. d3rlpy is an open-source library built on top of the PyTorch and scikit-learn libraries, providing a high-level and user-friendly interface specifically designed to support both offline and online reinforcement learning.

Initially, the project started with the use of Stable-Baselines3 [13], a widely adopted library for reinforcement learning. However, it became clear that its focus on online learning and lack of native support for offline training made it less suitable for the goals of this thesis. For this reason, the decision was made to switch definitively to d3rlpy, which offers built-in compatibility with offline training workflows.

Moreover, d3rlpy allows one to easily save and load the trained agents and to transfer a trained policy from one algorithm to another. The library enables consistent training workflows, efficient hyperparameter tuning, and a standardized collection of performance metrics. Integration with monitoring tools such as TensorBoard allows real-time visualization of training progress and model evaluation.

In general, d3rlpy has proved to be a key enabling technology for this work, facilitating reproducibility, flexibility in experimentation, and rigorous benchmarking of offline RL algorithms.

2.4 OpenAI Gym

OpenAI Gym [14] is an open-source library that is widely used and provides a standardized interface for reinforcement learning environments. It was developed to facilitate the development, sharing, and benchmarking of RL algorithms by offering a collection of pre-built environments (ranging from classic control problems to Atari games) along with a simple and consistent API. This API defines how agents interact with environments through functions such as reset() to start a new episode, step(action) to take an action and observe the result, and render() to visualize the current state. By defining clear conventions for the interaction between agents and environments, Gym has become a foundational component in the RL research community and continues to influence the design of many modern RL frameworks and toolkits.

Although the experiments in this thesis were conducted in a custom-built simulation environment tailored to HVAC control, the structure and design of the environment were inspired by the Gym interface. Following this standard allowed for smoother integration with RL libraries like d3rlpy, and made the development

of training and evaluation pipelines more modular and easier to maintain. In the initial phase of the study, the library was utilized to facilitate familiarization with the world of reinforcement learning. OpenAI Gym provides a selection of basic environments, including "cartpole" and "pendulum", which are useful in evaluating the capabilities of the library and facilitating an understanding of the operational principles of an RL algorithm.

2.5 Energym

Energym [15] is a Python-based open-source library designed for the testing and development of controllers for HVAC systems in simulated building environments. It has been developed with the objective of facilitating research on energy consumption optimization, providing a user-friendly framework that is particularly suited for the implementation of advanced control strategies. The library includes a variety of simulation models, based on the widely adopted platforms Modelica or EnergyPlus. These models vary in terms of building size, number of rooms, installed technical systems, and available control points. Each environment is associated with different weather profiles, which dynamically affect the simulation conditions.

Energym is especially suited to the development of Reinforcement Learning (RL) agents, offering an interface closely resembling that of the well-known OpenAi Gym library. The environment follows the standard structure of returning observations, rewards, termination signals, and additional information at each simulation step, ensuring compatibility with most RL frameworks. As an open-source project, it is under continuous development and improvement, driven by contributions from its active community.

Chapter 3

Literature Review

The objective of this chapter is to provide an overview of the scientific contributions related to the development of DRL algorithms in HVAC systems. Considering the complexity of an HVAC system, many different applications have been proposed in the last years. This review takes into account the diverse applications of the algorithms, without focusing on a particular aspect of the heating and cooling systems. Each scientific paper is classified according to macro categories, including the section of the HVAC system that has been controlled, the control's action space (discrete or continuous), and the algorithm that has been employed. The study reveals that the most prevalent algorithm is DQN, an algorithm specific to controllers with a discrete action space. For controls with a continuous action space, the algorithms are more heterogeneous. Each paper gives great priority to reducing energy consumption while maintaining thermal comfort. In some cases, additional objectives, which are not marginal, are also addressed. The studies used to build the state-of-the-art are classified in Tab. 3.1

Initially we will focus on those studies that implement a discrete action space algorithm. Heidari et al. [16] propose a discrete control strategy for the heat pump, which simply alternate ON/OFF, with the objective of achieving a balance between energy consumption, hygiene, and comfort in the context of water heating systems. The DDQN algorithm, tested in real life during COVID home period, achieved a reduction in energy consumption of 24.5% over the deployment phase, despite the relatively simple action space. Wei et al. [17] control the airflow rate of different zones of temperature, training and testing a DQN algorithm in a simulative environment modeled with EnergyPlus. In their study, Sakuma and Nishi [18], focus on a specific aspect of air conditioning control: the direction of airflow. They identify three actions, labelled as left, center and right. The performances of the DRL controller outperform a rule-based method, achieving a reduction in energy consumption by 35% on average. Zhiang and Khee [19] implement a DRL-based optimal control method for a radiant heating system in an office building. The

real-life test yields optimal results, however, they suggest more investigation into the problem of the delayed rewards caused by the slow thermal response of the HVAC system. Lork et al. [20] concentrated on the development of an automated, data-driven AC controller capable of regulating the airflow setpoint temperature with a power saving percentage of 3.6%, in comparison to a rule-based control scheme. By aggregating the training data from multiple rooms in a neural network framework, the issue of overfitting and data imbalance can be addressed. McKee et al. [21] seek to regulate the AC of a simulated house comprising two distinct zones. When a decision is required, the controller considers the current energy price and the projected energy price over the subsequent thirty minutes. The system, subject to certain constraints, is capable of reducing energy consumption while maintaining thermal comfort for occupants. In Overgaard et al. [22], a Q-learning algorithm is trained in simulation for a period of five months in order to learn how to control a mixing loop. The results demonstrate that after 55 days of training, the proposed controller reached the same performances of an industrial controller. Furthermore, after five months of training, the controller was capable of reducing costs by 20.5%. In Solinas et al. [23], an adaptive controller is developed based on a system identification model representing the thermodynamic response of a target building, utilising historical data. The controller has a discrete action space, represented by the setpoint temperature at which the air is heated before being supplied to the room. The DDQN algorithm with different values of its parameters is finally compared with a rule-based controller and a MDP controller. The comparison demonstrates that the DRL controller outperforms the two baselines in both the thermal comfort and energy consumption. Yu et al. [24] conducted a training session for a DQN agent during lesson period in a university classroom. During the training phase, the algorithm considers the varying metabolic rates of students, which are higher during examinations and lower during lessons, in order to calculate the PMV (predicted mean vote), which is an estimation of the thermal comfort of occupants. The agent has been trained and tested in a real-life environment, resulting in a reduction of 13% in the average carbon dioxide concentration. The objective of the study conducted by Kazmi et al. [25] was to develop a DRL agent for the purpose of controlling a hot water system. The deployment of the controller in 32 houses in the Netherlands has resulted in a reduction in energy consumption for the production of hot water by 200 kWh per annum, on average, for a single house. In their study, Peirelinck et al. [26] focus on the optimization of an electric water heating system with through the utilisation of a RL algorithm. A secondary objective is to minimize the initial learning time, with the aim of avoiding a period of thermal discomfort and high energy consumption due to the random behavior of the algorithm. The proposed solution is to pre-train the agent's policy offline and then deploy it online. The results demonstrate that the pre-training phase ensures an 8.8% cost reduction

compared to starting the learning phase from scratch. Mocanu et al. [27] investigate the possibility of applying a DRL agent within the context of a smart grid for a smart home energy management system. Additionally, the researchers focus on the distinctions between an agent starting from scratch in an online setting and one that is trained offline. In particular, the offline training phase initially prevails over the online training, yet the performance of both agents reaches a point of equivalence after approximately 80% of the total episodes.

We will now examine studies that developed a controller with a continuous action space. The study conducted by Gao et al. [28] illustrates that algorithms requiring a continuous action space are more efficient than those requiring a discrete action space. This due to the fact that the former do not need the discretization of the potential control range and require less training data. The DDPG algorithm was compared with some discrete algorithms, such as DQN, Q-learning and SARSA resulting in a thermal comfort improvement by 13.6%, 17.6% and 8.6% respectively. Bo et al. [29] attempted to reduce the training time by developing a multi-agent framework. Each agent in this framework controls the airflow rate of a specific zone of a simulated building. During the training phase, the agents learn from each other's observations through a process of centralised training. However, when performing an action, they rely only on the local state (distributed execution). The proposed method achieve comfortable thermal conditions while reducing costs. furthermore, it has demonstrated good convergence and stability. Sometimes it may be helpful to integrate the DRL algorithm with an MPC model, as demonstrated by Fu and Zhang [30]. In their study, they utilise an MPC algorithm to generate the input for a TD3 algorithm, in order to define the optimal temperature setpoint. The TD3-MPC algorithm produces superior results compared to the conventional DDPG due to of its capacity to delay policy updates, thus reducing errors and preventing the emergence of suboptimal policies. Silvestri et al. [31] seek to bridge the gap between the simulated environment and the real world. The proposed work investigates the deployment of an RL algorithm, which has been pre-trained offline in a simulative environment, into a real building with real occupants interacting with the environment, such as opening doors and windows. The results obtained from this deployment are then compared against those obtained from the deployment of more popular PI and RBC algorithms. The resulting data confirm the superiority of DRL algorithms over the other competitors and illustrate the potential of these algorithms to be effectively deployed in real-world scenarios without the necessity for invasive technical operations. In a recent study, Schmitz et al. [32] implemented a DRL controller for a residential heat pump in Germany. The controller does not require a building model. researchers proposed two distinct reward functions that consider the variable energy price over time. The results are based on space heating data exclusively. The authors recommend that future studies would incorporate also domestic hot water usage. Delgoshaei et al. [33] conducted research on a

optimal controller for a component of the hydronic system of a laboratory at the NIST (National Institute of Standards and Technology). The DRL agent employed an actor-critic algorithm on a continuous action space that involved the control of a valve regulating the air flow from the thermal energy storage. They determined that historical experimental data for offline training required an extensive preprocessing phase to reduce noise and enhance convergence. In 2020, Yu et al. [34] published their study on the use of DRL for the optimization of smart home energy management, with a particular focus on the charging and discharging phases of the energy storage system powered by renewable energy sources. The DDPG algorithm, which is entirely model-free, has been trained and tested offline on a dataset that includes the variable electricity price. The results demonstrate that the proposed algorithm achieves better performance than the two baselines used for evaluation.

Study	**Control actions**	**Algorithm**	**Training notes**	**Testing notes**
Heidari et al. [16]	Heat pump ON/OFF (discrete)	DDQN	Offline on TRNSYS model	Real life office for 14 weeks
Wei et al. [17]	Airflow rate of different zones (discrete)	DQN	100 month of training over a simulation environment modeled in EnergyPlus	Three buildings with one, four and five zones modeled in EnergyPlus
Sakuma and Nishi [18]	Direction of airflow (discrete)	DQN	Simulated on CFD software	Simulated on a residential building with large rooms
Zhiang and Khee [19]	Radiant heating system supply water temperature setpoint (discrete)	A3C	Offline on a simulated model (EnergyPlus)	Real life office building for three months
Lork et al. [20]	Airflow setpoint temperature (discrete)	DQN	Trained on historical weather conditions and user ON/OFF sequences	Evaluated on real data, no real world test
McKee et al. [21]	Air conditioning ON/OFF of two zones (discrete)	DQN	Trained in a simulated environment	No real world test
Overgaard et al. [22]	Mixing loop temperature setpoint and pump speed (discrete)	Q-learning	5 months of offline training over data gathered on an office building	Tested on a mixing loop hardware combined with a building model
Solinas et al. [23]	Temperature setpoint at which the air is heated before being supplied to the room (discrete)	DDQN	Offline on three month of building and weather historical data generated with EnergyPlus	Tested over the same simulated building with different weather data
Yu et al. [24]	Room temperature and fresh air flow rate (discrete)	DQN	30 days of experiments in a campus classroom with 4 fans and three conditioners	Tested in the same classroom for one month during summer
Kazmi et al. [25]	Vector of ON/OFF actions (discrete)	Deep PILCO	Trained directly online on real houses	Tested on 32 real houses, 13 with rule-based controller and 19 with the proposed framework
Peirelinck et al. [26]	Electric water heating system ON/OFF (discrete)	DDQN	Offline pre-training	Tested in simulation

Study	**Control actions**	**Algorithm**	**Training notes**	**Testing notes**
Mocanu et al. [27]	Three devices ON/OFF (discrete)	DQN and DPG	Trained offline and online on a simulated environment	Tested on training environment
Gao et al. [28]	Temperature and humidity setpoints (continuous)	DDPG	Offline over an HVAC system simulated with TRNSYS, one year of simulation data	5000 hours for testing performances, no real-world test
Bo et al. [29]	Airflow rate of different zones (continuous)	Multi-Agent TD3	Simulated building on TRNSYS, episode corresponds to one day, 15 minutes slot	7 random days of the training environment
Fu and Zhang [30]	Temperature setpoint (continuous)	TD3-MPC	100 episodes (1 day each) on simulated building with five zones	Tested in the same environment of training phase
Silvestri et al. [31]	Percentage of valve opening in a cooling system (continuous)	SAC	Trained on a simulated RC system, 30 episodes (91 days each)	Real-life building
Schmitz et al. [32]	Heat pump power (continuous)	PPO	Trained offline on 5 years of data with timeslot of 15 minutes on	Tested on simulation
Delgoshaei et al. [33]	Mixing valve position (continuous)	Actor-critic	Trained on historical data	Tested in a laboratory
Yu et al. [34]	Energy storage system charging/discharging power and HVAC input power (continuous)	DDPG	Trained offline over two summer months of data	Tested offline on one summer month of data

Table 3.1: RL studies taxonomy

Chapter 4

Methodology

This section presents the system model and the problem formulation.

4.1 Problem definition

The main objective of this work is to evaluate and compare different approaches for applying Reinforcement Learning (RL) algorithms to the control of HVAC systems, with the ultimate goal of enabling their deployment in real-world and commercial settings. HVAC control presents a complex problem that involves the trade-off between energy efficiency and thermal comfort, making it a suitable candidate for optimization strategies such as RL.

In this context, RL agents are trained and tested in a simulated environment that replicates the thermal behavior of a building equipped with a heat pump and a radiator. The agents' goal is to modulate the output power of the heat pump to maintain a desired indoor temperature profile, specifically 289.15K (16°C) during nighttime hours and 293.15K (20°C) during daytime, while minimizing energy consumption.

The benchmark for comparison is a traditional Proportional-Integral-Derivative (PID) controller, which represents a widely adopted and well-understood control strategy in the HVAC industry. The effectiveness of each RL approach is, therefore, assessed by its ability to achieve lower energy consumption than the PID controller, without sacrificing thermal comfort.

The study investigates not only the performance of different RL algorithms, but also the impact of various training methodologies, including offline training based on historical data, online training through interaction with the environment, and hybrid strategies that combine both. By comparing these approaches, the aim of the work is to highlight their respective strengths, limitations, and potential applicability in real-world deployment scenarios.

4.2 Simulation environment

The building model used in this study is the "Simple-HouseRad" model from Energym, developed in Modelica. This environment simulates a single thermal zone that represents a poorly insulated American-style house. The HVAC system includes a water-to-water heat pump connected to a radiator. This type of heat pump transfers heat between two water loops, a source side and a load side, and is commonly used in both heating and cooling applications. However, for the purpose of this study, only the heating mode is considered due to the limitations of the simulator. A schematic diagram of the system is shown in Fig. 4.3.

The simulation operates with a time step of 5 minutes, ensuring a fine temporal resolution suitable for control applications.

To integrate this system with d3rlpy reinforcement learning algorithms, which require environments that conform to the OpenAI Gym interface, the Energym environment was wrapped as a custom Gym environment. This involved implementing standard Gym methods like reset() and step(action), as well as custom utility functions such as get_obs() and get_reward(), defining the observation space and reward logic. This adaptation enabled direct integration of Energym within th d3rlpy training and evaluation pipelines, ensuring compatibility and simplifying experimentation.

Accurate modeling of the thermal behavior of the building requires realistic external weather data. For this purpose, five weather files in Modelica's standard .mos format were obtained from NASA's POWER (Prediction Of Worldwide Energy Resources) Data Access Viewer. These files represent the climate of Aosta, Italy, for the years 2019 to 2024, and were sourced from the NASA Langley Research Center (LaRC) POWER Project, funded by the NASA Earth Science / Applied Science Program.

The weather datasets were used as follows:

- 2019 was reserved exclusively for testing.
- 2020–2024 were used for training.

Since Energym only simulates the heating, only the months of January, February, March, October, November, and December were considered for each year. This ensures that the agent is trained and evaluated in periods when the heating demand is significant.

Because only six months per year are usable, a "training year" in this context refers to two combined heating seasons from different calendar years (e.g., the six months of 2021 joined with those of 2022). The heating and cooling seasons are illustrated in Fig. 4.2.

Figure 4.1: Environment framework

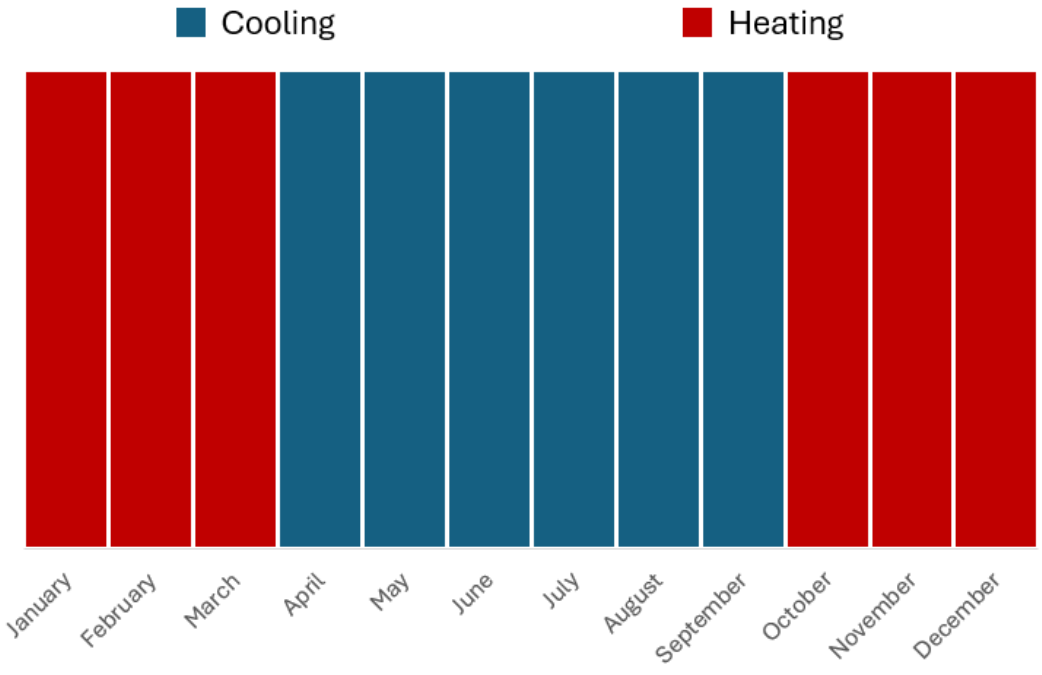

Figure 4.2: Heating and cooling seasons

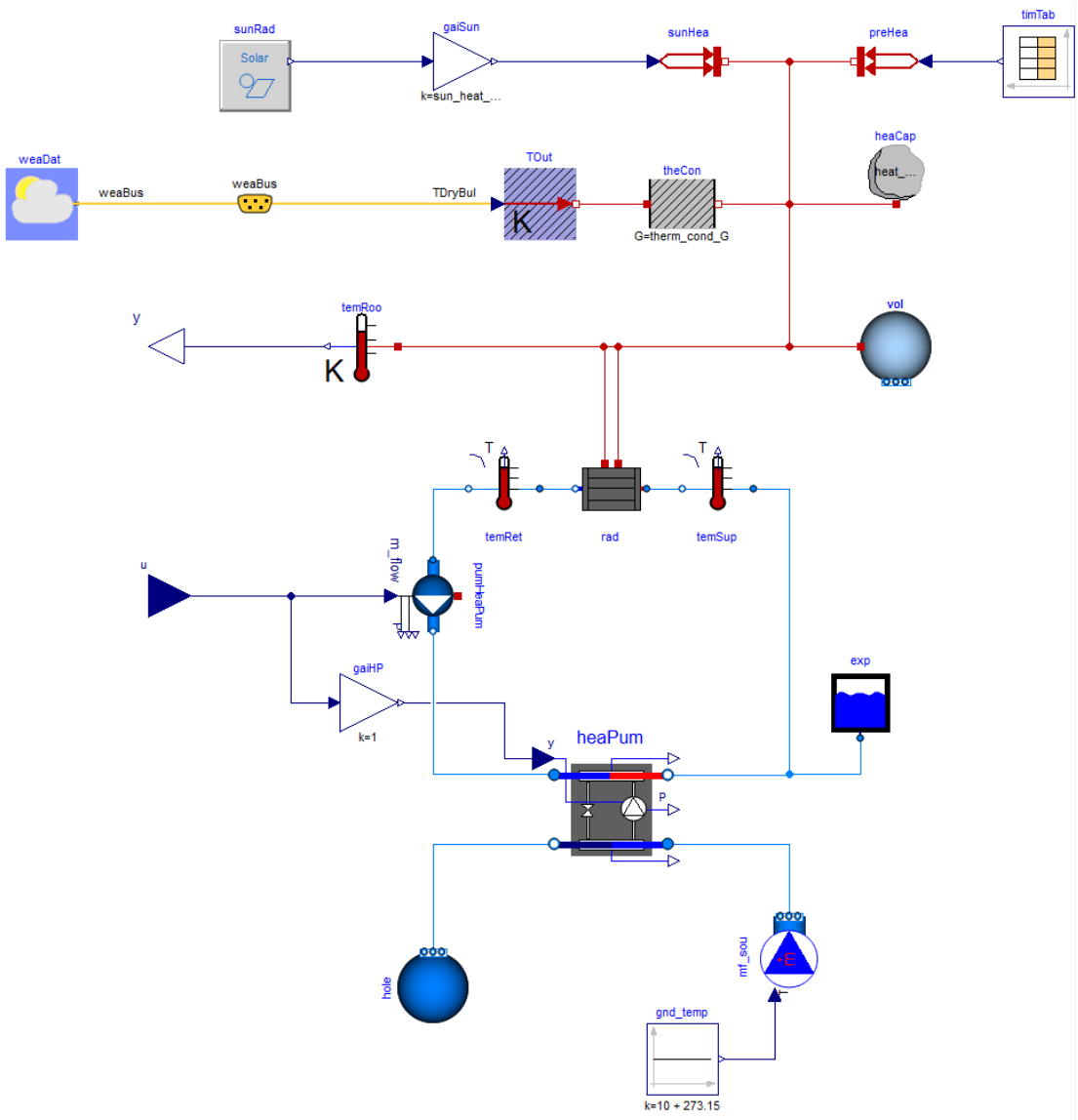

Figure 4.3: *SimpleHouseRad* heating model

4.3 PID

As a baseline to evaluate the performance of reinforcement learning agents, a traditional Proportional-Integral-Derivative (PID) controller [35] was implemented. The PID controller serves as a well-established benchmark in control systems due to its simplicity, robustness, and widespread use in HVAC applications. It operates by continuously computing the error between the current indoor temperature and a desired setpoint and adjusting the heating power accordingly to minimize this error.

The controller output is the sum of three components:

- A proportional term K_p that reacts to the current error.
- An integral term K_i that accumulates past errors.
- A derivative term K_d that predicts future error trends.

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)\, d\tau + K_d \cdot \frac{de(t)}{dt} \tag{4.1}$$

For this study, the PID parameters were manually adjusted to provide a reasonable balance between comfort and energy efficiency in the simulation environment. Specifically, the proportional gain was set to $K_p = 0.1$, the integral gain to $K_i = 0$, and the derivative gain to $K_d = 220$. These values prioritize fast and anticipative responses (through a strong derivative component) while avoiding integral wind-up, given the slow dynamics of the thermal system.

The PID controller was used to generate the datasets for offline training and as a reference to assess the relative improvement of the trained agents in terms of comfort and energy consumption.

4.4 Offline training dataset

The datasets used for offline training were generated by running the PID controller across multiple simulated heating seasons. These simulations produced diverse state-action-reward trajectories, which were recorded and later used to train the offline agents. To assess the effect of data availability, datasets of varying lengths were collected, allowing an analysis of each algorithm's learning capabilities under constrained versus abundant data conditions.

The data collected reflect a wide range of operational scenarios, including different outdoor temperatures and heating demands. This diversity is crucial for training robust policies capable of generalizing to unseen conditions. Using this consistent and reproducible dataset, it is possible to evaluate the performance of

offline agents in a controlled yet realistic setting, without requiring live interaction with the environment.

4.5 MDP formulation

At each time step t, the environment is fully characterized by the current conditions, without dependence on past states, thus satisfying the Markov property. Consequently, the control problem for the building's HVAC system can be formulated as an MDP. The main components of this formulation are detailed below and resumed in Tab. 4.1.

4.5.1 Action space

The control input for the HVAC system is the setpoint of the heat pump power fraction, denoted by u. This continuous control variable is normalized between 0 and 1, representing the fraction of the maximum heat pump power to be applied. A value of *0W* corresponds to no heating, while *1* corresponds to the maximum power *5000W*. An increase in the heat pump power results in a higher water supply temperature to the radiator, enhancing the system's heating capacity.

$$A_t = (u) \tag{4.2}$$

4.5.2 State space

As stated above, the state constitutes the input of the DRL agent. In this research, the optimal control output is determined based on the following observations: the current heat pump power P which is equal to the previous control taken, the temperature of the water supplied to the radiator T_s, the outdoor air temperature T_o, the temperature difference ΔT between the room setpoint temperature and the current room temperature. The first three observed values are normalized between 0 and 1 while the ΔT has a value between $243.15K$ and $303.15K$.

$$S_t = \begin{pmatrix} P\,[W] \\ T_s\,[K] \\ T_o\,[K] \\ \Delta T\,[K] \end{pmatrix} \tag{4.3}$$

4.5.3 Reward function

The reward function in this study is designed as a penalty to be minimized by the DRL agent. Specifically, the reward at each time step is calculated as the

negative weighted sum of two terms, the heat pump power consumption $P[W]$ and the temperature deviation $\Delta T[K]$. The reward is defined as follows:

$$r_t = -\left(0.6 \times P + 2.4 \times |\Delta T|\right) \tag{4.4}$$

where the *weight_e* 0.6 and the *weight_t* 2.4, were optimized using the Optuna hyperparameter optimization library [36]. The optimization process involved 280 trials, each of which an agent was trained over a training year. After training, each trial was evaluated on the basis of the normalized sum of total energy consumption and temperature deviation. The objective was to identify the weight combination that led the agent to achieve the best trade-off between minimizing energy usage and maintaining thermal comfort. The purpose of this reward structure is twofold: to encourage energy-efficient operation of the HVAC system by minimizing power consumption and to maintain thermal comfort by reducing deviation from the desired indoor temperature. By penalizing both excessive energy use and large temperature errors, the agent learns to balance comfort requirements with energy efficiency. Fig. 4.4 shows a heatmap of the optimization results obtained with Optuna.

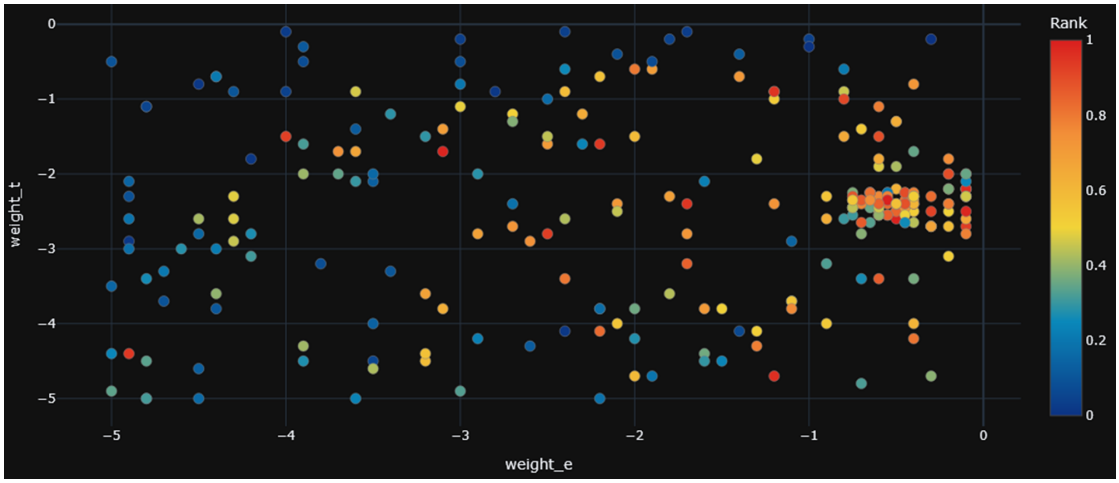

Figure 4.4: Reward function weights heatmap

Table 4.1: Summary of action, state, and reward

Element	**Definition**	**Range**
Action A_t	Heat pump power fraction u	$[0, 1]$
State S_t	Heat pump power P [W]	$[0, 1]$
	Supply water temperature T_s [K]	$[0, 1]$
	Outdoor air temperature T_o [K]	$[0, 1]$
	Temperature difference ΔT [K]	$[-30, 30]$
Reward r_t	$-(0.6 \times P + 2.4 \times \lvert\Delta T\rvert)$	$(-\infty, 0]$

Chapter 5

Experimental Setup

This chapter outlines the methodology adopted to carry out the experiments whose results will be presented and analyzed in the final part of this thesis. The objective is to define a clear and reproducible experimental protocol for evaluating different reinforcement learning strategies in the context of HVAC control.

The chapter is organized according to the three training paradigms considered: online, offline, and offline-to-online learning. For each of these, we describe how the agents were trained, how the datasets (if any) were generated, and under what conditions the agents were evaluated. Common settings such as the simulation environment, episode structure, evaluation metrics, and baseline comparisons are also detailed.

By clearly defining the experimental setup, this chapter establishes the foundation for an objective comparison of algorithm performance in terms of energy efficiency and thermal comfort, under consistent and controlled conditions.

5.1 Algorithms hyperparameters

The performance of reinforcement learning algorithms is strongly influenced by the choice of hyperparameters. In this work, all hyperparameters for DDPG, TD3, and SAC were optimized using Optuna. Optimization was carried out separately for each training setting: online, offline, and offline-online.

In the following sections, we first describe how Optuna was used to tune the algorithms, and then report the final selected values for each configuration.

5.1.1 Hyperparameter optimization with Optuna

For each algorithm and training setting, a search space that includes relevant hyperparameters such as learning rates, discount factor (γ), soft update coefficient (τ),

batch size, and others depending on the specific algorithm (e.g., target smoothing parameters for TD3, entropy temperature learning rate for SAC).

Each trial consisted of training the agent for the duration of one simulated year, followed by an evaluation phase on a fixed 6-month test period. The objective function to maximize was defined as the negative sum of the average power consumption and the average temperature error over the test period. This formulation reflects the trade-off between energy efficiency and comfort that is central in HVAC control tasks.

Optimization was performed separately for online training, offline training, and offline-online training. In the offline-online training, the offline phase was initialized using the same hyperparameters previously selected during the stand-alone offline training.

Typically, 100 to 140 trials per algorithm were performed, with an early stop when convergence was detected. The best-performing set of parameters for each configuration is reported in the following sections.

5.1.2 Selected hyperparameters

Table 5.1: Optimized hyperparameters for DDPG

Parameter	**Online**	**Offline**	**Offline-Online**
Discount factor γ	0.95	0.88	0.95
Actor learning rate	0.0007	9.34e-05	0.0007
Critic learning rate	0.0002	8.58e-04	0.0002
Batch size	2048	256	2048
Soft target update coefficient τ	0.05	0.05	0.05
Buffer length	50000	–	50000
Update interval	8	–	4
Number of updates	2	–	1

Table 5.2: Optimized hyperparameters for TD3

Parameter	**Online**	**Offline**	**Offline-Online**
Discount factor γ	0.9	0.88	0.88
Actor learning rate	5.97e-04	3.90e-04	9.93e-04
Critic learning rate	1.30e-04	2.35e-04	2.67e-05
Batch size	2048	128	2048
Soft target update coefficient τ	0.05	0.001	0.02
Target smoothing σ	0.2	0.5	0.2
Target smoothing clip	0.5	0.3	0.5
Update actor interval	2	–	–
Buffer length	50000	–	50000
Update interval	9	–	2
Number of updates	1	–	1

Table 5.3: Optimized hyperparameters for SAC

Parameter	Online	Offline	Offline-Online
Discount factor γ	0.88	0.88	0.88
Actor learning rate	9.93e-04	1.33e-04	9.93e-04
Critic learning rate	2.67e-05	1.19e-05	2.67e-05
Temperature learning rate	2.02e-04	1.12e-05	2.02e-04
Batch size	2048	512	2048
Soft target update coefficient τ	0.02	0.05	0.02
Number of critics	4	–	4
Buffer length	50000	–	50000
Update interval	8	–	2
Number of updates	1	–	1

5.2 Training methods

5.2.1 Online training

To evaluate the performance of online reinforcement learning, each algorithm (DDPG, TD3 and SAC) was trained multiple times over progressively longer simulation periods, specifically 1, 2, 3, 6, and 12 months. For each duration, five agents were independently trained using different combinations of weather files.

This setup enables the analysis of the impact of training duration on both final performance and learning dynamics. The evaluation includes performance in a fixed test period after training, as well as the evolution of the reward function during training. This approach allows for an assessment of the convergence behavior and the identification of the point at which each algorithm begins to produce satisfactory results.

Furthermore, the feasibility of applying online training in real-world conditions was investigated by analyzing agent performance during the training process itself, without interrupting or resetting the simulation. This evaluation aims to determine whether acceptable performance can be achieved while learning is still ongoing, as would be required in a real deployment scenario.

5.2.2 Offline training

To evaluate the effectiveness of offline reinforcement learning, a comprehensive experimental setup was implemented. For each of the three algorithms, a total of 25 agents were trained on datasets generated by a PID controller. These datasets differ in size and weather conditions.

Specifically, five dataset sizes were considered: 1 month, 2 months, 3 months, 6 months, and 12 months. For each size, five distinct datasets were generated using different weather files, resulting in a diverse range of training experiences. Each dataset was used to train a separate agent, allowing the evaluation of both variability due to weather conditions and the influence of the size of the dataset on learning outcomes.

This experimental design allows for the investigation of two key aspects of offline learning: the effect of dataset size on the quality of the learned policy, and the robustness of each algorithm to variations in the training data distribution. By systematically increasing the amount of data, the minimum dataset size required to achieve satisfactory performance can be estimated.

All agents were evaluated within the same fixed test environment to ensure comparability. This setup enables a direct analysis of the impact of both algorithmic configurations and dataset characteristics on generalization and performance in previously unseen scenarios.

5.2.3 Offline with online fine-tuning

To evaluate the impact of combining offline pre-training with online fine-tuning, a hybrid training approach was applied. For all algorithms, the agent trained offline that performed the best was selected based on its performance in a fixed test period. The selected DDPG and TD3 agents had been trained offline using 1 month of data, while the SAC agent had been trained using 2 months of data.

These agents were then used as starting points for further online training over different durations: 1 week, 2 weeks, 1 month, 2 months, and 3 months. For each configuration, five agents were individually fine-tuned using distinct weather conditions, following the same simulation framework adopted in the pure online training experiments.

This setup enables an analysis of the effectiveness of leveraging prior offline knowledge in an online learning context. The results can be compared with those obtained from purely offline and purely online training to assess whether pre-training contributes to improved sample efficiency, faster convergence, or enhanced control performance.

5.3 Online training phase analysis

To gain a deeper understanding of how each agent behaves during training, a detailed step-by-step analysis of the best-performing agents was carried out for each algorithm. Specifically, for each of the algorithms tested, the following agents were selected:

- The best agent trained fully online for a period of 6 months.
- The best agent trained offline and then online, over a period of 1 month.

The analysis was performed by examining every training step from the respective training periods. Each step was classified into one of four possible categories based on two key criteria: thermal comfort and energy consumption. These criteria were used to construct a two-dimensional classification table with two binary axes.

For thermal comfort, the classification was based on the absolute temperature error (ΔT) between room temperature and the setpoint:

- Good control: $\Delta T \leq 0.5\,\mathrm{K}$
- Bad control: $\Delta T > 0.5\,\mathrm{K}$

For energy efficiency, the agent's energy usage at each step was compared to that of a reference PID controller:

- Saving: the agent used less energy than the PID
- Waste: the agent used more energy than the PID

This allowed for a direct comparison between agents trained online from scratch and those initialized with offline policies, in order to determine whether real-time fine-tuning is feasible without causing critical performance degradation during the learning process.

In addition to this per-step classification, a second analysis was carried out to assess the total energy difference between the agent and the PID over time. Specifically, for each time step, the difference in energy consumption (agent versus PID) was recorded and aggregated to calculate the cumulative energy gain or loss throughout the training period.

For both analyses, results are grouped by time intervals to highlight the evolution of agent behavior, monthly results are reported for agents trained online for 6 months, and weekly results are reported for agents trained offline with online optimization over 1 month.

This dual perspective enables a more complete interpretation of how agents improve over time, both in terms of comfort-quality trade-offs and overall energy performance.

Chapter 6

Experimental Results

This chapter presents the results obtained from the evaluation of reinforcement learning agents trained in different strategies. Each configuration is assessed in terms of its ability to control the HVAC system effectively and efficiently.

The evaluation test is conducted over a six-month heating season, using the Aosta 2019 weather file, which was not included in the training data. This setup allows for a reliable assessment of the agents' generalization capabilities under realistic and previously unseen conditions.

The key performance indicators (KPIs) considered to evaluate the quality of a test are:

- The mean absolute error between indoor temperature and the target setpoint, representing thermal comfort and control accuracy.
- The average power consumption of the heat pump during the evaluation period, representing the energy efficiency.

All agents were tested under identical environmental and operational conditions, allowing for a consistent and fair comparison between learning strategies. The results are presented and discussed in the following sections.

An additional section follows the evaluation of the trained and tested agents, focusing on the analysis of online training to assess its deployability in a real-world environment.

6.1 Online

The performance of agents trained exclusively in an online model is illustrated through the boxplots shown in Fig. 6.1 (average temperature error) and in Fig. 6.2

(average power consumption), while the numerical results of the tests are listed in Tab. 6.1.

From the results, it is evident that DDPG and TD3 perform consistently well over different training durations, particularly in terms of maintaining low temperature errors. For example, after 1 month of training, both algorithms are already able to keep the average temperature deviation below 1K, with TD3 performing slightly better in terms of stability.

In contrast, SAC appears to struggle during the early stages of training. For 1 and 2 month durations, SAC shows significantly higher temperature deviations (up to 6.3K) and elevated energy consumption. However, its performance improves substantially with longer training durations, converging toward the levels of DDPG and TD3 after 6 and 12 months. This suggests that SAC may require more extensive interaction with the environment to achieve stable and efficient behavior.

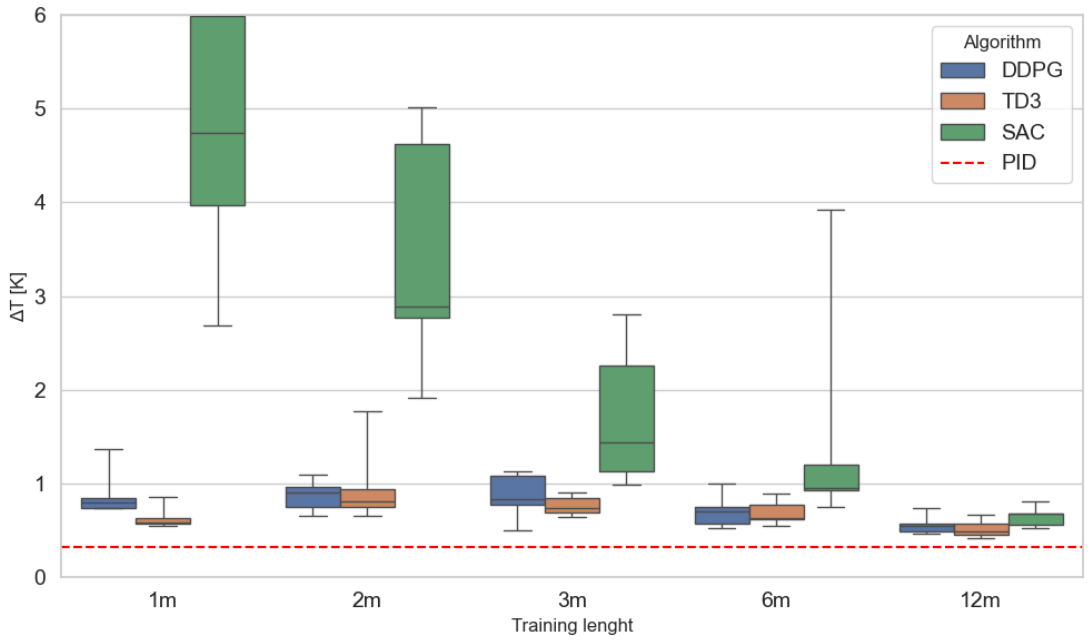

Figure 6.1: Boxplot of the average temperature error (ΔT) of agents trained online for different training durations (1, 2, 3, 6, and 12 months). Each box summarizes five independent runs using 2019 weather data.

A notable phenomenon is the performance drop observed for DDPG and TD3 between the 2-month and 3-month training durations. Both algorithms exhibit an increase in the temperature error and variability in energy consumption. This temporary degradation could be due to instabilities in the policy updates caused by over-exploration or learning dynamics that fail to adapt efficiently to changing

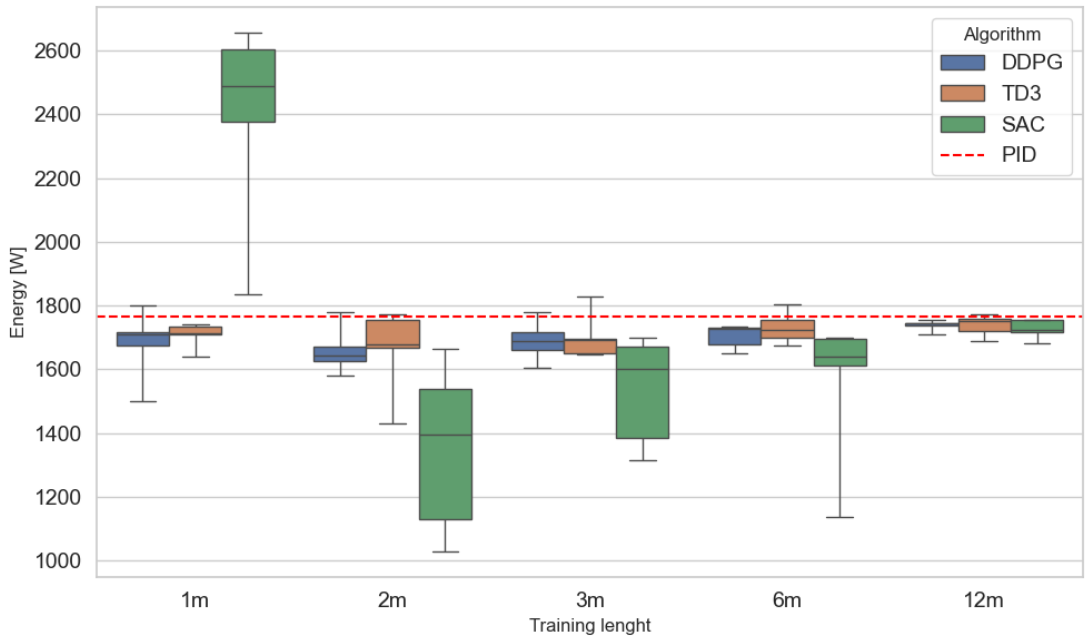

Figure 6.2: Boxplot of the average power consumption (P) of agents trained online for different training durations. Each box summarizes five independent runs using 2019 weather data.

seasonal conditions. It is also possible that the replay buffer retained outdated experiences that temporarily disrupted learning. However, after this phase, both agents show signs of recovery and convergence: by 6 and 12 months, DDPG and TD3 once again achieve strong control performance with minimal energy usage and high temperature accuracy.

The convergence of the reward function reflects the evolution of the agents during the training phase. In particular, the quick convergence of DDPG and TD3 to an optimal reward value after just 14 days is shown in Fig. 6.3 and Fig. 6.5, and the slow and constant improvement of SAC is represented in Fig. 6.7. Looking at the graphs of DDPG and TD3 zoomed in Fig. 6.4 and Fig. 6.6 it's evident the drop of performance after 60 days. Although the difference in the graphs may appear small, an increase of more than 100 points in the daily cumulative penalty translates to a per-step penalty increase of approximately 0.35 points, which is approximately 20% higher and results in a significant performance degradation.

These observations highlight the robustness of DDPG and TD3 in online learning, although with some sensitivity to the duration and timing of training. SAC, while initially less efficient, proves to be a competitive alternative given sufficient time to learn. All algorithms yield a savings in terms of energy consumption, but with a

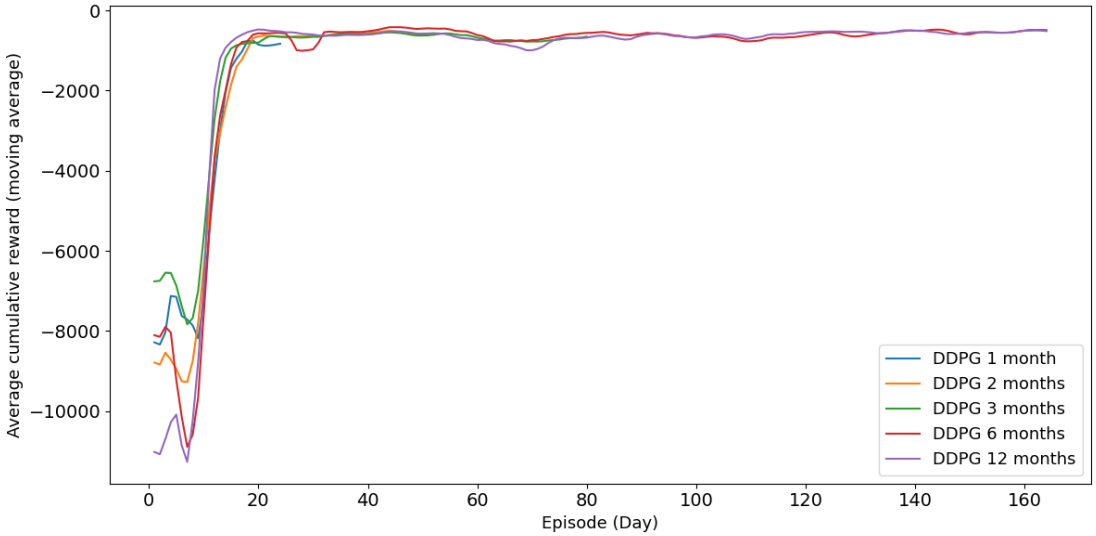

Figure 6.3: DDPG average reward for different training lengths. Each line plots the average cumulative reward per day, calculated across five independently trained agents for each respective training length.

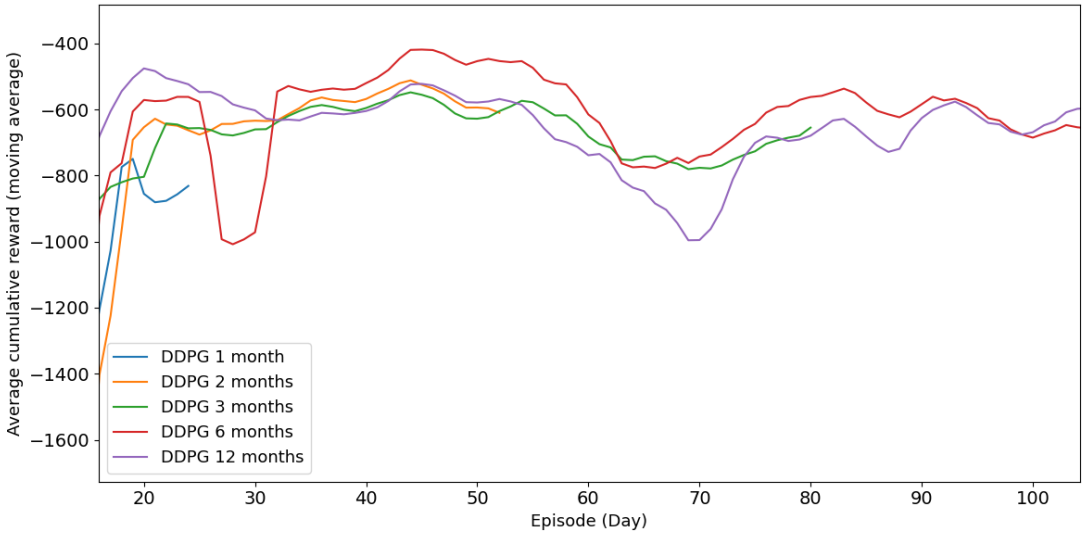

Figure 6.4: Zoom of DDPG reward function

slight loss of thermal comfort with respect to the PID controller.

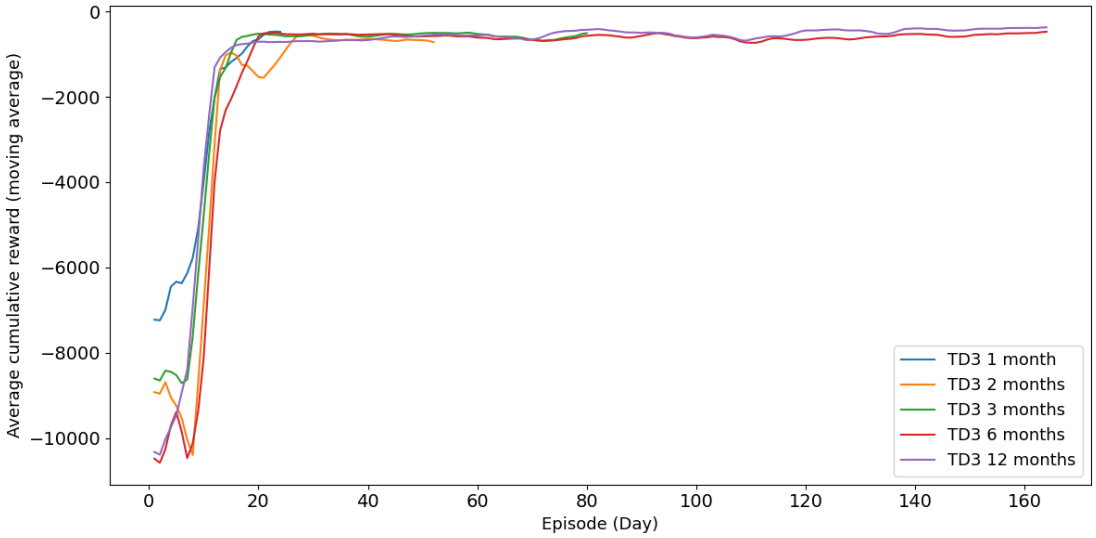

Figure 6.5: TD3 average reward for different training lengths. Each line plots the average cumulative reward per day, calculated across five independently trained agents for each respective training length.

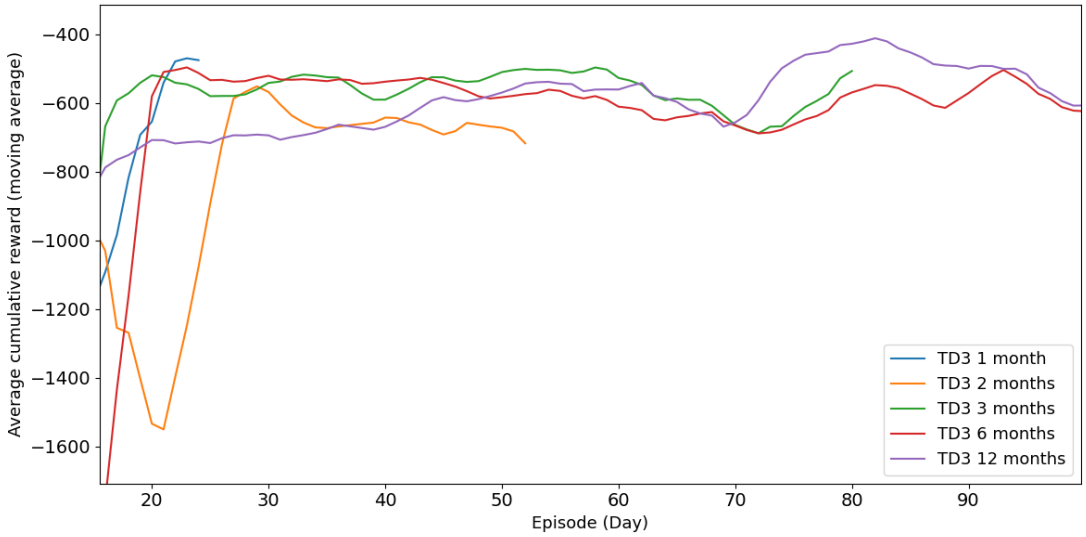

Figure 6.6: Zoom of TD3 reward function

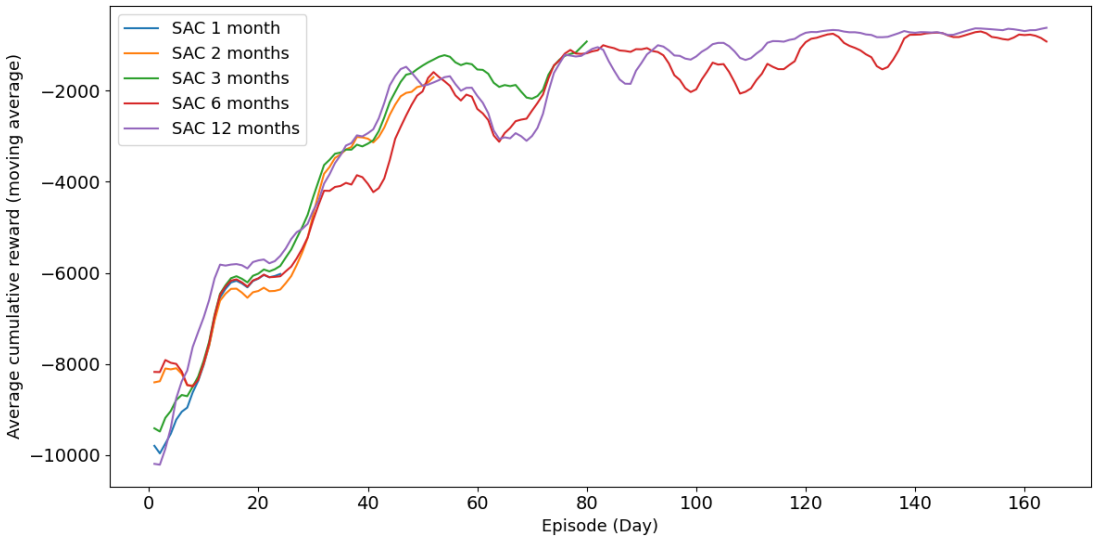

Figure 6.7: SAC average reward for different training lengths. Each line plots the average cumulative reward per day, calculated across five independently trained agents for each respective training length.

Table 6.1: Average temperature error (ΔT) and power consumption (P) of online-trained agents over 5 evaluation runs, including energy saving compared to a PID baseline (1766.284 W).

Length	**Algorithm**	**Mean ΔT (K)**	**Std ΔT**	**Mean P (W)**	**Std P**	**Saving (%)**
1m	DDPG	0.90	0.25	1,680.23	119.06	4.87
1m	TD3	0.64	0.12	1,719.28	42.51	2.66
1m	SAC	4.52	1.32	2,390.04	304.84	−35.31
2m	DDPG	0.87	0.15	1,660.56	85.44	5.98
2m	TD3	0.99	0.44	1,660.07	155.50	6.01
2m	SAC	3.04	1.43	1,351.94	257.98	23.46
3m	DDPG	0.85	0.22	1,689.31	67.03	4.36
3m	TD3	0.76	0.09	1,702.29	76.75	3.62
3m	SAC	1.73	0.62	1,534.57	145.07	13.12
6m	DDPG	0.71	0.18	1,711.49	30.02	3.10
6m	TD3	0.69	0.12	1,729.04	55.05	2.11
6m	SAC	1.75	1.33	1,558.43	243.05	11.77
12m	DDPG	0.56	0.10	1,737.75	22.71	1.61
12m	TD3	0.52	0.10	1,738.80	33.88	1.56
12m	SAC	0.66	0.12	1,725.92	33.77	2.28

6.2 Offline

The performance of agents trained offline is illustrated through the boxplots shown in Fig. 6.8 (average temperature error) and Fig. 6.9 (average power consumption). Each box summarizes five evaluation runs using unseen 2019 weather data.

From the results, SAC consistently outperforms both DDPG and TD3 in terms of control precision and robustness over longer training durations. Even with just 2 months of data, SAC is able to maintain a low temperature error, typically around 1K, with limited variability. As the amount of training data increases, SAC continues to deliver stable performance with minimal degradation, suggesting a strong ability to generalize from offline experiences.

Interestingly, TD3 achieves remarkably good performance even with very limited offline data. After just 1 month of training, TD3 is able to maintain a temperature deviation below 1K, with low variance and moderate energy usage. This makes it a particularly compelling choice in scenarios where collecting extensive datasets is not feasible. However, its performance becomes less stable as the dataset grows: after 2 and 3 months, TD3 exhibits noticeable outliers and increased variability in both temperature error and power consumption, possibly due to sensitivity to dataset composition or the accumulation of suboptimal experiences.

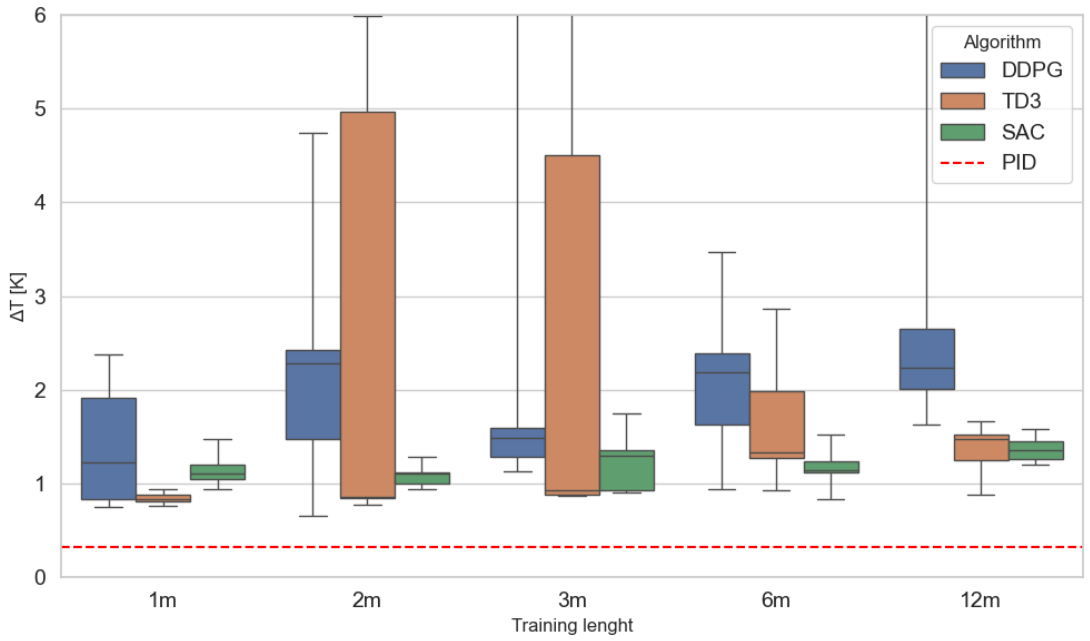

Figure 6.8: Boxplot of the average temperature error (ΔT) of agents trained offline over different dataset lengths (1, 2, 3, 6, and 12 months). Each box summarizes five independent runs using 2019 weather data.

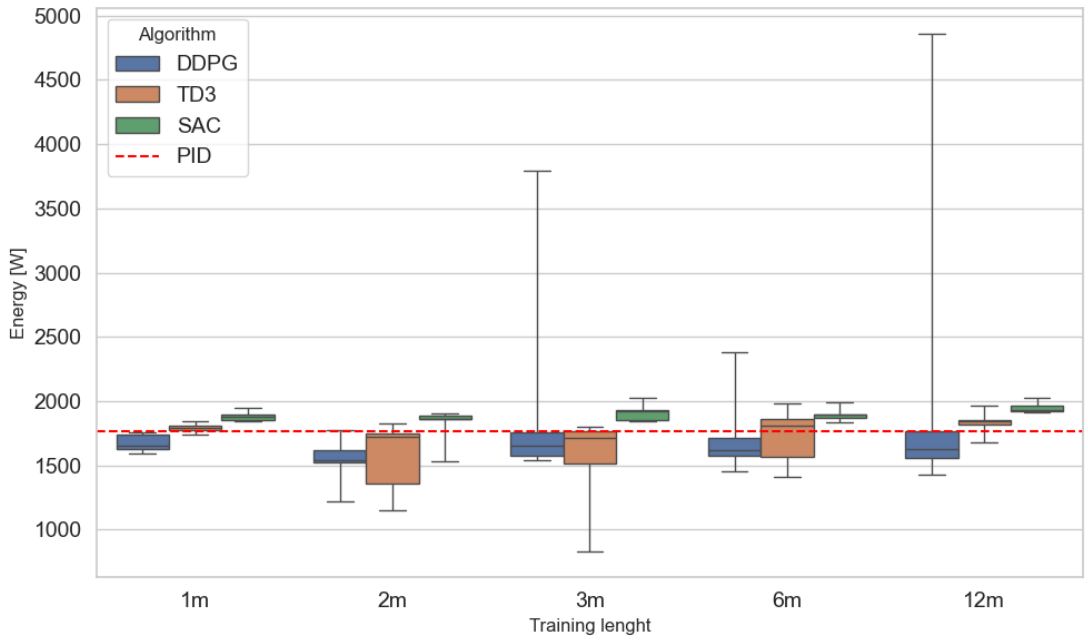

Figure 6.9: Boxplot of the average power consumption (P) of agents trained offline over different dataset lengths. Results are based on five simulations per configuration using unseen weather data.

DDPG, on the other hand, shows the most erratic behavior across all durations. It suffers from severe performance drops at 3 and 12 months, with extreme temperature errors (up to 13K) and energy peaks exceeding 4800W. Although it occasionally performs well, its results are inconsistent and generally less reliable than those of TD3 and SAC.

Despite these issues, both DDPG and TD3 tend to recover partially with longer training durations. After 6 months, both algorithms demonstrate improved control performance and reduced variance, although SAC remains the most stable overall. This pattern suggests that while all three algorithms are capable of learning useful control policies offline, SAC provides the most robust and data-efficient solution, while TD3 stands out for its effectiveness with minimal data, although at the cost of lower long-term stability.

With the only use of offline training, all three algorithms were unable to achieve good performance compared to the PID benchmark or to the online trained agents.

Table 6.2: Average temperature error (ΔT) and power consumption (P) of offline-trained agents over 5 evaluation runs, including energy saving compared to a PID baseline (1766.284 W).

Length	Algorithm	Mean ΔT (K)	Std ΔT	Mean P (W)	Std P	Saving (%)
1m	DDPG	1.42	0.61	1,675.70	65.59	5.13
1m	TD3	0.85	0.07	1,790.09	39.09	−1.35
1m	SAC	1.15	0.19	1,884.22	36.30	−6.68
2m	DDPG	2.32	1.51	1,534.08	202.23	13.15
2m	TD3	2.49	2.10	1,559.30	289.99	11.72
2m	SAC	1.09	0.13	1,809.53	152.79	−2.45
3m	DDPG	2.41	2.99	2,063.83	878.88	−16.85
3m	TD3	3.02	2.88	1,525.10	382.66	13.65
3m	SAC	1.25	0.34	1,915.87	65.42	−8.47
6m	DDPG	2.12	1.00	1,745.73	328.33	1.16
6m	TD3	1.68	0.75	1,724.61	218.92	2.36
6m	SAC	1.17	0.26	1,897.16	52.78	−7.41
12m	DDPG	4.37	4.56	2,247.21	1,346.14	−27.23
12m	TD3	1.36	0.33	1,831.36	101.70	−3.68
12m	SAC	1.37	0.13	1,948.80	42.07	−10.33

6.3 Offline with online fine-tuning

The performance of agents trained offline and fine-tuned online is illustrated through the boxplots shown in Fig. 6.10 (average temperature error) and in Fig. 6.11 (average power consumption) while the numerical results of the tests are listed in Tab. 6.3.

As for the agents trained exclusively online, DDPG and TD3 maintain solid performance throughout all training durations. For example, after just one week of training, both algorithms already achieve average temperature deviations comparable to the agents trained for six months online results, with TD3 generally showing slightly better stability.

A notable observation is the variability in power consumption for SAC during the earliest phase (1 week), where the energy use is erratic and occasionally significantly lower or higher than for the other algorithms, accompanied by higher temperature errors. This suggests that SAC experiences instability in its early learning stage and possibly struggles with the offline-to-online transition. However, unlike the online-only training case, SAC shows a more immediate improvement as the training duration increases, with temperature errors quickly dropping below 1K and energy

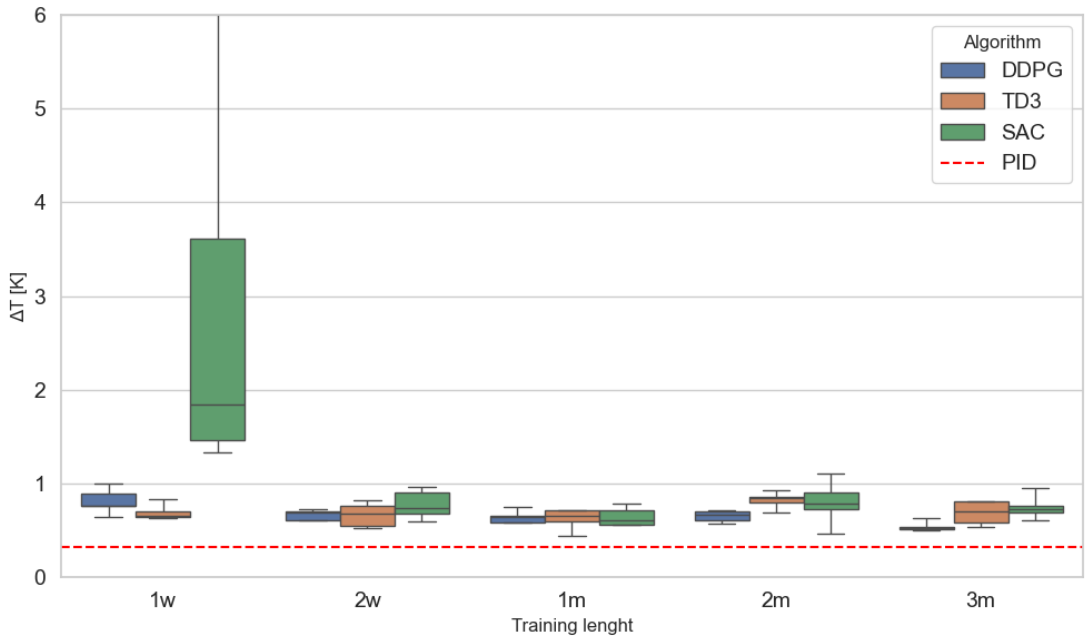

Figure 6.10: Boxplot of the average temperature error (ΔT) of agents trained offline and optimized online for different training durations (1 week, 2 weeks, 1, 2, and 3 months). Each box summarizes five independent runs using 2019 weather data

consumption becoming more consistent by 2 weeks and beyond.

Overall, the results of the offline-to-online suggest that DDPG and TD3 agents are capable of quickly adapting and maintaining performance across all durations tested, while SAC benefits from longer training, but demonstrates faster convergence and stabilization compared to purely online training. This confirms that offline pretraining followed by online fine-tuning effectively reduces the initial instability of SAC and improves its data efficiency.

As demonstrated in the reward function plots shown in Fig. 6.12, Fig. 6.13 and Fig. 6.14, the efficacy of offline pre-training is evident. In all three figures, the initial lower peak is significantly superior to the corresponding peak in the fully online case, with the exception of the SAC, which exhibits an initial unstable phase, as previously mentioned.

All algorithms trained offline and then fine-tuned online achieved energy savings compared to the PID controller, with a moderate trade-off in thermal comfort. Similarly to the fully online case, this compromise is gradually reduced over time, but thanks to the initial offline training, satisfactory performance is reached in a shorter training duration.

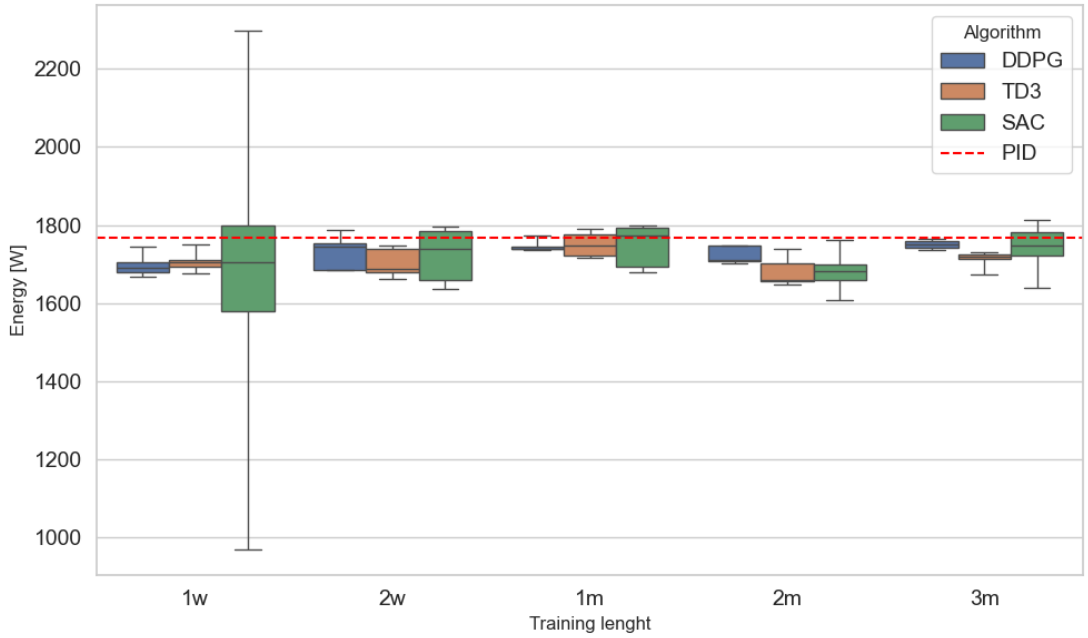

Figure 6.11: Boxplot of the average power consumption (P) of agents trained offline and optimized online for different training durations. Each box summarizes five independent runs using 2019 weather data

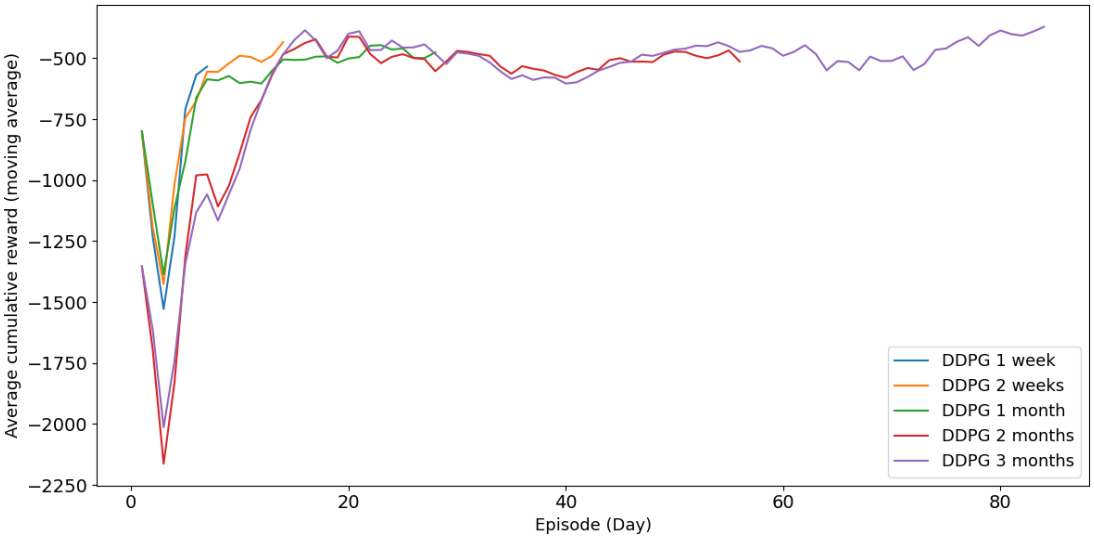

Figure 6.12: DDPG average reward for different training lengths of the offline trained and than optimized online agents. Each line plots the average cumulative reward per day, calculated across five independently trained agents for each respective training length

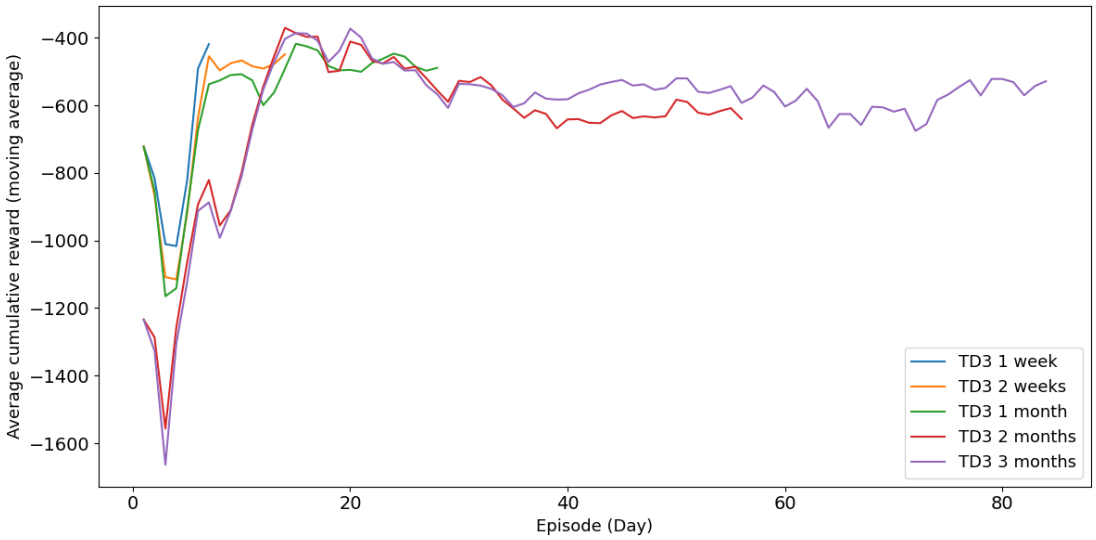

Figure 6.13: TD3 average reward for different training lengths of the offline trained and than optimized online agents. Each line plots the average cumulative reward per day, calculated across five independently trained agents for each respective training length

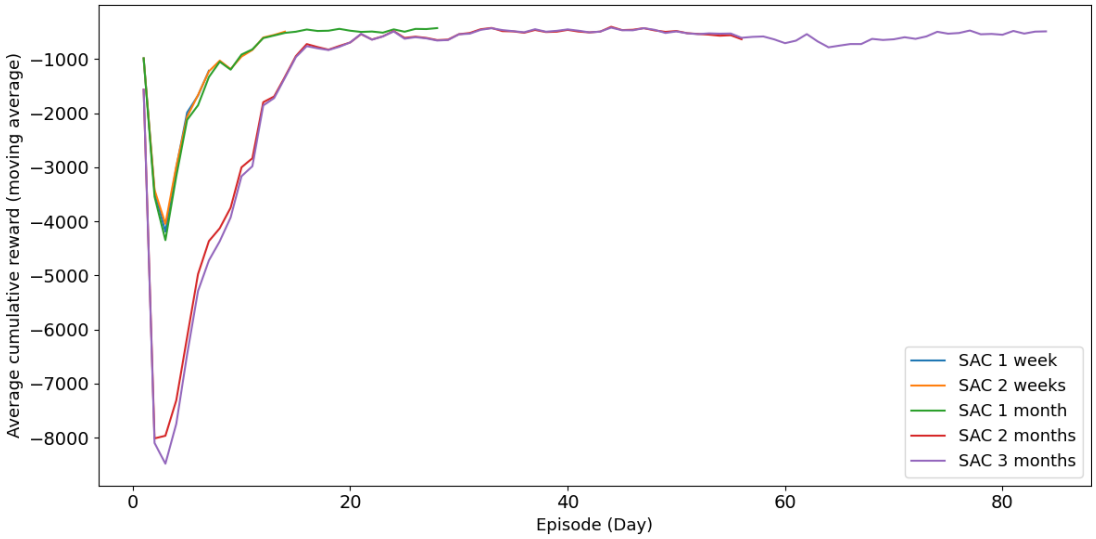

Figure 6.14: SAC average reward for different training lengths of the offline trained and than optimized online agents. Each line plots the average cumulative reward per day, calculated across five independently trained agents for each respective training length

Table 6.3: Average temperature error (ΔT) and power consumption (P) of offline-to-online trained agents over 5 evaluation runs, including energy saving compared to a PID baseline (1766.284 W).

Length	**Algorithm**	**Mean ΔT (K)**	**Std ΔT**	**Mean P (W)**	**Std P**	**Saving (%)**
1w	DDPG	0.812	0.119	1,697.83	28.16	3.88
1w	TD3	0.693	0.074	1,706.02	27.20	3.41
1w	SAC	3.243	2.172	1,670.87	466.62	5.40
2w	DDPG	0.668	0.053	1,730.36	40.90	2.03
2w	TD3	0.668	0.116	1,703.85	34.00	3.53
2w	SAC	0.778	0.145	1,722.32	70.36	2.49
1m	DDPG	0.640	0.070	1,746.92	15.15	1.10
1m	TD3	0.626	0.099	1,750.21	29.27	0.91
1m	SAC	0.645	0.091	1,747.67	56.46	1.05
2m	DDPG	0.653	0.057	1,723.79	20.93	2.41
2m	TD3	0.823	0.088	1,680.56	39.84	4.85
2m	SAC	0.797	0.219	1,682.81	54.08	4.73
3m	DDPG	0.541	0.050	1,750.00	10.66	0.92
3m	TD3	0.685	0.126	1,711.08	20.66	3.13
3m	SAC	0.748	0.132	1,739.91	64.67	1.49

6.4 Online training phase analysis

For this analysis, it is important to remember that the percentage of steps with "bad control" between 12% and 15% is partly structural. They disproportionately occur around setpoint transitions, where a delay in temperature response is expected. This delay also affects PID, which routinely accumulates a 12–15% bad-control rate for the same reason. Therefore, we adopt the thresholds that bad control below 15% and good control with saving above 50% indicate a practically satisfactory policy balancing comfort and energy.

The results of the online-only training, shown in Fig. 6.15 and Fig. 6.16, reveal a clear learning curve across all algorithms. In the first month, performance is markedly poor: thermal control is imprecise, energy usage is high, and most time steps fall into inefficient quadrants, often accompanied by significant energy losses compared to baseline PID.

From the second month onward a divergence emerges, the DDPG and TD3 agents begin to demonstrate more than 50% of their time steps in the "good control with energy savings" quadrant, indicating more effective temperature regulation and tangible energy savings relative to PID.

Analyzing the cumulative energy difference with respect to PID: all agents start with negative energy savings (i.e., higher energy consumption). However, by month 2 for DDPG and TD3 and month 3 for SAC they start to exhibit positive net savings.

In considering the practical implementation of such a controller, it is evident that the training phase would require a period of at least two months, during which a significant amount of energy and thermal discomfort would be incurred. This observation renders the notion of a straightforward plug-and-play application of an online algorithm rather implausible.

The findings represented in Fig. 6.17 and Fig. 6.18 demonstrate that the results for agents that have been trained offline with online fine-tuning differ from those observed for agents that have been trained exclusively online. Following a preliminary period in which effective regulation proved difficult and resulted in considerable thermal discomfort, the different algorithms demonstrated consistently superior performance throughout the subsequent week, with a steady enhancement continuing until the final week. Following completion of the four-week online training period, it was observed that DDPG and TD3 exhibited a mere 14% occurrence of suboptimal controls. This is analogous to the PID, and is attributed to the temporal requirement to attain the new setpoint subsequent to a setpoint alteration.

A notable observation is that, in the second week, both TD3 and DDPG achieve significant energy savings, approximately 125 kW and 198 kW respectively, despite still issuing a considerable number of suboptimal control actions. However, this

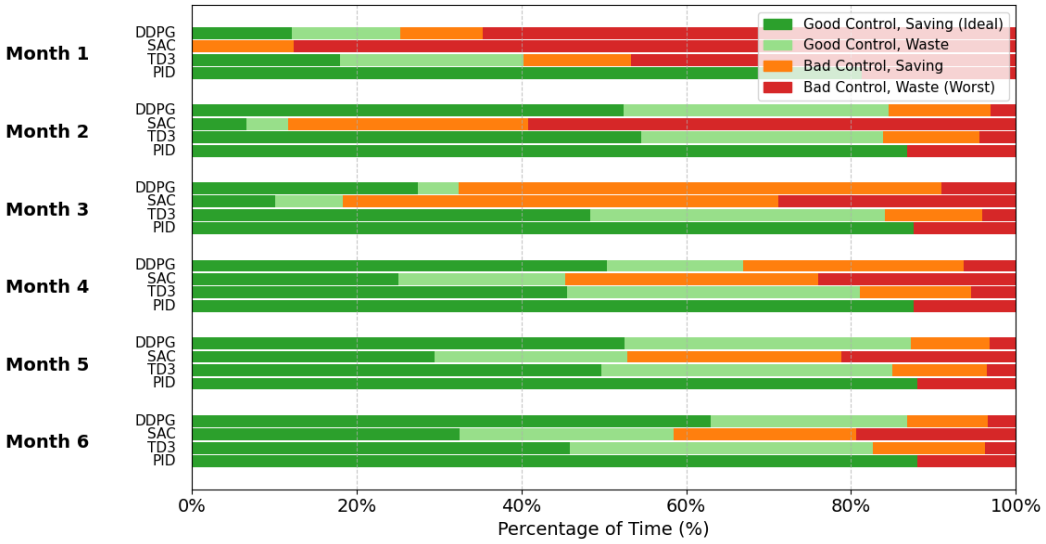

Figure 6.15: Energy savings and control quality by month for the agents trained online

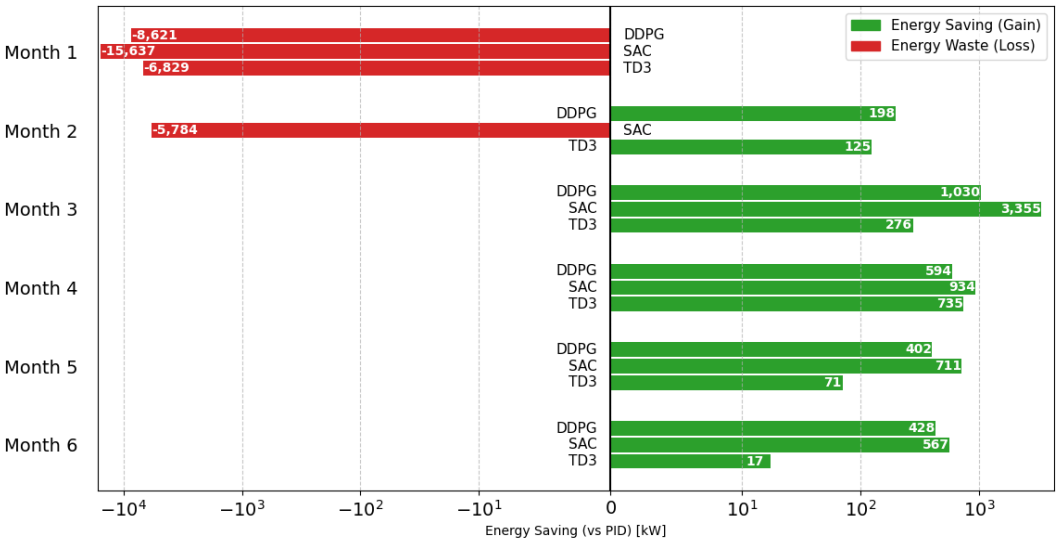

Figure 6.16: Monthly energy savings during online training

may not pose a practical concern in real-world deployments: in systems where the RL controller is introduced during the final stages of construction, when the building is still unoccupied, a three-week setup period to reach optimal performance could be commercially acceptable.

This analysis, in general, corroborates the significance of offline pre-training in

anticipation of real-life deployment.

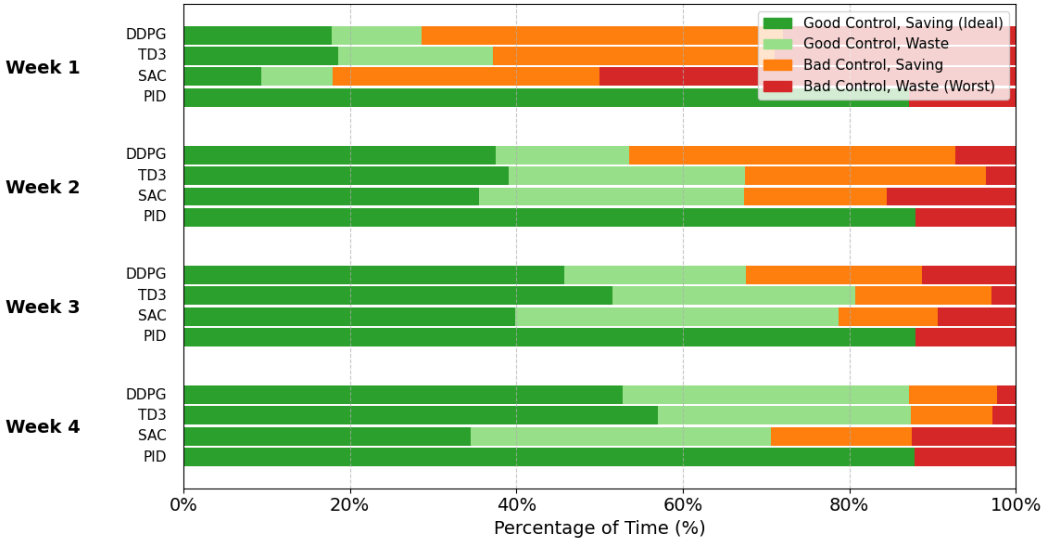

Figure 6.17: Energy savings and control quality by week for the agents pre-trained offline

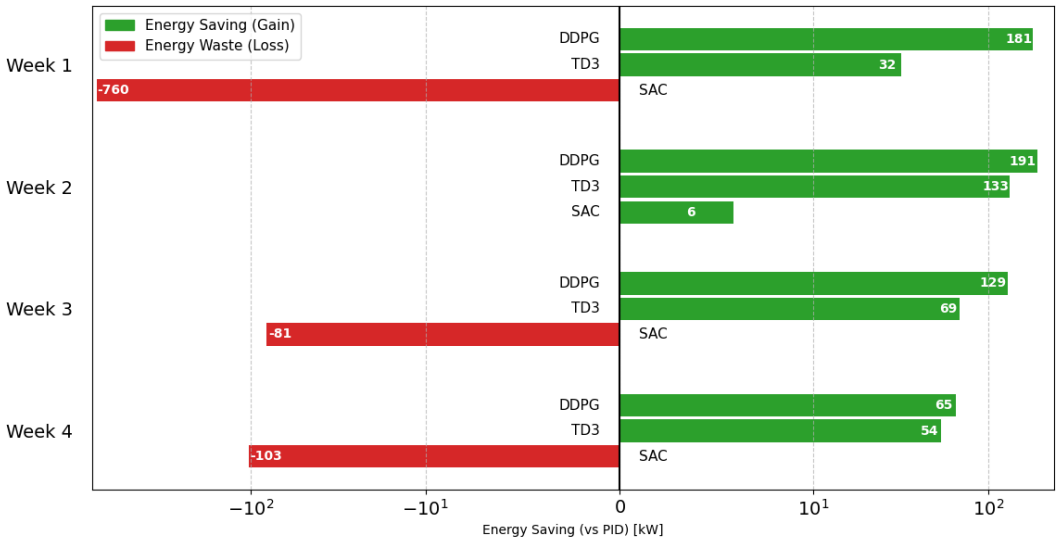

Figure 6.18: Weekly energy savings during online fine-tuning

Chapter 7

Conclusions

The present chapter thus brings the study to a conclusion by offering a synthesis of the key findings and a response to the initial research questions. Furthermore, the concluding section of this study proposes several prospective future investigations and enhancements.

7.1 Final conlusions

The present study has evaluated the potential of reinforcement learning technology in the context of a HVAC controller, highlighting its feasibility with such a specific application. The challenge that must be overcome to facilitate the effective implementation of a smart controller based on an RL algorithm in real-world scenarios is the significant potential for substantial energy wastage and the resulting uninhabitable state of the building during the training phase of the agent.

The results of the evaluation test demonstrate that certain RL algorithms, including DDPG and TD3, demonstrate enhanced proficiency in on-sight training, resulting in a two-month training period following which the control group begins to comprehend the dynamics of the environment. However, these results do not align with the characteristics of a commercially available product. A product that consumes energy to this extent over a period of two months is likely to have a negative impact on the sales of the smart control.

The study demonstrated that training agents exclusively on historical data yielded constrained outcomes. The agents were unable to enhance their policy towards optimal behavior. Nevertheless, for algorithms such as TD3 and SAC, a dataset of only one month and two months, respectively, is sufficient to achieve the optimal result that an offline trained agent can attain in this environment. This is advantageous in cases where data mining is challenging.

In general, DDPG and TD3 have been shown to be more conducive to online

training, while the SAC algorithm performs better when training is carried out offline.

As anticipated, optimal outcomes are achieved through the integration of offline and online training methodologies. The two algorithms that are more suitable for online training have been demonstrated to achieve favorable results after a minimal training phase, with DDPG and TD3 requiring only two weeks. Simultaneously, the SAC has corroborated the challenges it faces in terms of its capacity to direct learning from the environment, which it perceives to be arduous and time-consuming. Unlike their counterparts, which were exclusively trained online, both DDPG and TD3 demonstrated a propensity to avoid energy expenditure throughout the designated two-week training period. In the context of a commercial environment, this capacity to conserve energy could be a significant advantage. It is important to note that while agents may not exhibit optimal control during the training phase, they will not incur financial losses compared to a PID control. As was stated during the presentation of the results, such a training phase could be implemented during the construction phase of the building, at a time when it is still unoccupied, so no one experiences thermal discomfort.

Overall, RL agents have the ability to obtain a percentage reduction, between 2% and 5%, of energy consumption compared to the PID controller. This may lead to the perception that the outcomes don't worth the effort. However, it is imperative to take into account the fact that this simulated environment provides a highly elementary control system, with the PID controller already functioning in an optimal way. The primary objective in the context of a real-world application is to identify the most suitable application of RL technology, for example, a rule based controller could find more difficulties controlling a more complex system where is not known which control is optimal.

It has been demonstrated that RL control based on DDPG and TD3 trained both offline and online has exhibited the potential of AI in the HVAC context, and their capacity to learn optimal control strategies in short periods of time.

7.2 Future improvements

Although the results presented in this work are promising, there is still space for further improvements and extensions. Several directions could be explored to enhance the robustness and applicability of the study:

- Comparing additional algorithms with those presented in this work.
- Using real-world data to build the historical dataset.
- Including the cooling season in the training and testing phases.

- Developing more complex controllers, possibly by adding an additional control variable and extending the action space.

Bibliography

[1] L.T. Graham, T. Parkinson, and S. Schiavon. «Lessons learned from 20 years of CBE's occupant surveys». In: *Buildings and Cities* (2021) (cit. on p. 1).

[2] Luis Pérez-Lombard, José Ortiz, and Christine Pout. «A review on buildings energy consumption information». In: *Energy and Buildings* 40.3 (2008), pp. 394–398. ISSN: 0378-7788 (cit. on p. 1).

[3] Radu Godina, Eduardo M. G. Rodrigues, Edris Pouresmaeil, João C. O. Matias, and João P. S. Catalão. «Model Predictive Control Home Energy Management and Optimization Strategy with Demand Response». In: *Applied Sciences* 8.3 (2018) (cit. on p. 1).

[4] David Silver, Aja Huang, Chris J. Maddison, et al. «Mastering the game of Go with deep neural networks and tree search». In: *Nature* 529 (2016), pp. 484–489. ISSN: 1476-4687 (cit. on p. 2).

[5] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. «End-to-End Training of Deep Visuomotor Policies». In: *CoRR* abs/1504.00702 (2015). arXiv: 1504.00702. URL: http://arxiv.org/abs/1504.00702 (cit. on p. 2).

[6] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. *Learning to Drive in a Day*. 2018. arXiv: 1807.00412 [cs.LG]. URL: https://arxiv.org/abs/1807.00412 (cit. on p. 2).

[7] Edward L. Thorndike. «The Law of Effect». In: *The American Journal of Psychology* 39.1/4 (1927), pp. 212–222. ISSN: 00029556. URL: http://www.jstor.org/stable/1415413 (visited on 10/01/2024) (cit. on p. 4).

[8] Ajith Abraham. «Artificial neural networks». In: *Handbook of measuring system design* (2005) (cit. on p. 8).

[9] Timothy Paul Lillicrap, Jonathan James Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daniel Pieter Wierstra. *Continuous control with deep reinforcement learning*. US Patent 10,776,692. Sept. 2020 (cit. on p. 9).

[10] Scott Fujimoto, Herke van Hoof, and David Meger. *Addressing Function Approximation Error in Actor-Critic Methods*. 2018. arXiv: 1802.09477 [cs.AI]. URL: https://arxiv.org/abs/1802.09477 (cit. on p. 11).

[11] Tuomas Haarnoja et al. *Soft Actor-Critic Algorithms and Applications*. 2019. arXiv: 1812.05905 [cs.LG]. URL: https://arxiv.org/abs/1812.05905 (cit. on p. 14).

[12] Takuma Seno and Michita Imai. *d3rlpy: An Offline Deep Reinforcement Learning Library*. 2022. arXiv: 2111.03788 [cs.LG]. URL: https://arxiv.org/abs/2111.03788 (cit. on p. 16).

[13] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. «Stable-Baselines3: Reliable Reinforcement Learning Implementations». In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html (cit. on p. 16).

[14] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. *OpenAI Gym*. 2016. arXiv: 1606.01540 [cs.LG]. URL: https://arxiv.org/abs/1606.01540 (cit. on p. 16).

[15] Paul Scharnhorst et al. «Energym: A Building Model Library for Controller Benchmarking». In: *Applied Sciences* 11.8 (2021). ISSN: 2076-3417. DOI: 10.3390/app11083518. URL: https://www.mdpi.com/2076-3417/11/8/3518 (cit. on p. 17).

[16] Amirreza Heidari, Francois Marechal, and Dolaana Khovalyg. «An adaptive control framework based on Reinforcement learning to balance energy, comfort and hygiene in heat pump water heating systems». In: *Journal of Physics: Conference Series* 2042.1 (Nov. 2021), p. 012006. DOI: 10.1088/1742-6596/2042/1/012006. URL: https://dx.doi.org/10.1088/1742-6596/2042/1/012006 (cit. on pp. 18, 22).

[17] Tianshu Wei, Yanzhi Wang, and Qi Zhu. «Deep reinforcement learning for building HVAC control». In: *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 2017, pp. 1–6. DOI: 10.1145/3061639.3062224 (cit. on pp. 18, 22).

[18] Yuiko Sakuma and Hiroaki Nishi. «Airflow Direction Control of Air Conditioners Using Deep Reinforcement Learning». In: *2020 SICE International Symposium on Control Systems (SICE ISCS)*. 2020, pp. 61–68. DOI: 10.23919/SICEISCS48470.2020.9083565 (cit. on pp. 18, 22).

[19] Zhiang Zhang and Khee Lam. «Practical Implementation and Evaluation of Deep Reinforcement Learning Control for a Radiant Heating System». In: *2018 ACM BuildSys*. Nov. 2018. DOI: 10.1145/3276774.3276775 (cit. on pp. 18, 22).

[20] Clement Lork, Wen-Tai Li, Yan Qin, Yuren Zhou, Chau Yuen, Wayes Tushar, and Tapan Saha. «An uncertainty-aware deep reinforcement learning framework for residential air conditioning energy management». In: *Applied Energy* 276 (July 2020). DOI: 10.1016/j.apenergy.2020.115426 (cit. on pp. 19, 22).

[21] Evan McKee, Yan Du, Fangxing Li, Jeffrey Munk, Travis Johnston, Kuldeep Kurte, Olivera Kotevska, Kadir Amasyali, and Helia Zandi. «Deep Reinforcement Learning for Residential HVAC Control with Consideration of Human Occupancy». In: *2020 IEEE Power & Energy Society General Meeting (PESGM)*. 2020, pp. 1–5. DOI: 10.1109/PESGM41954.2020.9281893 (cit. on pp. 19, 22).

[22] Anders Overgaard, Brian Kongsgaard Nielsen, Carsten Skovmose Kallesøe, and Jan Dimon Bendtsen. «Reinforcement Learning for Mixing Loop Control with Flow Variable Eligibility Trace». In: *2019 IEEE Conference on Control Technology and Applications (CCTA)*. 2019, pp. 1043–1048. DOI: 10.1109/CCTA.2019.8920398 (cit. on pp. 19, 22).

[23] Francesco M. Solinas, Andrea Bellagarda, Enrico Macii, Edoardo Patti, and Lorenzo Bottaccioli. «An Hybrid Model-Free Reinforcement Learning Approach for HVAC Control». In: *2021 IEEE International Conference on Environment and Electrical Engineering and 2021 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*. 2021, pp. 1–6. DOI: 10.1109/EEEIC/ICPSEurope51590.2021.9584805 (cit. on pp. 19, 22).

[24] Kuan-Heng Yu, Yi-An Chen, Emanuel Jaimes, Wu-Chieh Wu, Kuo-Kai Liao, Jen-Chung Liao, Kuang-Chin Lu, Wen-Jenn Sheu, and Chi-Chuan Wang. «Optimization of thermal comfort, indoor quality, and energy-saving in campus classroom through deep Q learning». In: *Case Studies in Thermal Engineering* 24 (Jan. 2021), p. 100842. DOI: 10.1016/j.csite.2021.100842 (cit. on pp. 19, 22).

[25] Hussain Kazmi, Fahad Mehmood, Stefan Lodeweyckx, and Johan Driesen. «Gigawatt-hour Scale Savings on a Budget of Zero: Deep Reinforcement Learning based Optimal Control of Hot Water Systems». In: *Energy* 144 (Dec. 2017). DOI: 10.1016/j.energy.2017.12.019 (cit. on pp. 19, 22).

[26] Thijs Peirelinck, Chris Hermans, Fred Spiessens, and Geert Deconinck. «Domain Randomization for Demand Response of an Electric Water Heater». In: *IEEE Transactions on Smart Grid* 12.2 (2021), pp. 1370–1379. DOI: 10.1109/TSG.2020.3024656 (cit. on pp. 19, 22).

[27] Elena Mocanu, Decebal Constantin Mocanu, Phuong H. Nguyen, Antonio Liotta, Michael E. Webber, Madeleine Gibescu, and J. G. Slootweg. «On-Line Building Energy Optimization Using Deep Reinforcement Learning». In: *IEEE Transactions on Smart Grid* 10.4 (2019), pp. 3698–3708. DOI: 10.1109/TSG.2018.2834219 (cit. on pp. 20, 23).

[28] Guanyu Gao, Jie Li, and Yonggang Wen. «DeepComfort: Energy-Efficient Thermal Comfort Control in Buildings Via Reinforcement Learning». In: *IEEE Internet of Things Journal* 7.9 (2020), pp. 8472–8484. DOI: 10.1109/JIOT.2020.2992117 (cit. on pp. 20, 23).

[29] Wanlin Bo, Xinli Wang, Gang Jing, Haojin Xu, and Lei Jia. «Comfort and energy management of multi-zone HVAC system based on Multi-Agent Deep Reinforcement Learning». In: *2023 IEEE 18th Conference on Industrial Electronics and Applications (ICIEA)*. 2023, pp. 1641–1646. DOI: 10.1109/ICIEA58696.2023.10241916 (cit. on pp. 20, 23).

[30] Chenhui Fu and Yunhua Zhang. «Research and Application of Predictive Control Method Based on Deep Reinforcement Learning for HVAC Systems». In: *IEEE Access* 9 (2021), pp. 130845–130852. DOI: 10.1109/ACCESS.2021.3114161 (cit. on pp. 20, 23).

[31] Alberto Silvestri, Davide Coraci, Silvio Brandi, Alfonso Capozzoli, Esther Borkowski, Johannes Köhler, Duan Wu, Melanie N. Zeilinger, and Arno Schlueter. «Real building implementation of a deep reinforcement learning controller to enhance energy efficiency and indoor temperature control». In: *Applied Energy* 368 (2024), p. 123447. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2024.123447. URL: https://www.sciencedirect.com/science/article/pii/S0306261924008304 (cit. on pp. 20, 23).

[32] Simon Schmitz, Karoline Brucke, Pranay Kasturi, Esmail Ansari, and Peter Klement. «Reinforcement Learning for Residential Heat Pump Operation». In: URL: https://api.semanticscholar.org/CorpusID:270561374 (cit. on pp. 20, 23).

[33] Parastoo Delgoshaei, Amanda Pertzborn, and Mohammad Heidarinejad. «Reinforcement Learning for Building Management Systems». In: (2021) (cit. on pp. 20, 23).

[34] Liang Yu, Weiwei Xie, Di Xie, Yulong Zou, Dengyin Zhang, Zhixin Sun, Linghua Zhang, Yue Zhang, and Tao Jiang. «Deep Reinforcement Learning for Smart Home Energy Management». In: *IEEE Internet of Things Journal* 7.4 (2020), pp. 2751–2762. DOI: 10.1109/JIOT.2019.2957289 (cit. on pp. 21, 23).

[35] Mark J Willis. «Proportional-integral-derivative control». In: *Dept. of Chemical and Process Engineering University of Newcastle* 6 (1999) (cit. on p. 28).

[36] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. «Optuna: A Next-generation Hyperparameter Optimization Framework». In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019 (cit. on p. 30).